[TITLE PAGE]

Multi-Objective Reinforcement Learning for Autonomous Drone Navigation in Urban Areas with Wind Zones

Jiahao Wu[1], Yang Ye[2], Jing Du, Ph.D.[3*]

[1]Ph.D. Student, Informatics, Cobots and Intelligent Construction (ICIC) Lab, Department of Civil and Coastal Engineering, University of Florida, Gainesville, FL 32611; e-mail: jiahaowu@ufl.edu

[2] Ph.D. Candidate, Informatics, Cobots and Intelligent Construction (ICIC) Lab, Department of Civil and Coastal Engineering, University of Florida, Gainesville, FL 32611; e-mail: ye.yang@ufl.edu

[3]Professor, Informatics, Cobots and Intelligent Construction (ICIC) Lab, Department of Civil and Coastal Engineering, Univ. of Florida, Gainesville, FL 32611 (corresponding author); Email: eric.du@essie.ufl.edu

Reinforcement Learning for Autonomous Drone Multi-Objective Navigation in Urban Areas with Wind Zones

ABSTRACT

Drones can be used for tasks such as data collection and logistics in civil engineering. Current research on autonomous drones mainly focuses on planning a safe path and avoiding obstacles in a static environment. However, navigating a drone in complex environments like urban areas involves many dynamic constraints, such as building layout, winds, and signal coverage, which are interdependent. The wind factor is the most important among these environmental factors, which may cause a drone to lose control or even crash. This paper presents a multi-objective navigation reinforcement learning algorithm (MONRL) for the drone to navigate and avoid obstacles in an unknown environment when dynamic wind zones present, with only imagery data about the building layouts. Based on a deep reinforcement learning and memory architecture, the drone develops policies to prioritize navigation decisions, optimizing the path while minimizing negative impact of winds with only sparse sensor data, in our case, camera inputs. By leveraging the advantages of the proposed method in estimating the environmental factors from previous trials, no aerodynamic force sensors are needed for the drone to develop effective strategies to navigate to target while counteracting to milder winds and dodging aways from stronger winds. The proposed method was tested in a virtual environment, and a real model of New York City. The method is expected to contribute to new automation algorithms for urban aerial logistics and future automated civil infrastructure inspection.

KEYWORDS: Autonomous drones; Reinforcement Learning; Policy network; Wind simulation

1. INTRODUCTION

Unmanned Aerial Vehicles (UAVs), or drones, have gained increasing attention and popularity in many industrial applications due to their flexibility, versatility, and efficiency in disaster response [54], search and rescue [42], and environmental monitoring [15]. Most UAVs are piloted by a human operator from a distance, while recent advances in control algorithms and sensor systems have started to support autonomous drones in complex tasks and workplaces

[24]. Autonomous drones can plan out their path and achieve the flying goals automatically based on dynamic environmental conditions from the onboard or remote sensors, enabling them to reach distant areas or access difficult or dangerous locations that would be challenging or impossible for humans to access [51]. In addition, autonomous drones can free human pilots from repetitive tasks such as logistics delivery, helping the growth of industries despite the shortage of human pilots [59]. Especially, it is expected that the scaled adoption of autonomous drones in the logistics industry can help reduce traffic congestion and carbon emissions by delivering goods through the air instead of on the ground [53]. These drones can also be equipped with a variety of sensors and cameras, such as thermal imaging and LiDAR, to collect high-quality data and imagery on the fly, supporting future tasks such as aerial mapping, surveying, and monitoring [22] [65] [21].

The key to the safe and effective implementation of autonomous drones in logistics is path planning, i.e., determining the optimal route for a drone to follow from its starting point to its destination while satisfying various requirements, such as avoiding obstacles and minimizing travel time [34]. Most existing path planning methods for the drone's navigation system mainly focus on identifying the shortest route and avoiding obstacles [39]. While in real-world applications, dynamic environment conditions (e.g., weather conditions) could have unexpected impacts on the navigation controls of autonomous drones. For example, in urban areas surrounded by tall buildings, the changing aerodynamic conditions can significantly impact the desired outcomes of drone navigation. Another example is that unexpected turbulences occurring between two close buildings may influence the stabilization of a drone [12]. On a larger scale, the layout of buildings in a region, such as a metropolitan district, can affect the wind zone distribution in a complicated way, making a pre-determined drone path plan less optimal [11]. Dynamic winds happening anywhere in a large region, including strong gusts of wind, can cause the drone to lose control or even crash. Even milder winds can affect the maneuverability of the drone, causing it to drift away from its intended path. Lastly, headwinds can impact the drone's battery life and flight time, as the drone requires using more power to compensate for the wind resistance. All of these dynamic environmental conditions make the traditional path planning of autonomous drones less effective in achieving goals in a safe way.

This paper proposes a multi-objective navigation reinforcement learning (MONRL) algorithm for the drone's path planning while safely across the normal wind zones and dodging the strong wind zones. A strong wind zone refers to an area where the wind speed exceeds the safe allowance of a particular drone, which could cause clashes, while the normal wind zones would cause drafting problems and affect the ability of the drone to maintain the desired path. The goal is for an autonomous drone to develop a successful strategy to adapt to aerodynamic conditions while accomplishing navigation tasks with a learning approach. To simulate a more realistic scenario, in our investigation, we consider only an onboard camera as the sole sensor for autonomous drones to sense the environment and learn from the errors. The rationale is that there is a latent relationship between the layout of buildings and the distribution of wind zones, and we hope that the proposed MONRL can still learn effective strategies based on only sparse information about the environment (i.e., visual information). During the MONRL training, we implemented curriculum learning to train the model and used a 3D bounding box algorithm to detect the obstacles based on the picture captured by the camera. Curriculum learning is a machine learning technique that involves training a model on increasingly complex or difficult examples over time. The idea is to gradually increase the difficulty of the training data so that the model can learn more efficiently and effectively. Besides, we develop a conversion algorithm to quantify the wind zone distribution around tall buildings based on computational fluid dynamics (CFD) results. After the training, the drone should be able to sense multiple environmental and physical conditions of the near field, and develop dynamic policies for prioritizing the navigational decision to optimize the path while minimizing the negative environmental impact. The remainder of this paper introduces the point of departure of our study, the proposed method, case studies of the MONRL training, and a validation case based in Manhattan, NYC.

2. RELATED WORKS

2.1. Autonomous Drones Navigation

Autonomous drones have gained significant attention due to their ability to operate without direct human intervention [24]. The versatility of autonomous drones has resulted in a wide range of applications across industries, including agriculture, disaster management, infrastructure inspection, and wildlife monitoring [50] [4] [18] [38]. Autonomous drones employ various self-

navigation methods, such as GPS-based, vision-based, and sensor-based navigation for path planning and flying controls. GPS-based drone navigation is considered the most common and mature method, which relies on GPS data for navigation through pre-defined waypoints [35]. In most cases, a global map that provides a high-level representation of the navigation mission space is available, allowing the drone to plan its trajectory and avoid obstacles more efficiently [57]. Even for partially unknown and dynamic 3D environments, access to GPS data can still enable hybrid navigation methods to facilitate self-governing drone operations, such as rapid reflexive reactions to newly detected obstacles [20].

However, many drone missions are performed in GPS-denied environments characterized by tall buildings or dense plants, limiting the use of GPS-based navigation [64]. As a result, vision-based or sensor-based techniques are explored. Vision-based navigation utilizes computer vision algorithms and onboard cameras to perceive and analyze the environment, enabling drones to avoid obstacles and follow paths [5]. Examples of vision-based navigation systems include Simultaneous Localization and Mapping (SLAM) [66] and Visual-Inertial Odometry (VIO) [61]. Sensor-based navigation techniques leverage sensors like LiDAR, radar, and ultrasonic to measure distances between the drone and surrounding objects, which is useful in environments with limited or impaired visual navigation [13]. These techniques often employ sensor fusion, combining data from multiple sources to enhance the accuracy and robustness of the navigation system [36]. Recently, the use of machine learning has greatly improved the capabilities of autonomous drones in complex environments [37]. For example, deep learning techniques, such as Convolutional Neural Networks (CNNs) [6] and Recurrent Neural Networks (RNNs), have been applied to enhance vision-based navigation by enabling more accurate object recognition, segmentation, and tracking [71]. Despite these advances, challenges remain for autonomous drones' navigation in complex and dynamic environments. Obstacle detection and avoidance in areas such as urban settings or dense forests is critical, but developing effective algorithms remains challenging due to the variety of objects and their proximity [40]. The following section unfolds the details of challenges related to drone navigation in dynamic environments and introduces the existing efforts in addressing the self-navigation in dynamic environments.

2.2. Self-Navigation in Dynamic Environments

One of the primary challenges in autonomous drone navigation is operating in dynamic environments. The challenge roots from the difficulty of environmental sensing to incorporate a large amount of dynamic information and the corresponding information processing and computational demands. On the one hand, dynamic environments with moving objects (e.g., vehicles and pedestrians) and adverse environmental conditions (e.g., poor lighting conditions) require advanced sensing methods to enable real-time adaptation and accurate movement prediction [41]. On the other hand, with the increasing input of sensor data, robust data processing, and control algorithms are needed to convert the external information about the environment into safe and effective drone navigation and path planning decisions. As for the sensing techniques for autonomous drones, a popular solution is the use of LiDAR-based sensors. Bolognini and Fagiano [13] developed a real-time algorithm for LiDAR-based drone control, enabling navigation in unknown environments. El-Sheimy and Li [19] proposed an innovative algorithm to reduce computational cost and energy consumption for indoor LiDAR target detection. Aldao, González-de Santos and González-Jorge [2] evaluated LiDAR's ability to detect dynamic obstacles' location and speed with positive results. Patrik, Utama, Gunawan, Chowanda, Suroso, Shofiyanti and Budiharto [48] used course-over-ground information and an IMU sensor for outdoor drone navigation without obstacles, relying on GNSS data. In addition, camera-based methods provide a more detailed visual representation of the environment, allowing drones to respond to a broader range of obstacles. Chakravarty, Kelchtermans, Roussel, Wellens, Tuytelaars and Van Eycken [16] used a single front-facing camera and trained a CNN for depth perception, guiding the drone to avoid obstacles. Sani and Karimian [55] proposed a complete solution for indoor drone navigation using two cameras, a Kalman filter, and an inertial sensor. García Carrillo, Dzul López, Lozano and Pégard [25] placed a stereoscopic camera on a drone and trained a CNN to process depth maps, detecting and locating drones up to eight meters away with a 10% error rate. Jung, Hwang, Shin and Shim [32] introduced a new CNN to estimate the locations of key objects in drone navigation, applying a line-of-sight guidance algorithm for drone direction, validated on cost-efficient hardware. Baskar and Gorodetsky [9] proposed a new dataset of wind simulation to validate the fixed-wing UAV route choice when considering the energy efficiency. Nathanael, Wang and Low [47] used steady-state RANS

simulation generated by Open FOAM to simulate the wind field in Singapore and then researched its impact on the flight of the Unmanned Aerial Systems (UAS).

Literature has also been investigating advanced algorithms capable of processing additional sensor information and functioning effectively in diverse conditions [68]. Sun, Li, Wang and Zhao [62] proposed the use of the Rapidly-exploring Random Tree (RRT) algorithm for path planning in complex environments. RRT has become a popular choice for UAV path planning due to its ability to quickly explore search spaces and generate feasible paths. Later, Wen, Su, Ma, Zhao and Zhang [69] presented an enhanced version of RRT, called Safe-RRT, which guarantees safe and optimal paths for UAVs in cluttered environments. Optimization-based methods have also gained traction for addressing path planning problems. Albert, Leira and Imsland [1] applied mixed-integer linear programming (MILP) to generate collision-free paths for multiple UAVs simultaneously. Their method demonstrated the ability to handle complex scenarios with multiple UAVs and obstacles efficiently. Similarly, Huang, Zhou, Ran, Wang, Chen and Deng [28] investigated the use of particle swarm optimization (PSO) for drone path planning, showing its effectiveness in generating optimal paths in real-time. Recent developments in machine learning have led to new approaches for UAV path planning and navigation. Fan, Chu, Zhang, Song and Li [23] explored the use of imitation learning to transfer expert knowledge for drone navigation, leveraging human demonstration data for improved performance. In addition to path planning algorithms, sensor fusion, and localization play a crucial role in autonomous drone navigation in a dynamic environment. Cadena, Carlone, Carrillo, Latif, Scaramuzza, Neira, Reid and Leonard [14] provided a comprehensive review of simultaneous localization and mapping (SLAM) techniques for drones, emphasizing the importance of integrating vision, inertial, and other sensors for accurate localization. Furthermore, Baca, Hert, Loianno, Saska and Kumar [8] demonstrated the effectiveness of vision-based navigation for UAVs, enabling them to operate in GPS-denied environments. Collision avoidance and obstacle detection are essential components of safe drone navigation. Imrane, Melingui, Mvogo Ahanda, Biya Motto and Merzouki [29] utilized a bio-inspired approach based on the artificial potential field (APF) method for obstacle avoidance, showing its effectiveness in real-world scenarios. Meanwhile, Andersson, Wzorek, Rudol and Doherty [3] developed a predictive collision avoidance algorithm using Bayesian methods, which offered improved performance of drone navigation in dynamic environments.

What is worth noting, many of the advanced control algorithms and machine learning techniques for autonomous drone self-navigation require excessive data from the sensing systems about the dynamic environment [63]. We consider a more realistic scenario where not all relevant sensor data is available or very limited, while the primary source of data is still from imagery sensors [40]. For example, for an algorithm that can control a drone through heavy wind zones (aerodynamic optimization), we consider the situation when wind sensors or similar pressure sensors are not accessible, but only onboard RGB cameras are available. The following section introduces the potential of reinforcement learning in tackling issues with only sparse data.

2.3. Reinforcement Learning for Self-Navigation with Limited Data

Reinforcement learning (RL) is a branch of machine learning methods that involves training an agent to make decisions based on interactions with an environment [33]. The agent learns to maximize a numerical reward signal that it receives from the environment in response to its actions. This trial-and-error process allows the agent to discover the optimal policy for achieving its goal [43]. Compared to other machine learning approaches, RL does not need a training set, and the agent will adjust the action based on continuous feedback to maximize the final reward. One key advantage of RL is its ability to learn from experience and adapt to changing environments, which is key to autonomous drone navigation in dynamic environments. Pham et al. used a PID+Q-learning algorithm as the training strategy to train the drone to navigate in an unknown environment without any obstacles [49]. They created a simulation environment for the drone to learn, then tested the navigation system in the real world. The method's validation and simulation yielded comparable outcomes, indicating that the drone could adeptly learn to navigate through its surroundings without relying on any pre-established model. Hodge et al. built a 2D training environment in Unity and tried safely training the drone to arrive at the destination while avoiding all the obstacles [27]. They used LiDAR sensors to obtain the obstacle information near the drone and then combined RL with the Proximal Policy Optimization algorithm to train the drone navigation system. Muñoz et al. used the DDQN to train a drone in the 3D simulation environment and proposed a new neural network named JNN to learn obstacle information based on depth cameras [45]. Shin et al. created a virtual environment for drone training and applied the U-Net as the reinforcement learning network

during the training [60]. To test the model's validity, they transferred it to the real drone, then tested it in a reconfigurable maze that was similar to the virtual maze. They used only one RGB camera as the model's input, which still performed well in the real-world test. Xue and Gonsalves suggested a novel deep reinforcement learning approach that used Variational Autoencoder (VAE) for image data preprocessing [70]. This approach facilitated fast and effective learning for UAVs to avoid obstacles without depending on sensors other than the depth camera. The advantage of this method over other visual obstacle avoidance algorithms was that it allowed for obstacle avoidance in a continuous action space and eliminated the need for complex image recognition networks during RL training. Tong et al. proposed an enhanced Deep Reinforcement Learning (DRL) approach to address the inherent limitations of UAV navigation in dynamic environments [63]. The proposed method involved a distributed DRL framework that breaked down the navigation task into two simpler sub-tasks, each tackled by a Long Short-Term Memory (LSTM) based DRL network using a portion of the interactive data. Additionally, a clipped DRL loss function is introduced to seamlessly integrate the two sub-solutions into a single comprehensive solution for UAV navigation.

In summary, RL methods offer significant advantages for autonomous drones in self-navigation, particularly when certain sensor data is missing. These advantages stem from RL's ability to learn from experience, adapt to dynamic environments, exhibit inherent robustness, generalize knowledge across environments, incrementally learn, and scale with deep learning techniques [46]. Through trial-and-error interactions with the environment, RL enables drones to develop navigation policies that account for missing data. RL's adaptability ensures effective navigation in dynamic and unpredictable scenarios, even with limited sensor data. The inherent robustness of RL algorithms allows drones to prioritize actions leading to successful navigation, compensating for missing information. Additionally, RL's generalization capabilities enable drones to apply learned navigation policies to similar environments without extensive retraining [17]. Incremental learning allows RL-based drones to continually refine their navigation policies in response to changing or missing sensor data, while the integration of deep learning techniques, such as Deep Q-Networks (DQNs) and Proximal Policy Optimization (PPO), ensures scalability and the ability to learn complex navigation policies in various environments [7]. Overall, RL provides a powerful solution for autonomous drones to overcome challenges posed by incomplete or missing sensor data and achieve successful self-navigation.

2.4. Multi-Objective Reinforcement Learning for Self-Navigation

In recent years, Multi-Objective Reinforcement Learning (MORL) has emerged as a pivotal approach to tackle complex decision-making problems involving multiple conflicting objectives. MORL techniques offer a framework to optimize agents' policies in environments where multiple objectives must be simultaneously considered and balanced. At its core, MORL seeks to find a balance between different objectives by exploring the trade-off space. Algorithms within the MORL framework aim to generate policies representing diverse solutions, each offering a distinct compromise between objectives. This set of policies is known as the Pareto frontier, and it empowers decision-makers to choose the most suitable policy according to their priorities. Ramezani Dooraki and Lee [52] built a multi-objective reinforcement learning network to train the drone navigation system in a dynamic environment, which includes static and moving obstacles. Shantia, et al. [58] proposed a two-stage training methodology for drone navigation based on visual input and proved that the performance for the multi-objective training network was better than the single-objective. MORL addresses scenarios where a single optimal solution may not suffice due to competing goals or trade-offs. This is particularly relevant in our context, where our problem involves optimizing drone navigation while minimizing wind impact and maximizing safety. MORL provides a robust methodology to navigate these challenges and derive policies that lead to a Pareto-optimal set of solutions, allowing decision-makers to make informed choices based on their preferences. By incorporating MORL into our study, we leverage its capabilities to tackle the intricate trade-offs inherent in our multi-objective drone navigation problem. MORL equips our approach with the flexibility to handle conflicting objectives, ensuring that our solution is not only technically robust but also practically adaptable to real-world scenarios.

In summary, Multi-Objective Reinforcement Learning offers a well-established framework to address multi-dimensional optimization challenges, making it an ideal choice for our problem of optimizing drone navigation while considering wind fields in urban environments. The adoption of MORL empowers us to explore the trade-off space and present a set of solutions that cater to a spectrum of decision-makers preferences.

3. METHODOLOGY

3.1. Overview

This paper proposes a MONRL method for drones to autonomously navigate through dynamic wind zones while reaching the given target and avoiding obstacles only based only on local information obtained from the RGB cameras and GPS equipment. Unlike previous studies that assume that global information about the environment is known, or relevant sensors are available for obtaining all necessary information, our approach assumes that only GPS equipment and limited imagery sensors, such as RGB cameras, are accessible. The dynamic and complex interactions between the built environment and the wind zones would be captured by the system while learning.

The framework showcased in **Fig. 1** provides a clear outline of the sequential steps that constitute the proposed Multi-Objective Navigation Reinforcement Learning (MONRL) technique, which is designed to enhance the navigation capabilities of drones. Here's an in-depth breakdown of the process:

The first step involves the sensing mechanisms employed by the drone agent. Equipped with two core sensing devices, namely the RGB camera and GPS equipment, the drone gathers crucial information from its environment. The RGB camera can offer a three-dimensional view of the surroundings by applying bounding box algorithms. This facilitates the drone's understanding of the landscape's geometry, helping it gauge the proximity of objects. The GPS equipment, on the other hand, not only provides accurate positional data for navigation but also aids in determining wind force.

In the second step, the drone utilizes the information it collects to estimate wind force through drifting information. When subjected to wind forces, the drone deviates from its intended trajectory. By comparing the anticipated position (based on movement commands) with the actual position (from GPS data), the drone calculates the extent of drifting deviation. This deviation information is then utilized to gauge the wind force acting upon the drone. Analyzing the degree and direction of drift provides valuable insights into the magnitude and orientation of the wind.

The third step involves the integration and pre-processing of data. The data acquired from the RGB camera and the estimated wind force from GPS are merged and subjected to preparatory

procedures. This ensures that the data is formatted in a manner conducive to its utilization within the subsequent reinforcement learning network. Potential processes here include data normalization and flattening, which streamline the input data.

In the fourth step, the integrated data is fed into a complex reinforcement learning network. This network is characterized by its deep architecture, encompassing different layers designed to handle diverse information. Convolutional layers process the depth information from the camera, while dense layers manage drifting and wind force data. The core objective of this network is to learn optimal navigation strategies for the drone across various environments, taking into account obstacles derived from geometry information as well as environmental factors such as wind.

The ultimate output of this network is a set of actions or decisions that guide the drone's real-time movement. These actions are carefully formulated based on the drone's assimilation of data and its understanding of the environment. In essence, the MONRL framework enables the drone to navigate intricate landscapes, making informed decisions that optimize its movements by intelligently adapting to the prevailing conditions.

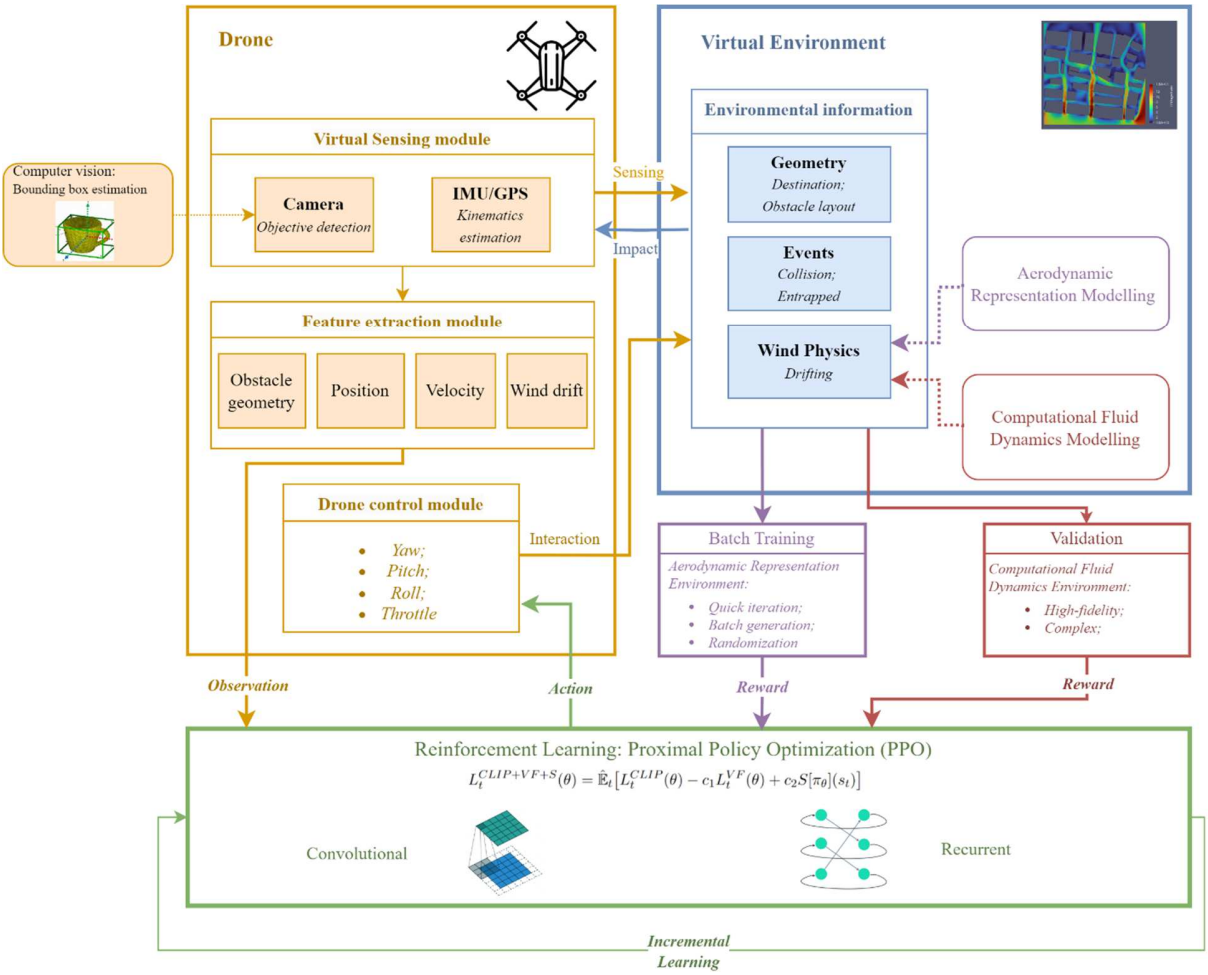


Fig. 1 Framework of proposed MONRL method

Among these inputs, the external inputs to the drone agent are solely processed camera data. We use a 3D bounding box algorithm to convert the imagery data about the built environments (such as buildings) captured by the front camera to 3D vectors [44]. The training can be divided into two phases. The first phase is to train the drone in the simulation environment with no wind zones by using the Proximal Policy Optimization (PPO) algorithm and the LSTM structure [26]. The PPO algorithm is a RL algorithm that offers advantages such as stability, efficiency, sample efficiency, and robustness, making it a powerful and reliable algorithm for learning good policies with relatively few samples [56]. In addition, LSTM is used to remember information from previous time steps in a sequence. This recurrent model can capture temporal dependencies in the input data and help the drone make more informed decisions. The experiment was performed

in multiple phases, gradually adding more environmental factors to the simulation. This is to test how incorporating environmental factors could alter the original decisions made by the drone AI. Specifically, the purpose of the first phase was to validate the feasibility of the RL architecture for identifying the shortest path from the starting point to the target point without considering other conditions, such as environmental factors. It also supports a simulated navigation system that grants the drone an obstacle avoidance ability in windless conditions. In the second phase, after the drone's basic navigation and obstacle avoidance system was built up (i.e., phase one), the wind information was added to examine how MONRL could facilitate navigation decision-making in an environment with more conditions, in our study, dynamic winds. Therefore, we added a simulation of the wind dynamics to the MONRL training during the second phase, i.e., adding aerodynamic forces in the simulated environments to replace the windless condition in the prior phase, especially when the drone approaches buildings. As will be shown later in the results, we found that incorporating environmental factors indeed changes the decision of the AI. Such a comparison is only possible with a multi-phase study. Because MONRL in our study is a multi-objective training, i.e., reaching the target, avoiding obstacles, and counteract the negative effects of dynamic wind zones on the gesture and path stability of the drone, a multivariate reward function is utilized for training the drone to learn to dodge the strong wind zones, react to miler wind zones, avoid visible obstacles, in addition to finding a new route to the target when needed. After the two training phases, real-world data is used to validate the training results. We use OpenFOAM [30], an open-source computational fluid dynamics (CFD) software, to create the CFD results for a metropolitan district in New York City. Once the CFD result is generated, the data is transferred to Unity and used as the wind zone input in the test environment.

3.2. Deep Reinforcement Learning Network

The core of the proposed MONRL method is a deep reinforcement learning (DRL) network that enables the robust learning of the drone to navigate in a dynamic environment. We designed a DRL network with seven layers, as shown in **Fig. 2**. The first layer is the input layer with 87 inputs. Among the 87 inputs, 72 inputs are assigned to store the local geometric information. In our case, it refers to the bounding box results from the drone-carried imagery sensor to represent obstacles (mainly buildings) in the visible proximity of the drone. For each obstacle, the 3D bounding box algorithm can return the 8 space positions of all its vertex, and each space position

contains three values, which represent the XYZ position data, respectively. In this way, we can use a short array to represent the key geometric features of a building (i.e., the contour of a building). The other 12 inputs are used to record all the information related to the drone agent. The information includes the distance between the drone and the target, the drone's current speed, the drone's position, and wind data from a given aerodynamic map. Each of the inputs is a vector that contains three values. To teach the drone to avoid the stronger wind zones, we allocated three inputs to enable the drone to memorize the position of the last strong wind zone that can be used in the reward function.

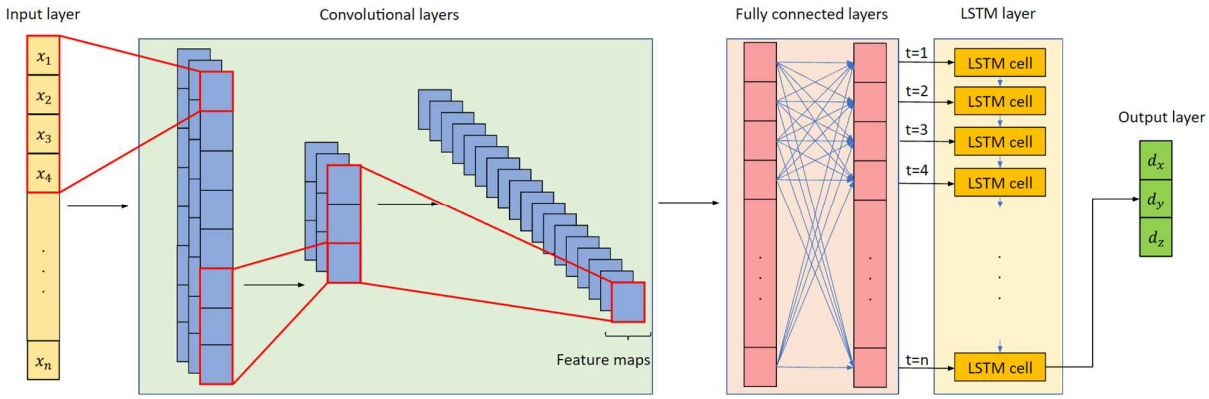


Fig. 2 Deep reinforcement learning network architecture

Moving forward from the input layer, there are two convolutional layers. Since the vertex information of a bounding box is interdependent, using convolutional layers helps extract the key features. Then there are 2 hidden layers, each with 256 nodes. These layers allow the network to capture more complex relationships between the input features. A single LSTM layer is used after the hidden layers. The advantage of this recurrent layer is that it can effectively capture long-term dependencies and temporal patterns in sequential data, which can let the drone avoid an infinite loop. The final layer is the outcome layer with three outputs. The output signifies the intended direction for the drone's movement along the x, y, and z axes. This vector comprises three values spanning from 0 to 1. Given its adjustable magnitude, the drone's behavior can be altered by modifying this magnitude. The ultimate velocity will be determined by the product of the output and the maximum speed of the simulated drone during the simulation.

In the simulation, the wind force would directly affect the drone's acceleration rate. Therefore, the agent must consider the movement decision based on the wind information. In our experiment, we assumed that the drone could fully follow the commands from the RL algorithm. In addition, our model did not consider the modeling of sensor noises other than the visual sensors, i.e., the 3D bounding box recognition based on the virtual visual sensors. We admit that this is a simplified assumption as in real world, drone controls could be affected by a variety of factors and thus, the control could be off. Nonetheless, the scope of this study is to examine that if the incorporation of multiple objectives at the same time, especially the aerodynamic forces, in the drone navigation decisions, could yield paths that are different than a direct consideration of shortest path. As a result, we focused on a more controlled experiment in the simulator while ignoring the real-world control challenges. It is recognized as one of the limitations in the discussion.

In general, its decision network can be defined as Eq (1):

$$D_t = F(b_t, d_t, v_t, w_t, p_t, D_{t-1}) \quad \text{Eq (1)}$$

Where D_t is the speed decision for this step, b_t represents the obstacle data (relative position, dimensions) from the 3D bounding box algorithm, d_t represents the relative distance in all directions between the agent and the target, v_t represents the current speed of the drone, w_t is the wind data based on the drifting of the drone, p_t refers to the local position of the drone, which is used to control the drone fly in the designed scope, D_{t-1} is the movement decision for the last step, which is the velocity.

3.3. Reward Function

According to the proposed MONRL, the goal of the agent is to learn a policy that maximizes its cumulative reward with a series of sub-objectives over time by taking actions in an environment and receiving feedback in the form of rewards. The reward function design is based on the project goal, previous experience, and the drone's performance during the experiment. The reward system that we designed in this experiment is as Eq (2).

$$R = r_{distance} + r_{target} + r_{wind} + r_{collision} + r_{stuck} + r_{timeout} + r_{time} \quad \text{Eq (2)}$$

Where R represents the total reward that the drone can receive. It is a linear combination of a set of award terms: $r_{distance}$ represents the reward depending on how close the drone moves towards the target, r_{target} is a binary reward if the drone safely arrives at the target, r_{wind} refers to the reward if the drone successfully avoids the strong wind zone, $r_{collision}$ represents the cumulative penalty when the drone has a collision with any obstacle, r_{stuck} is a penalty when the drone is stuck at an infinite loop in navigation, r_{time} is a penalty term based on the total time spent on the navigation task, and $r_{timeout}$ is a binary term to represent penalty when the drone uses out all the time. To simplify, we did not assign any weight to the sub-objectives. It deserves future investigations into how changing the priorities of each of the sub-objectives would affect the overall performance of the MONRL in drone navigation. The followings provide more detailed information about each of the sub-award terms.

The first reward term encourages the drone to move towards the target. It is a continuous reward that can guide the drone at every step in one epoch. In our experiment design, the positions of the drone and target change every epoch. In order to normalize the distance reward for every epoch, we design the distance reward function as Eq (3).

$$r_{distance} = \sum_{1}^{T} \frac{dis_t - dis_{t-1}}{dis_0} \times 100 \qquad \text{Eq (3)}$$

Where $r_{distance}$ represents the rewards the drone can receive, dis_t is the distance between the drone and the target at this step,dis_{t-1} is the distance between the drone and the target at the last step, dis_0 is a fixed value that represents the distance between the drone and the target at the start of the current epoch.

The second reward term provides a positive reward for reaching the target quickly and safely while avoiding stronger wind zones and obstacles, as shown in Eq (4). This can be a comprehensive approach to incentivize the drone to learn to navigate effectively in the wind zone.

$$r_{target} = \begin{cases} +10 & if\ arrives\ target \\ 0 & otherwise \end{cases} \qquad \text{Eq (4)}$$

The third reward term encourages the drone to find alternative routes to avoid the stronger wind zones. For every epoch, if the drone encounters a strong wind zone and avoids it, it will receive a positive reward, as indicated in Eq (5).

$$r_{wind} = \begin{cases} +10 & if\ avoids\ strong\ wind\ zone \\ 0 & otherwise \end{cases} \quad \text{Eq (5)}$$

The fourth reward function is a negative term that penalizes the drone once there is a collision. In the real world, collisions with obstacles can be disastrous for the drone. Besides, the epoch will stop immediately if a collision occurs, thus the drone will not receive any further rewards and will receive a very low total reward. By providing a significant penalty for collisions, the drone will learn to avoid them (see Eq (6)).

$$r_{collision} = \begin{cases} -10 & if\ collision\ occurs \\ 0 & otherwise \end{cases} \quad \text{Eq (6)}$$

The fifth reward term is a penalty for the drone flying around a strong wind zone, as shown in Eq (7). It is a negative reward for every step the drone spends in a strong wind zone. This will encourage the drone to minimize the time it spends in those areas and encourage it to find a route around them. We set this function as a linear function. The closer the drone is to a wind zone, the larger the penalty it will have. This will encourage the drone to move away from the strong wind zones.

$$r_{stuck} = \begin{cases} -0.1 \times \dfrac{dis_{wind}}{20}\ per\ step & if\ in\ strong\ wind\ zone \\ 0 & otherwise \end{cases} \quad \text{Eq (7)}$$

Where r_{stuck} represents the penalty the drone will receive, dis_{wind} is the distance between the drone and the stored strong wind position.

The sixth reward term is a penalty if the drone uses all the time but does not arrive at the target, as shown in Eq (8). Battery consumption is vital to drones as they rely on onboard power supplies to power their propulsion systems, sensors, cameras, and other components. If the drone spends too much energy on a one-trip flight, it may lose power and be unable to complete its tasks or return to the start location. Since it does not complete the tasks, a negative penalty is given to the drone.

$$r_{timeout} = \begin{cases} -10 & if\ step\ reaches\ maxmium \\ 0 & otherwise \end{cases} \quad \text{Eq (8)}$$

Finally, the last reward function is the time penalty function for the drone. As shown in Eq (9), the penalty increases as the time spent on the mission increase. The purpose of this penalty is to let the drone choose the fastest way to the target.

$$r_{time} = 0.001\ per\ step \quad \text{Eq (9)}$$

3.4. 3D Bounding Box Algorithm

It should be noted the advancements in stereoscopic vision systems offer extended range and improved computational efficiency. The compactness and enhanced capabilities of systems like the Intel RealSense indeed make them suitable for a variety of applications, including construction surveying. The compactness and enhanced capabilities of systems like the Intel RealSense indeed make them suitable for a variety of applications. In our study, we primarily relied on RGB information for drone navigation not only due to its technical feasibility but also its cost viability. RGB cameras, widely used in various applications, are generally more cost-effective compared to depth cameras. While advanced depth cameras, including models like the RealSense, offer superior capabilities, they often come at a higher cost. This factor is particularly crucial in the selection of sensory equipment for drones, especially for large-scale or budget-sensitive projects. Therefore, we use RGB camera combined with 3D bounding box to do the obstacle detection tasks. A 3D bounding box is a rectangular prism that surrounds a 3D object, such as a building. The bounding box is defined by its three dimensions - length, width, and height - as well as its position and orientation in space. 3D bounding boxes are commonly used in computer vision and robotics applications to represent the spatial extent of objects in a scene [44]. In general, for a 3D box projected on the image, we can always wrap it with a 2D box. Mousavian et al. proposed a deep learning model to predict the 3D bounding box position based on the 2D bounding box results [44]. They built up constraints between the 2D and 3D bounding box images, called point-to-side correspondence constraint equations. One of the equations is as follows.

$$x_{min} = \begin{pmatrix} & \frac{d_x}{2} & \\ K[R\ T]^{-} & \frac{d_y}{2} & \\ & \frac{d_z}{2} & \\ & 1 & \end{pmatrix} \quad \text{Eq (10)}$$

Where x_{min} represents the minimum x position of the 2D bounding box, K represents the camera intrinsic matrix, R represents the frame rotation, T represents the frame translation, d_x is the length of the 3D bounding box, d_y is the width of the 3D bounding box, d_z is the height of the 3D bounding box.

Because there are four vertex points in a 2D bounding box, which are x_{min}, x_{max}, y_{min}, y_{max}, a total of four equations of the above form can be written. This is the set of four constraint equations for the 2D bounding box and the 3D bounding box. And there are nine parameters to be solved in this set of equations (six affine transformation parameters and three bounding box dimensions). In our work, we built the 3D bounding box algorithm based on this model. The details of the algorithm are shown in **Fig. 3**:

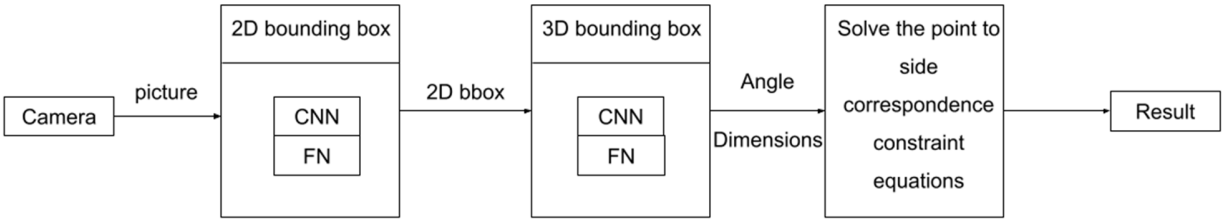


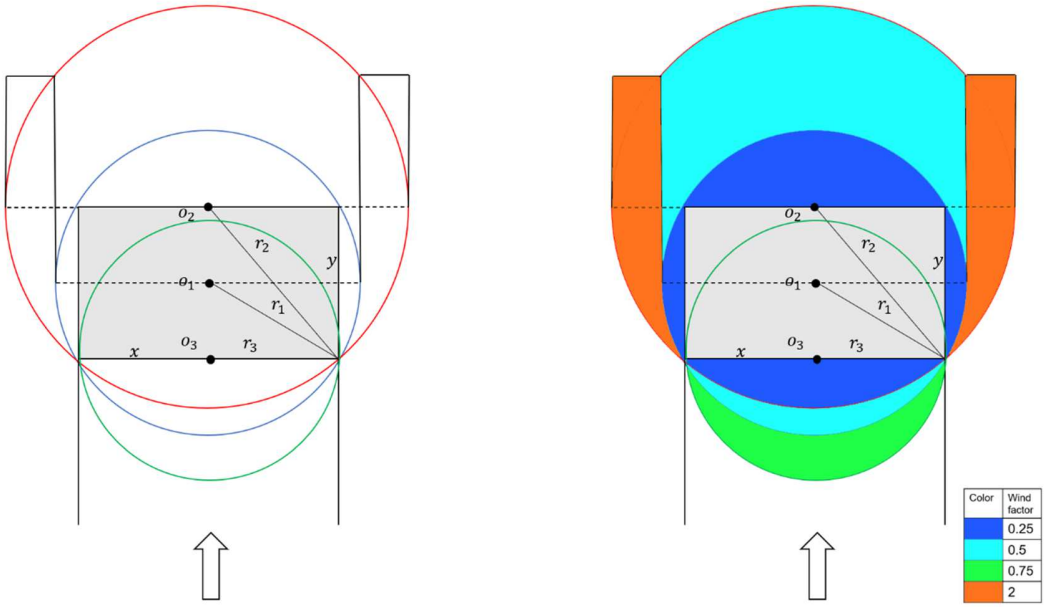
Fig. 3 Bounding box algorithm

The camera on the drone will provide a picture of all the obstacles in front of the drone. In our design, we limit the number of obstacles used in the model to simulate the limited data storage and processing capacities of most drones and stress test the performance of the proposed MONRL method given only sparse data. Specifically, if the number of obstacles captured by the sensor is larger than three, we only consider the three closest obstacles. Once a drone determines any obstacle, we will first apply the 2D bounding box algorithm to compress the raw data. A 2D detector is used to find the bounding box for the obstacles. Then, its results are used as the input

of the 3D bounding box network. This network can convert the convolutional features to the angle and dimensions of the obstacles, which are parameters in the four constraint equations. Finally, since we have already known six of the nine parameters, it is trivial for us to obtain the rest three parameters from the four constraint equations, which is the relative position of the 3D bounding box. By combining the dimensions and relative position of the obstacles, we can calculate all eight vertex positions of the obstacles and transfer them to the reinforcement learning model.

3.5. Aerodynamic Representation Modeling

We also design a framework for representing the aerodynamic characteristics of an environment. Although CFD can be used to simulate aerodynamic forces, we found it to be inefficient for meeting the needs of a large amount of simulation for the DRL algorithm. DRL usually requires thousands of simulated scenarios with changing conditions. The cost of performing CFD for each of them is significant. In contrast, we found that there were modellable aerodynamic features around a given structure if the initial conditions of the wind and the geometric features of the structure were known. Our proposed representation model can generate a wide range of wind conditions, including random gusts and turbulences. This allows the drone to be trained in a variety of changing conditions, making it more adaptable to different weather scenarios. For example, we can quickly magnify the wind factor in our work to create a strong wind zone between two close buildings. This will make the training environment more suitable for our goal. In addition, representation model simulation typically requires less computational time than CFD, which can be important for training large-scale drone fleets. This allows for faster DRL training and more efficient use of computing resources. During the training period, we change the obstacle layout every epoch.



(a) Wind zone geometric model (b) Wind zone with magnitude

Fig. 4 Representation modeling for simplifying aerodynamic features around building

Fig. 4 shows the details of the proposed aerodynamic representation modeling method based on the key features extracted from actual CFD results [31]. Different colors represent different wind factors. This modeling method aims to simulate the wind distribution around a single building in the wind zone. Assume that we have a single obstacle in front of the wind direction in the x-y plane. We set the obstacle's length as x and the obstacle's width as y. Then we need to determine circles with O_1, O_2 and O_3 as the center and r_1, r_2 and r_3 as the radius respectively, where O_1 is the center of the obstacle, O_2 is the middle point of the upper line, and O_3 is the middle point of the under line. Eq (11) – Eq (13) indicates the quantification of the geometric models of a wind zone.

$$r_1 = 0.5 \times \sqrt{x^2 + y^2} \quad \text{Eq (11)}$$

$$r_2 = \sqrt{0.25 \times x^2 + y^2} \quad \text{Eq (12)}$$

$$r_3 = 0.5 \times x \quad \text{Eq (13)}$$

After completing these three circles, we can also identify two tangents of the blue circle and find their intersection with the red circle. Finally, we divide the whole area around the obstacle into three regions. The first region is in front of the obstacle (front region), which contains three subregions. In the front region, the wind force is reduced due to obstructions. Based on the CFD results, we set all the wind directions in this area toward the closest corner of the obstacle. The second region is the sides of the obstacle (side region), where the valley effect may happen, i.e., the wind speed will increase since air cannot pile up in a large amount in a small space [12]. According to CFD results, the wind magnitude in this region is related to the distance between the wind vector and the obstacle. When the wind is closer to the obstacle in this region, its magnitude will be slower. Otherwise, the flow of the current will increase. As a result, the valley effect can occur if two buildings are very close, causing disturbances to the drone. Besides, the direction of the wind is parallel with the direction of the tangent of the red arc. The last region is behind the obstacle (back region). Similar to the front region, the obstructions cause a decrease in the strength of the wind. In summary, the aerodynamic force on the drone can be written as Eq (14) and Eq (15).

$$F_{drone} = F_{wind} \times f_{wind} \quad \text{Eq (14)}$$

$$a_{drone} = \frac{F_{drone}}{w_{drone}} \quad \text{Eq (15)}$$

Where F_{drone} represents the aerodynamic force exerted on the drone, F_{wind} is the origin aerodynamic force, f_{wind} is the wind factor, which depends on the position of the drone relative to the obstacle, a_{drone} is the passive acceleration of the drone, w_{drone} is the weight of the drone. It is important to note that this aerodynamic representation modeling framework is a simplified approximation of the precise CFD results. It may not function well in applications where accurate and precise high-fidelity CFD results are needed. However, the primary purpose of this research is to train the MONRL model to guide the drone to avoid stronger wind zones and counteract the milder winds. And the main function of this representation framework is to provide high-level, low-resolution aerodynamic information for training only. In our test case, we found that such a simplified representation of the actual aerodynamic features worked well in yielding effective MONRL training.

3.6. Proximal Policy Optimization (PPO)

In our architecture, we use Proximal Policy Optimization (PPO) to improve the stability and sample efficiency of policy gradient methods by constraining the size of the policy update in each iteration. PPO is an algorithm used in DRL that has gained significant attention in recent years due to its ability to achieve state-of-the-art performance on a wide range of DRL problems [56]. PPO is a model-free, on-policy DRL algorithm that learns directly from experience using a neural network to represent the policy function. The PPO algorithm aims to optimize a policy function $\pi(a|s)$, which represents the probability of taking an action a in state s. The goal is to maximize the expected cumulative reward from time step 0 to T, which is defined as:

$$J(\theta) = E[\Sigma_t = 0^T \gamma^t \, r_t] \quad \text{Eq (16)}$$

where θ represents the parameters of the policy function, 0^T represents the initial time step in the problem, T is the time horizon, γ is the discount factor, and r_t is the reward received at time step t. It employs a clipped surrogate objective function to update the policy, which helps to ensure stability and prevent catastrophic updates. This function is given by:

$$L(\theta) = E\left[\min\left(\frac{r_t(\theta)}{r_{t_{old}}(\theta)} \times A_t, clip\left(\frac{r_t(\theta)}{r_{t_{old}}(\theta)}, 1-\varepsilon, 1+\varepsilon\right) \times A_t\right)\right] \quad \text{Eq (17)}$$

where $r_t(\theta)$ is the probability of taking the action that was taken at time step t under the updated policy, $r_{t_{old}}(\theta)$ is the probability of taking that same action under the old policy, A_t is the advantage function, and ε is a hyperparameter that controls the size of the clip. The advantage function is defined as:

$$A_t = \Sigma_k = 0^T - 1 \, \gamma^k \left(r_k + 1 + \gamma V(s_k + 1) - V(s_k)\right) \quad \text{Eq (18)}$$

where V(s) is the value function, which estimates the expected cumulative reward from state s onwards, k is a time index that refers to the current time step in the advantage function calculation. By using stochastic gradient descent, the PPO algorithm updates the policy and value function parameters, with the clipped surrogate objective as the loss function. The value function is updated using the mean squared error loss between the estimated value and the actual reward. Overall, the PPO algorithm is designed to balance exploration and exploitation while ensuring stability and avoiding catastrophic updates.

3.7. LSTM for Capturing Temporal Dependencies

In our pilot test, we found that the drone would be easily stuck in the local optimum without memory from previous epochs. This is probably due to the complexity of the environment and the high dimensionality in the DRL input. As a result, we use Long Short-Term Memory (LSTM) to help the drone remember the previous action to avoid getting stuck in a dead-end cycle. LSTM is a type of recurrent neural network (RNN) that is designed to overcome the vanishing gradient problem, which is a common issue with traditional RNNs [26]. It uses a specialized architecture that includes memory cells, input gates, output gates, and forget gates. At each time step t, the network takes an input vector x_t and produces an output vector h_t. The network also maintains an internal state vector c_t, which represents the memory of the network at time step t. The key to LSTM's ability to model long-term dependencies is its memory cell, which is denoted by the variable c_t. The memory cell is updated at each time step using the following equations:

$$f_t = \sigma\left(W_f\ [h_{t-1}, x_t] + b_f\right) \quad \text{Eq (19)}$$

$$i_t = \sigma(W_i\ [h_{t-1}, x_t] + b_i) \quad \text{Eq (20)}$$

$$o_t = \sigma(W_o\ [h_{t-1}, x_t] + b_o) \quad \text{Eq (21)}$$

$$g_t = \tanh(W_c\ [h_{t-1}, x_t] + b_c) \quad \text{Eq (22)}$$

where σ denotes the sigmoid function, tanh denotes the hyperbolic tangent function, W_f, W_i, W_o, and W_c are weight matrices, b_f, b_i, b_o, and b_c are bias vectors, and $[h_{t-1}, x_t]$denotes the concatenation of the previous output $h_{t-1}$and the current input x_t. These equations define four gates that control the flow of information into and out of the memory cell. The forget gate f_t determines how much of the previous memory cell value to retain. The input gate i_t determines how much new information to add to the memory cell. The cell gate g_t computes a candidate memory cell value based on the current input and the previous output. The input gate i_t also determines how much of the candidate value from g_t to add to the memory cell and the output gate o_t controls how much of the current memory cell value to output, which is given by:

$$c_t = f_t * c_{t-1} + i_t * g_t \quad \text{Eq (23)}$$

$$h_t = o_t * \tanh(c_t) \quad \text{Eq (24)}$$

where * denotes element-wise multiplication. These gates control the flow of information into and out of the memory cells, allowing the network to remember or forget information from previous time steps selectively. This allows LSTMs to learn long-term dependencies and to model sequential data effectively.

4. EXPERIMENT

We used a simulator to train drones to navigate and avoid obstacles that offers several advantages over traditional physical training methods. The simulator allows for the creation of complex and varied training scenarios that may be difficult or impractical to replicate in the real world. This enables the drone to encounter a wide range of obstacles and challenges, allowing for more comprehensive and effective training. The simulated environments also provide a safe and cost-effective way to train drones. In traditional physical training methods, crashes or other accidents can be costly and time-consuming to repair, and may even pose a risk to human operators. In contrast, simulated crashes or other incidents in a virtual environment pose no risk to humans or property, allowing for more extensive and realistic training without the associated costs and risks. Finally, the simulated environments allow for more efficient and iterative training processes. Training algorithms and parameters can be quickly adjusted and refined, and the drone's performance can be evaluated and analyzed in real-time. This enables faster and more effective learning, leading to more skilled and capable drones in a shorter amount of time. We chose Unity 3D as the training platform for the drone, given its capability in creating a scaled simulation environment with analytical functions for reading and generating external data from the experiment [67].

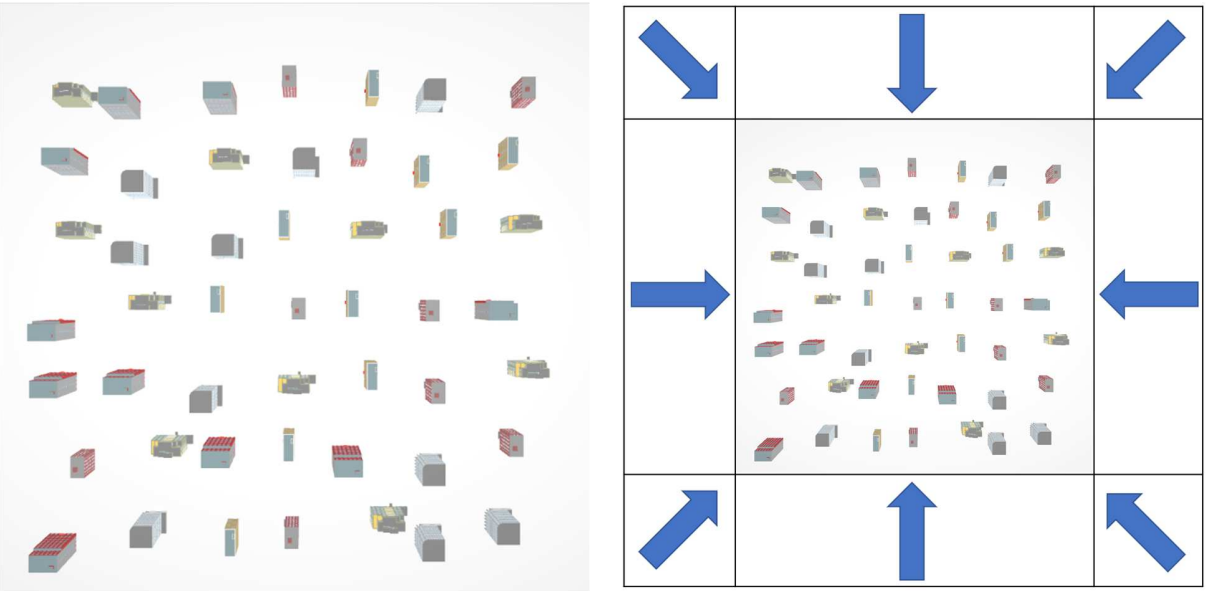
4.1. Simulated Environments

The simulated site represented a 3000 × 3000 feet district in an urban area. We randomly placed 49 tall buildings in the simulated environment, with the width and length ranging from 130 to 260 feet and height ranging from 200 to 400 feet. Besides, we did not set any physical boundary for the simulated site. Thus, the wind could pass through the site. For each epoch, the drone was randomly spawned at a location on one side of the site, while the target point was randomly placed on the other side. The drone must safely cross the whole site to reach the target. We

allowed 3D navigation for the drone. As such, the drone could adjust its altitude. However, in order to prevent the drone from always flying over the buildings, we set a height limitation of 160 feet for the drone. This ensured that the drone learned strategies to fly through the buildings and dynamic wind zones instead of flying over them. As a result, the total dimension of the main scenario is a $3000 \times 3000 \times 160$ feet cube.

As for the wind simulation, we set the initial wind speed at 11 mph. Using the aerodynamic representation framework as discussed earlier, the wind speed during the training was set to change from 3 to 45 mph, which covered the most common real-life situations. Another important parameter in the wind simulation is the initial wind direction. Given that we focused on the drone's navigation on the X-Y plane, we simulated eight initial wind directions shown in **Fig. 5**. For each epoch, the drone agent would experience one of the eight wind directions or windless conditions randomly, adding variability to the simulated environment. Depending on the flying direction of the drone and the initial wind direction, the wind should have exerted different effects on the drone. For example, if the wind direction is parallel with the drone's flight direction, the drone's speed would increase. A faster speed can be challenging for automatic navigation control, as there would be a short window for collision avoidance. Another challenging situation is the side wind, i.e., the wind direction and the drone's flight direction are perpendicular. In this case, it is nontrivial for the control algorithm to maintain the desired path or for stabilization controls. The goal of MONRL is to teach the drone to recognize the hidden patterns of the effect of the wind and spontaneously generate optimal solutions to offset its impact.

Regarding the simulation of GPS functionality in our experiment, we utilized Unity's local coordinate system to relay position information to the drone. The main objective of the GPS equipment in this experiment was to provide the local position information of both the drone and its target. It's important to note that our experimental design assumed the drone was equipped to access only local positional information. Consequently, there was no requirement for global positioning data in this context. This approach allowed us to focus on the drone's ability to navigate and interact effectively within its immediate environment, based on the local information provided.



(a) Scenario design (b) Wind direction design

Fig. 5 Example of simulated environments and drone

4.2. Drone Simulation

To closely emulate the realistic behaviors of a drone in our simulation, it is important to ensure that all parameters utilized in the drone simulation accurately reflect those in the real world. In this simulation, our selection is the DJI Mavic 2 Pro drone—an esteemed creation by DJI, a renowned pioneer in unmanned aerial vehicles (drones) and camera stabilization systems. This drone boasts sophisticated GPS technology that guarantees meticulous positioning and navigation. This pivotal attribute augments flight stability while enabling intelligent flight modes that simplify the capture of intricate shots. Drawing from its official specifications, the drone attains a top speed of 45 mph and achieves a maximum flight altitude of approximately 1640 feet. However, given the considerations pertaining to urban environments and air traffic control, prudent restrictions are placed on the drone's height and velocity. Considering the foregoing, the drone's specifications for the simulation have been set as follows:

- Weight: 2 lbs
- Maximum speed: 30 mph
- Maximum altitude: 160 feet

4.3. Curriculum Learning

We utilized curriculum learning to help the drone learn complex missions, such as detouring the strong wind zone. Curriculum learning is a popular and effective technique in machine learning that aims to train models by gradually increasing the complexity of the tasks they need to perform [10]. The basic idea behind curriculum learning is to present the model with a sequence of tasks that start off relatively easy and become progressively more difficult over time. By following this curriculum, the model can learn more efficiently and effectively while avoiding getting stuck in sub-optimal solutions. There are various metrics available to decide the appropriate number of epochs required for training each lesson, which may include loss, entropy, or mean final reward. This approach is inspired by the way humans learn, starting with simple concepts and gradually building upon them to tackle more complex ideas. In our work, there is a multi-objective task for the drone, which includes navigation, dodging the obstacles, controlling the drone in the wind zone, and detouring when there is a strong wind zone. It is hard for the drone to learn all these things in a single training environment. Therefore, we applied the curriculum learning strategy in our work and divided the whole training process into four phases as follows.

In the first phase, the primary purpose was to train the drone to learn to navigate and dodge all obstacles. For the environment setting, we only placed building obstacles on the scenario with no wind simulation. For every epoch, the positions of the drone and target were randomly generated. Besides, all the obstacles were also randomly distributed in the scenario. The purpose of this setting was that we did not want the drone to remember the obstacles' location. During the training, if a collision was encountered or the drone flew out of the scenario, a negative reward was given to the drone, stopping the epoch. In addition, if the step reached the maximum number, the drone would be assigned a timeout penalty. A limit of 15,000 steps was used for every epoch. Once the success rate reached a high number, we moved to the next phase.

In the second phase, since the drone was able to navigate and dodge the obstacles in a no-wind environment after phase 1, a wind zone was added to the training environment. The main goal of this phase was to train the agent to learn to control the drone in a wind zone environment. Except for the wind simulation, all other settings were the same as phase 1. For every epoch, the

drone agent would choose one direction as the wind direction, and then this direction would be used to calculate the simulation wind zone around each building.

After the first two phases of training, the drone was able to fly safely in an environment with a random wind zone. However, in some cases, the drone could still be caught in an infinite loop and be stuck at a specific position due to the strong wind zone's existence and the complexity of the layout. In order to allow the drone to learn to detour, a more dynamic training environment was required. In the third phase, we changed the wind direction choices to make sure all the initial wind direction was opposite to the drone's flight direction. In addition, we developed a new training strategy for the training in this phase. Unlike the previous approach, where an epoch stopped if the step reached the maximum in previous phases, we retained all the environmental factors and the drone status as in the last epoch. This strategy, combined with a specific penalty function r_{stuck}, aimed to enable the drone to overcome its inclination to local traps and find a solution for an infinite loop situation.

The final phase combined all the previous training features to improve the model's performance. One of the advantages of curriculum learning is that the system does not discard previously learned instances. Instead, each lesson or training criterion generates a unique set of weights that builds upon the previous ones during the training process. However, the sub-training criterion can add too much weight to the network. In that case, the whole model will be overfitted for a specific task and unsuitable for the multi-objective model. During this phase, the model would be retrained to seek better performance in the comprehensive environment. In addition, a variable learning rate was used to overcome issues with local optima and saddle points, which helped escape from the trap.

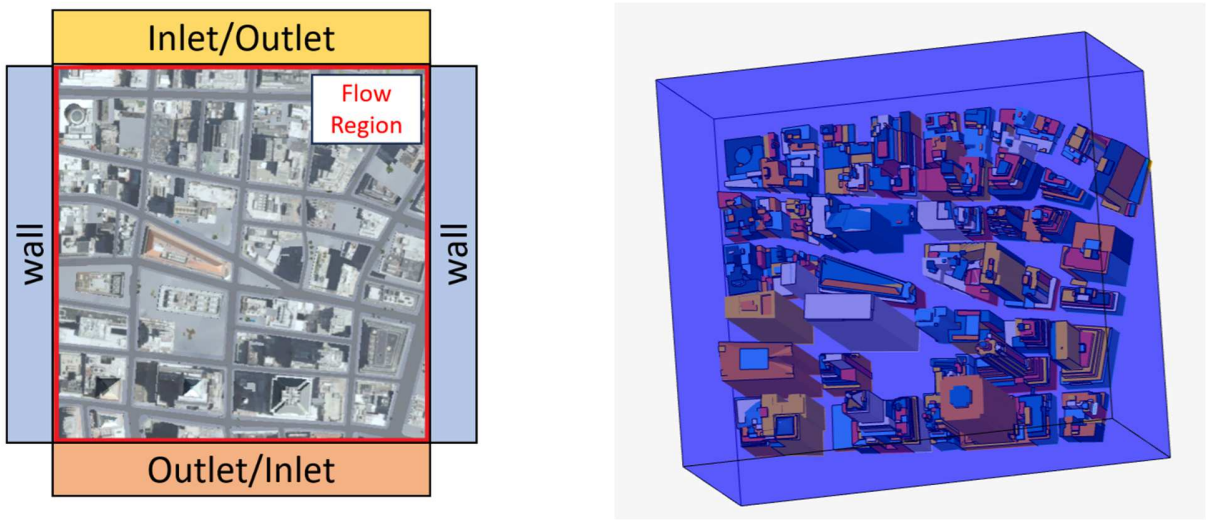
In the reinforcement learning segment of our research, we utilized the ML-Agents toolkit as our platform. ML-Agents, which stands for Machine Learning Agents, is a toolkit provided by Unity. Its primary purpose is to transform games and simulations into conducive environments for training machine learning algorithms. This bridges the gap between Unity's game development environment and the world of machine learning, thereby paving the way for the development of smarter and more interactive applications. **Table. 1** presents the settings we used for reinforcement learning within ML-Agents.

Parameter	**Value**
Max Steps	1,000,000,000
Time Horizon	1,000
Summary Frequency	100,000
Keep Checkpoints	10
Hyperparameter	
Batch Size	2,048
Buffer Size	20,480
Learning Rate	0.0003
Beta	0.01
Epsilon	0.2
Lambda	0.95
Num Epoch	3
Learning Rate Sch	Linear
Beta Schedule	Constant
Epsilon Schedule	Linear
Network Setting	
Normalize	True
Hidden Units	256
Number of Layers	2
Visualization Type	Simple
Memory Size	256
Sequence Length	64
Reward Signal	
Extrinsic Gamma	0.99
Extrinsic Strength	1.0

Table. 1 Details of ML-Agents setting

4.4. Computational Fluid Dynamics Setting

Our study generated the Computational Fluid Dynamics (CFD) data to evaluate our algorithm's performance rigorously. We set up the CFD simulations using the OpenFOAM software package to provide a brief overview. We defined the simulation domain, boundary conditions, and relevant physical parameters to the model, as shown in **Fig. 6.**



(a) (b)

Fig. 6 Details of CFD setting

Aspect/Parameter	**Details/Choice**
Material in Flow	Air
Air Density	1.196 kg/m^3
Wind Situations	Four different situations with two wind directions
Wind Speed (Inlet)	5 m/s and 10 m/s
Air Pressure (Outlet)	0.015 kPa and 0.0625 kPa
3D Domain for CFD	As shown in **Fig. 6(b)**
OpenFOAM Solver	simpleFoam
Meshing Generation	BlockMesh, snappyHexMesh
Grid Convergence Study	Yes

Table. 2 Details of OpenFOAM setting

Fig. 6(a) shows the domain of the fluid and the boundary conditions of the CFD setting environment. The material in the flow region is air, whose density is assigned to 1.196 kg/m3. In

782 the validation case, we assume four different wind situations with two different wind directions.
783 For each wind direction, we assign the wind speed at the inlet as 5 m/s and 10 m/s, and the air
784 pressure at the outlet is 0.015 kPa and 0.0625 kPa, respectively. **Fig. 6(b)** shows the 3D domain
785 for the CFD calculation. **Table. 2** lists all the settings in OpenFOAM.

786 For the meshing part in OpenFOAM, we began with the blockMesh utility to generate a
787 structured base hexahedral mesh for our computational domain. This provided a coarse mesh that
788 roughly captured the geometrical boundaries of our domain. The snappyHexMesh utility was
789 then used to refine the mesh and conform it to our complex geometry. After mesh generation, we
790 checked the quality of the mesh using Open Foam's checkMesh utility. This ensured that the
791 mesh met the quality criteria, such as non-orthogonality, skewness, and aspect ratio. Based on
792 the mesh quality metrics, iterative refinements were made until the mesh met the desired quality
793 standards for our simulations. **Table. 3** shows the mesh quality metrics of our model.

Metric	Min	Average	Max
Skewness	0	0.090	2.836
nonOrthogonality	0	10.918	70.305
Aspect Ratio	1	1.229	3.592
Volume Ratio	1	1.346	17.222

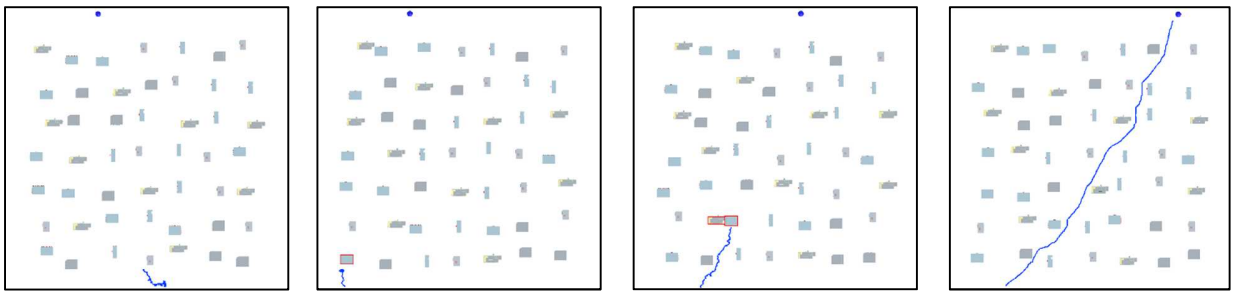
794 **Table. 3** Mesh quality metrics

795 ## 5. RESULTS

796 ### 5.1. Simulated Cases

797 We performed a series of tests to examine the performance of the proposed MONRL method in
798 piloting a drone to find the final target while avoiding and dodging against dynamic wind zones.
799 **Fig. 7** shows the process of the drone agent learning how to navigate and avoid collision with the
800 obstacles in the training environment in phase 1. The blue sphere in **Fig.7** refers to the target, the
801 red rectangle represents the bounding box result, and the buildings are represented as the cube.
802 To clearly display the drone's route, we use a blue line to describe the drone's trajectory in space**.**
803 **Fig. 7(a)** and **Fig. 7(b)** show the behavior of the drone at the earlier phase of the training. Since
804 the agent did not know its task and the meaning of the inputs, it randomly explored the
805 environment and often collided with the obstacles or the geometries. **Fig. 7(c)** shows that the

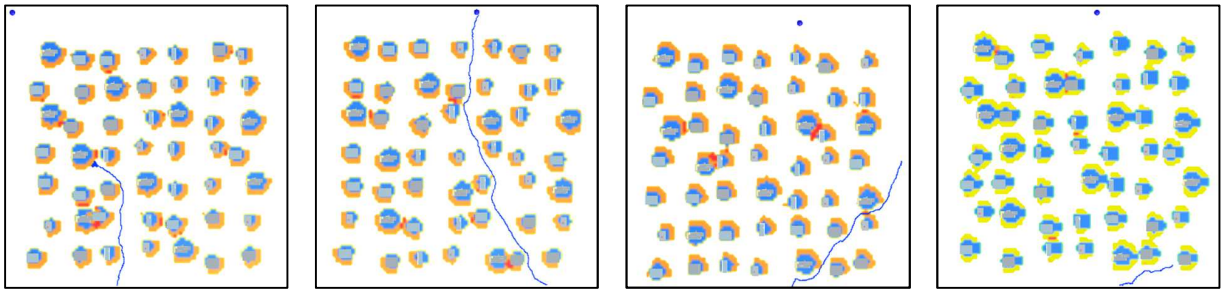
drone already learned its task and started moving toward the target based on previous rewards. Besides, it could control its flight in the training scene's scope and does not collide with the geometries. However, the agent still hit buildings. The reason is that the agent had not understood the meaning of the input, which represented the locations and dimensions of the building obstacles. **Fig. 7(d)** shows the drone could safely arrive at the target while avoiding all the obstacles in its way, and its success rate was 98% after 100,000 epochs. Furthermore, the drone learned how to choose the shortest path based on the local information from the start point to the target.



(a) (b) (c) (d)

Fig. 7 Examples of training results in phase 1

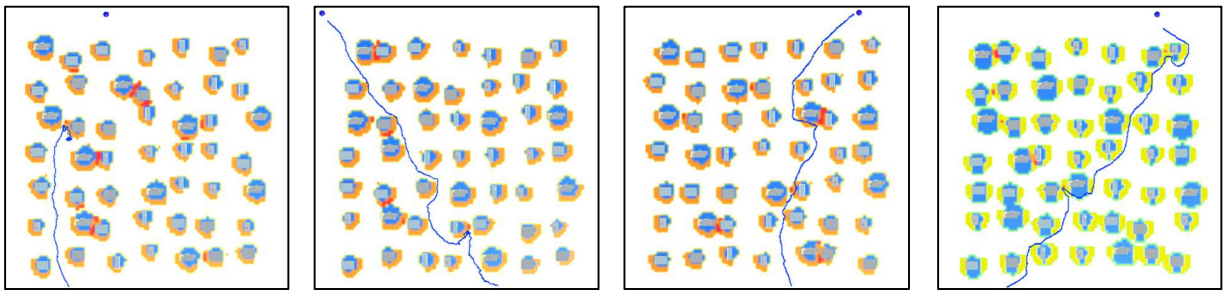
Fig. 8 shows the process of the drone agent learning how to adapt to the wind force and change its decision network in phase 2. As we mentioned, we added simulation of wind zones into the training scenario in this phase, which may have significantly affected the drone's flight route. To visualize the strength and the kind of wind, we created a heat map on the ground of the scene. The blank background means that there is no initial wind. Depending on the strength of the wind, the wind zones around the buildings were assigned three different colors. The red zone referred to the strongest wind zones, whose speed was larger than the drone's max speed (30 mph). The orange zone showed the magnitude of the wind that was larger than 50% of the drone's max speed but less than the maximum. Then the weaker wind zones were shown in blue.



(a) (b) (c) (d)

Fig. 8 Examples of training results in phase 2

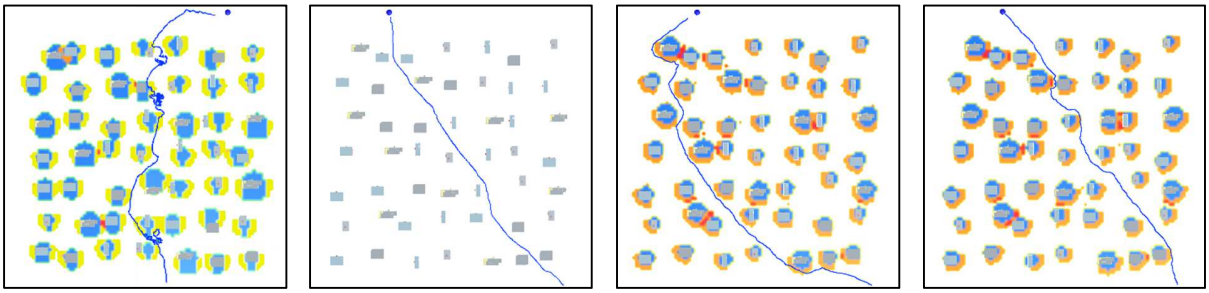
Fig. 8(a) shows an example of the poor performance of the drone in the earlier phase of the training scenario with a wind zone. Unlike the previous high success rate in a windless environment, the drone failed in most cases during the initial part of the training because it still generated the same strategies as the wind force did not exist. Then the wind forced the drone to deviate from its original route until it collided with the buildings. **Fig. 8(b)** shows that the drone starts recognizing the existence of the wind and changing its action decision. **Fig. 8(c)** shows that the drone could safely reach the target while dodging all the obstacles and crossing the wind zones with a success rate of 88% after 40,000 epochs. However, about 10% of the total cases were timeout cases in training phase 2. **Fig. 8(d)** shows the behavior of drones in timeout cases. As we can see, the drone was trapped at the boundary of the red wind zone, which indicated that the magnitude of the wind in this area was larger than the drone's maximum speed. In other words, if the drone aimed to enter this area and the wind direction was opposite to the drone's moving direction, the drone would be pushed back. And once the drone returned to its previous position, it would still decide to go forward because all the environmental information was the same, making an infinite loop of local trap.



(a) (b) (c) (d)

Fig. 9 Examples of training results in phase 3

Fig. 9 shows the procedure of the drone agent learning how to detour and avoid the strong wind zone, which was red in the figure. In training phase 3, to increase the probability of timeout cases, we retrained the drone in the same environment once there was a timeout case. Attributed to the LSTM network, the drone could memorize its previous actions, making it possible to bypass the strong wind area. **Fig. 9(a)** shows that the drone started to find another route to leave away from the strong wind zone but failed in the early stage of training. A high penalty for staying around the strong wind zone forced the drone to fly towards different directions to detour, but the drone would return to the area when it left the range of the penalty area. **Fig. 9(b)** shows the drone could find the alternate route after spending sufficient time around the strong wind zone. This means the drone successfully recognized the existence of a strong wind zone (even without any aerodynamic force sensor) and could change its decision based on previous decisions. **Fig. 9(c)** and **Fig. 9(d)** show that after enough epochs of training, the drone could turn quickly when it entered into a strong wind zone and chose to fly away to bypass it. The success rate in this training environment was 98% after 20,000 epochs.

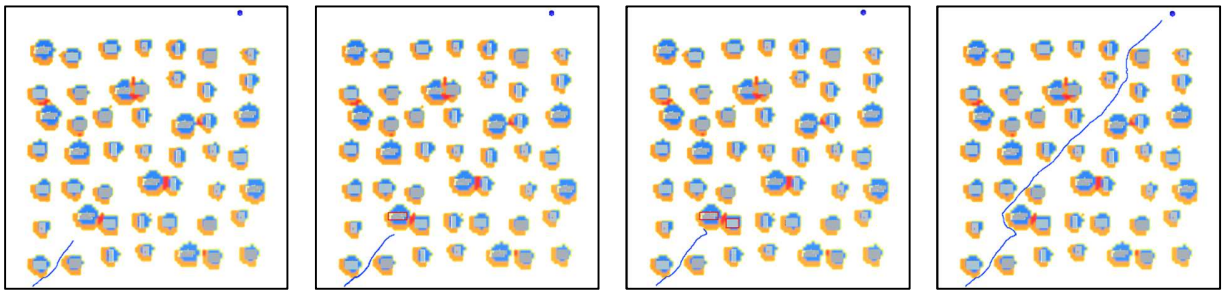


(a) (b) (c) (d)

Fig. 10 Examples of training results in phase 4

Fig. 10 shows the procedure of training the drone agent in the comprehensive environment in training phase 4. **Fig. 10(a)** shows the poor behaviors of the drone in the early phase with all possible wind directions at the beginning of the training. The model seemed overfitted, and the agent had forgotten how to control the drone in a downwind environment. **Fig. 10(b)-(d)** show that the drone agent could successfully bypass the strong wind zone and safely arrive at the target on the same map with varying wind directions after 20,000 epochs. In **Fig. 10(b)**, since there was no wind zone in the environment, the drone chose the shortest path to the

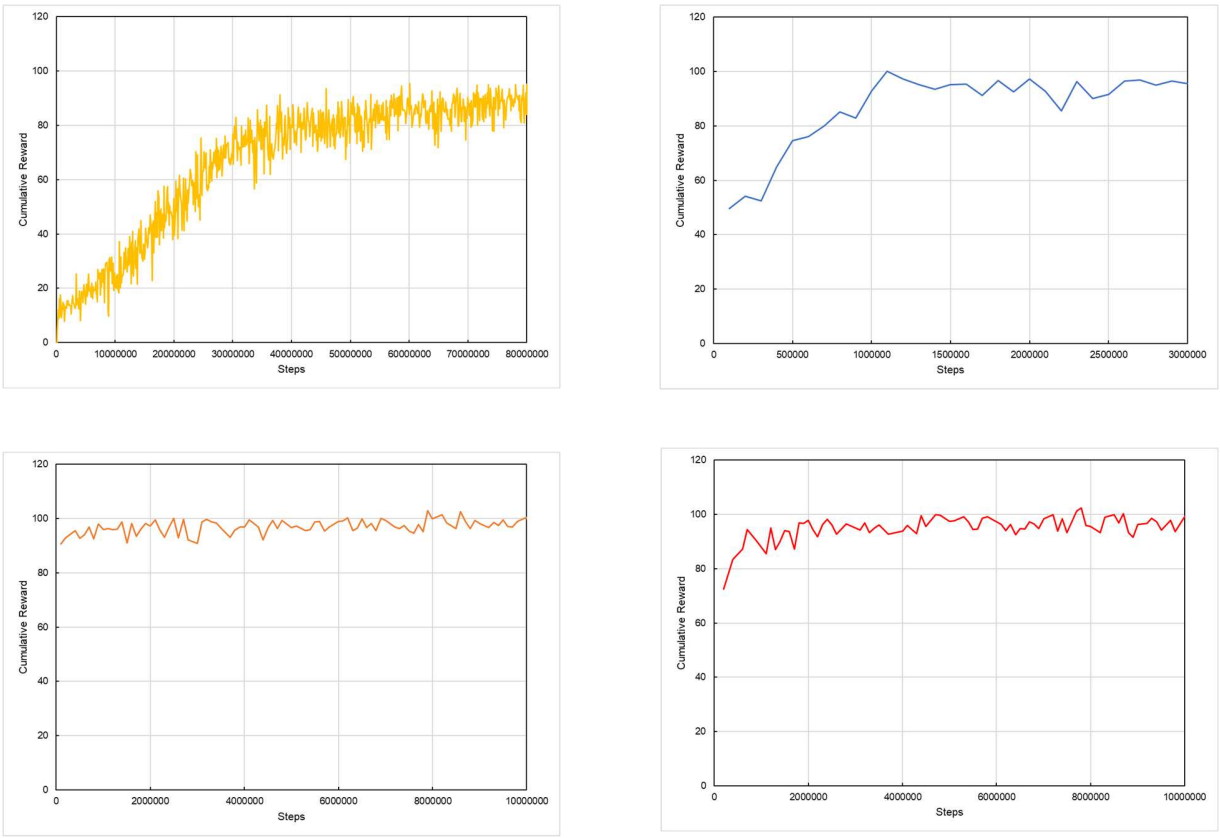
target, which proved that the agent may have learned the optimal global route based on the local information. In **Fig. 10(c)**, the map's initial wind direction was from the top right to the bottom left. In this case, the drone chose to fly in the partial wind direction to save energy and increase the speed. At the same time, we found that the drone always kept a certain distance from the nearby buildings to reduce the wind impact. Similarly, in **Fig. 10(d)**, the drone's flight path was more to the right compared to the windless situation. After the training, the final success rate for the model was about 93% after 20,000 epochs.



(a) (b) (c) (d)

Fig. 11 Process of drone learning to detour strong winds based only on imagery data

Fig. 11 shows the whole process of a drone bypassing a strong wind zone without entering it. **Fig. 11(a)** shows that the drone was flying toward the obstacle, and its camera captured the building on the left of the strong wind zone. At this moment, the drone did not change its direction. **Fig. 11(b)** shows that the drone continued flying in this direction, and its camera captured both buildings next to the strong wind zone, shown in the red box. **Fig. 11(c)** shows that the drone immediately changed its flight direction, bypassed the left building, and continued moving toward the target. **Fig. 11(d)** shows the drone's whole flight route in this case. This drone's behavior was very dynamic because the drone could only obtain the wind information for the imagery data (such as the relative locations of two adjacent buildings) and its own positions. Since the drone was not equipped with any aerodynamic force sensors, there was no relative information about the aerodynamic forces in the inputs of the model. However, the drone agent could still decide to detour before entering the strong wind zone. This proves that the drone could establish some knowledge about the relationship between the wind and the building layout information, such as the distance between the two obstacles and their dimensions.



(a) (b)

(c) (d)

Fig. 12 Learning curve of all phases

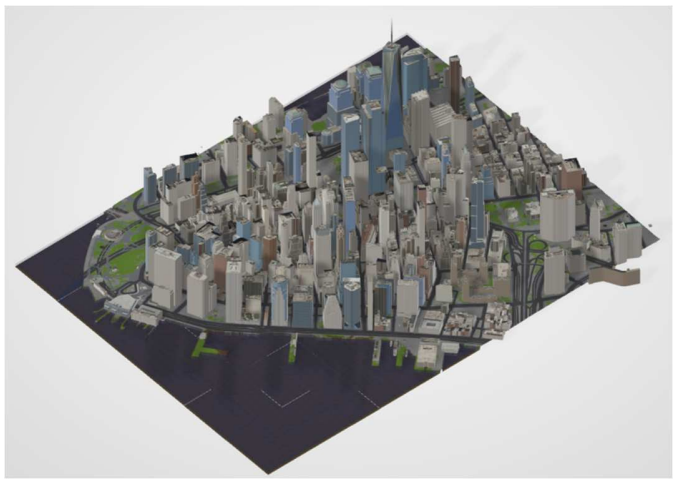
As shown in **Fig. 12**, we collect the data from every epoch and record the mean reward every 100,000 steps. For training phase 1, we can observe from **Fig. 12(a)** that the agents received greater rewards as the number of iterations increased. The reward curve raised rapidly at the first period of the training, then increased slower when the steps reached 40M, and eventually, it became stable when the steps reached 80M. **Fig. 12(b)** shows the mean rewards change in phase 2. At first, the rewards were very low. After 1M training, the rewards became stable, which meant the drone could fly safely in an environment with winds presenting. As shown in **Fig. 12(c)**, there was little improvement in the mean reward in phase 3 because the agent had already learned how to control the drone with winds presenting. The goal for the agent in phase 3 was to learn how to quickly detour in a more dynamic situation, which only reduced

902 the time penalty on the rewards. **Fig. 12(d)** shows the overfitted phenomenon during the training.
903 The mean rewards were lower the expected at the beginning of the training, but the agent quickly
904 reviewed the previous lessons and took 2M epochs to recover its high-performance strategies.

905 **5.2. Validation Case**

906 To further validate the efficacy of the proposed MONRL method in guiding a drone navigating
907 in a more realistic urban setup with precise aerodynamic data, we chose the Manhattan district in
908 New York City as the validation case, as shown in **Fig. 13**. Manhattan is known for its tall
909 buildings, particularly in its central business district. The island has a high concentration of
910 skyscrapers, including iconic structures such as the Empire State Building, the Chrysler Building,
911 and the One World Trade Center. These buildings not only define the city's skyline but also
912 impact its local environment by creating unique wind patterns, shading effects, and other
913 microclimatic conditions. The main wind direction for Manhattan is generally from the
914 northwest, as the city is located on the east coast of the United States and is influenced by
915 prevailing westerly winds. Therefore, we choose two opposite wind directions, northwest, and
916 southeast, as the main direction of the wind to test the algorithm. **Fig. 13 (c)** and **(d)** show the
917 first-person view from the drone in the virtual simulator.

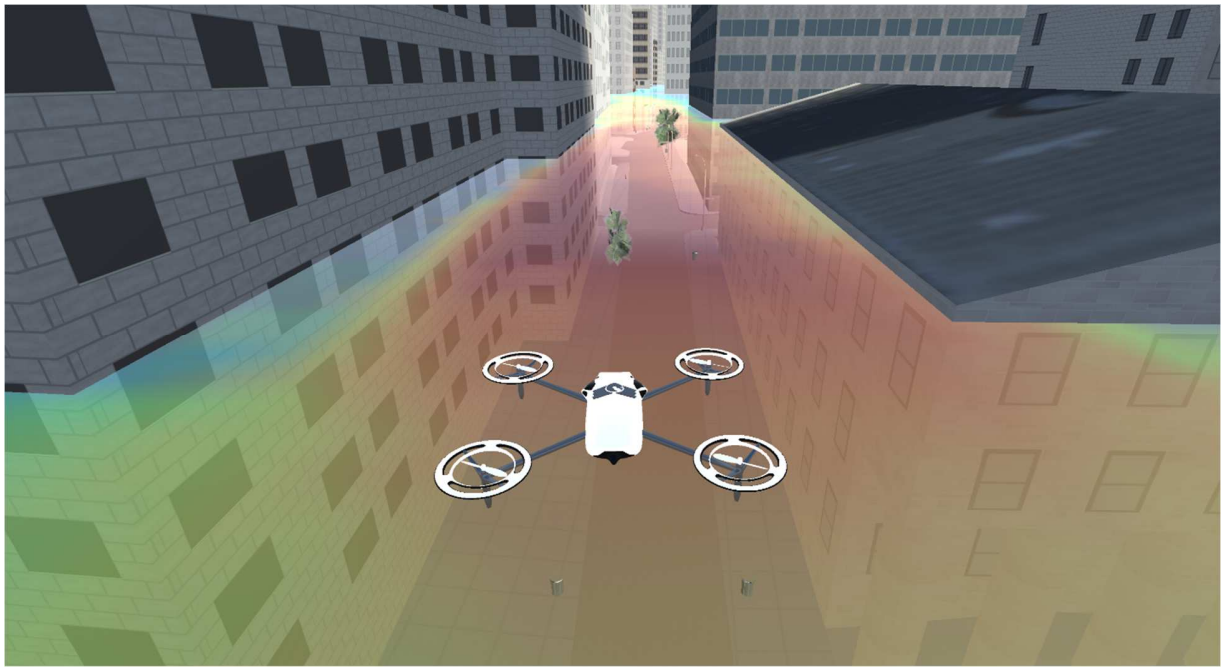
(a) Selected area in CFD case

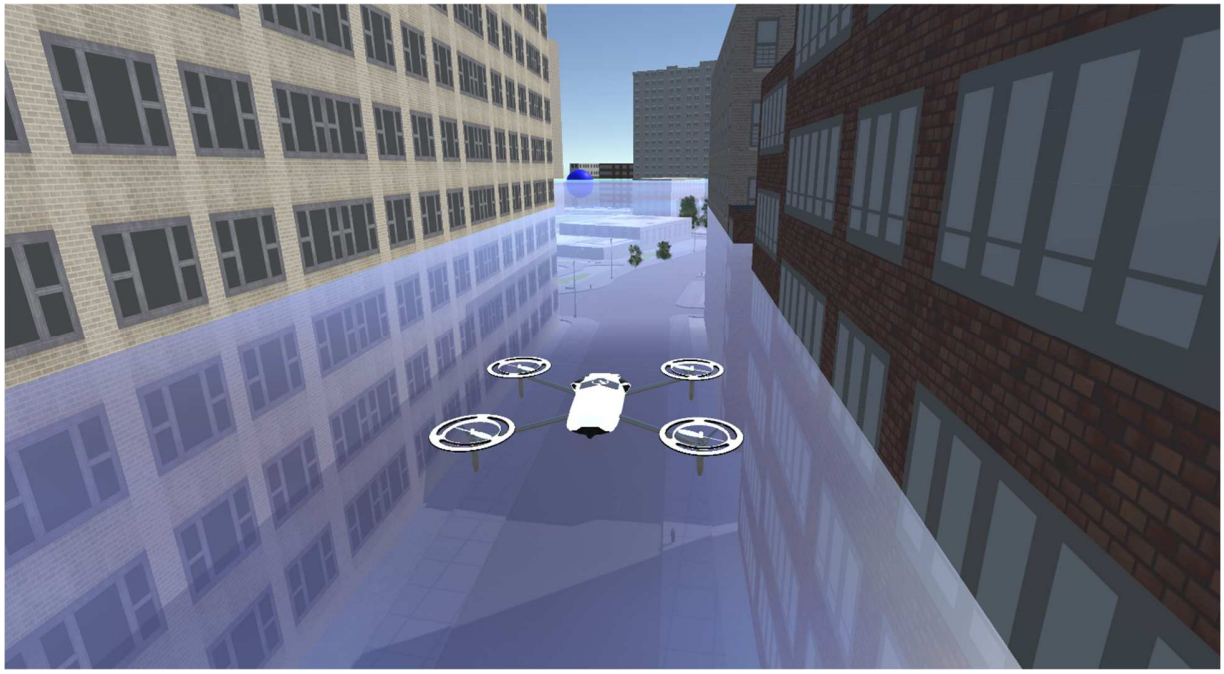


(b) Manhattan district 3D model

918

919





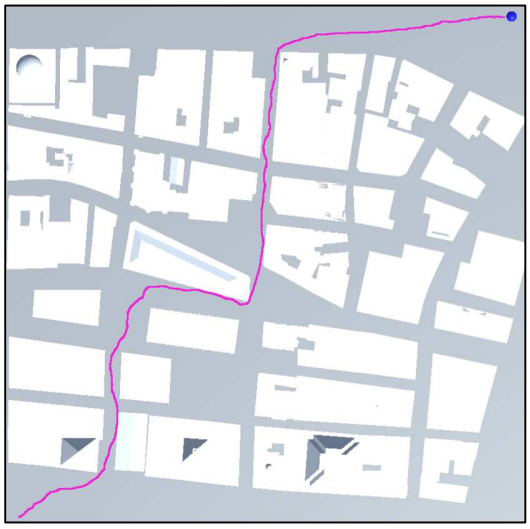
(c) Drone is flying along with airflow direction

(d) Drone is approaching target (blue ball)

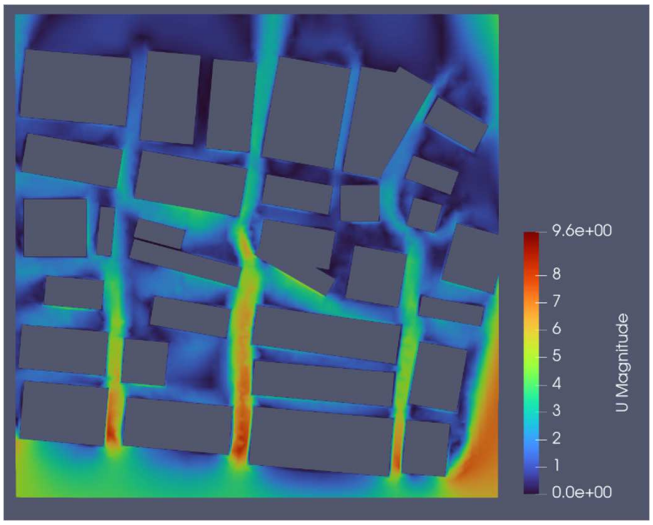
Fig. 13 Details of Manhattan district model

There are two main differences between the training environment and the validation test case. The first difference is the layout of the environment. In the training environment, the layout of the buildings is generated and distributed randomly by the Unity program. However, in the validation case, we use this Manhattan district in New York City, a real city model, to test its behavior. The second difference is that we use "Aerodynamic Representation Modeling," introduced in section 3.5, to simulate the wind map during the training. However, we use Open FOAM to generate the CFD results for the Manhattan model in the validation case, which has a higher authenticity of the wind flow data.

Fig. 14 illustrates the varying trade-off behaviors of the drone within the New York City model, where two conflicting objectives are at play. These objectives involve selecting the most efficient route to the target while circumventing areas of high wind intensity. The results effectively demonstrate the capability of our proposed algorithm to train the drone in adapting to wind dynamics and proficiently navigating through unfamiliar terrain. In **Fig. 14 (a)**, the drone's behavior is showcased under calm wind conditions. In this context, the drone intelligently opted for the shortest path between the starting point and the destination. **Fig. 14 (c)** and **(e)** present distinct route choices made by the drone in response to varying wind strengths. Specifically, when the wind direction shifted from southeast to northwest, favorably aiding the drone's flight, the choice to embark on a longer, direct journey emerged as preferred, as highlighted in **Fig. 14 (e)**. On the other hand, **Fig. 14 (g)** and **(i)** exhibit alternative route selections when the wind forces opposed the drone's movement. In these scenarios, the presence of potent wind zones, indicated by the red segments in **Fig. 14 (g)** and **(i)**, impeded the drone's ability to follow the shortest path observed in **Fig. 14 (a).** To mitigate the impact of adverse wind forces, the drone strategically elected routes that traversed through milder wind zones, relying solely on imagery data and previously memorized positions from earlier epochs. This deviation from conventional paths underscores the drone's calculated trade-off decision, opting to cover more distance in order to minimize the detrimental influence of the environmental variable—wind force. A demo can be found here: https://youtu.be/ycMFNs-YCRU.

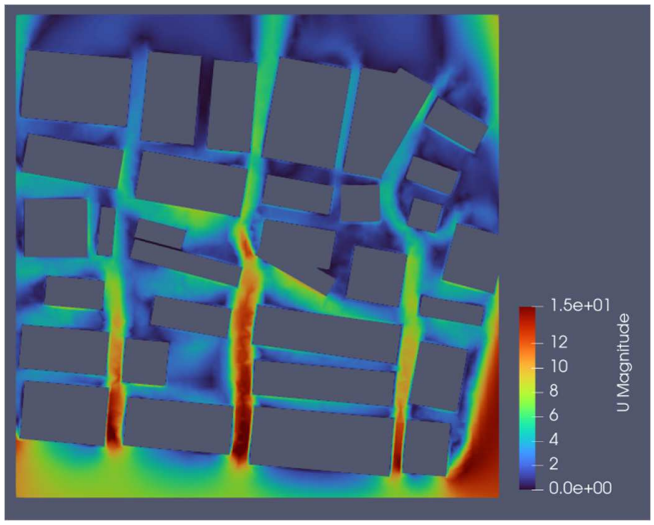


(a) Drone's route in windless environment



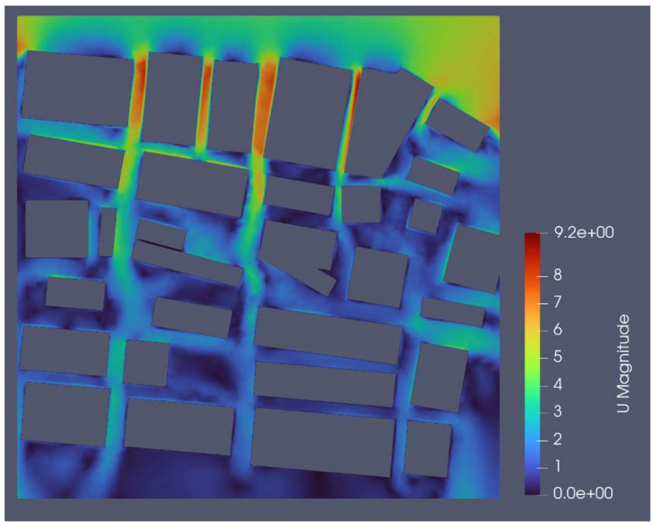
(b) CFD case 1 (wind from SE to NW, w_v = 5m/s)

(c) Drone's route in case 1



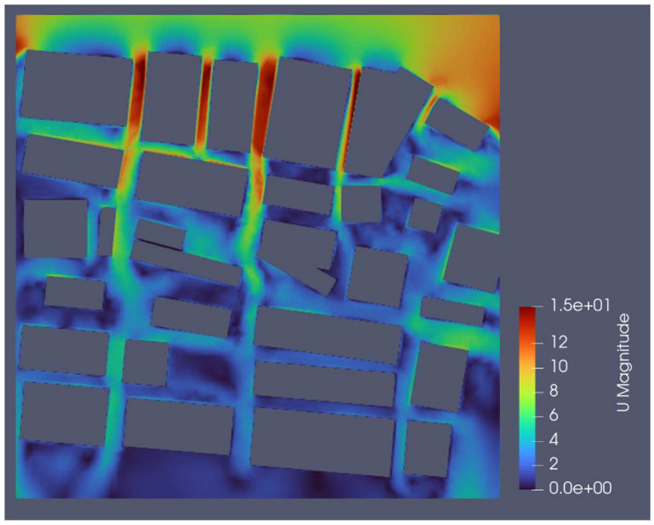
(d) CFD case 2 (wind from SE to NW, w_v = 10m/s)

(e) Drone's route in case 2

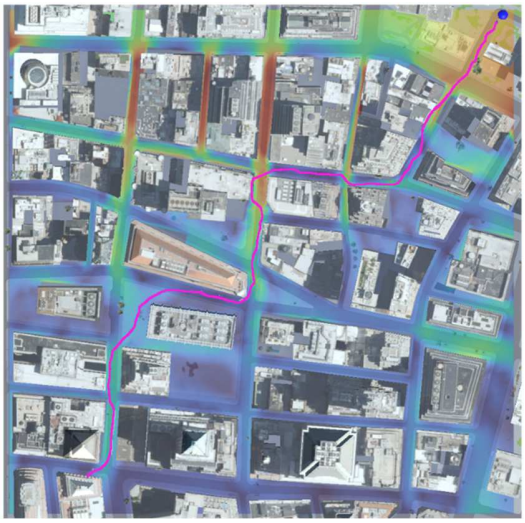


(f) CFD case 3 (wind from NW to SE, w_v = 5m/s)

(g) Drone's route in case 3



(h) CFD case 4 (wind from NW to SE, w_v = 10m/s)



(i) Drone's route in case 4

Fig. 14 Validation case study results

6. CONCLUSIONS

In this paper, we introduced a cutting-edge approach for drone navigation within urban environments, incorporating the influence of winds. This method relies on multi-objective deep reinforcement learning, LSTM, and a simulation environment. We highlight that conventional drone navigation and obstacle avoidance systems often neglect the impact of environmental factors, particularly the lateral wind surrounding tall buildings in urban areas. Acknowledging the potential simplicity of our empirical test, it's essential to emphasize our intent. While designed in a specific setting, our framework encapsulates some of the more intricate aspects of drone navigation in dynamic environments. By focusing on autonomous drones' ability to adapt to ever-changing aerodynamic conditions using minimalistic inputs, specifically from an onboard camera, we underscored the pivotal relationship between urban building layouts and wind zone distributions. Such an approach challenges the drone to maximize learning from sparse environmental cues, a testament to our methodology's robustness. Consequently, we devise a novel mathematical methodology to simulate wind distribution around buildings and generate a training environment using this method. Specifically, we employed an LSTM architecture combined with a PPO policy to train the drone to navigate and avoid building obstacles in a windless environment. Then, we introduced simulated wind zones into the environment, teaching the drone to execute detours. Finally, we validated the drone's navigation and obstacle avoidance capabilities within an environment featuring varying wind conditions, using an actual New York

City model integrated with CFD results. The findings demonstrate that our proposed system effectively adapts to wind effects in unknown environments, successfully reaching its destination without collisions. Moreover, the drone's flight path confirms the agent's capacity to plan an optimal global route based on local information, eliminating the need for a global map. During the training process, we can also ascertain that transferring a drone model trained with simulation wind methodology to an unknown environment with CFD results is feasible. The agent learns to control the drone under wind effects from the wind zone created by the wind simulation methodology.

Despite the promising results demonstrated by our proposed system, several limitations warrant further investigation. First, the current approach predominantly focuses on wind as the primary environmental factor affecting drone navigation within urban areas. Other factors, such as precipitation, temperature, signal coverage, air pressure, and the inherent decision to assign equal weights to all objectives in tailoring specific objectives, are not accounted for. This could significantly influence drone performance in diverse scenarios. Furthermore, our system's validation in a virtual Manhattan district, although rigorous, introduced challenges due to its heavy reliance on a camera and GPS. Complex environments characterized by densely built structures and varying topographies can sometimes obscure visual feedback, leading to the misidentification of narrow gaps between structures. Navigating such intricate settings and environments with dynamic obstacles, such as moving vehicles and pedestrians, remains a significant challenge. Another noteworthy limitation is the simulation's use of GPS equipment, mirroring a GPS function but lacking in replicating real-time GPS interactions, which are vital for an authentic and comprehensive simulation experience. Addressing sensor noise, especially in the context of environmental factors like electromagnetic interference and adverse weather conditions, remains critical. The current reinforcement learning model, while promising, may encounter limitations in more intricate environments or with diverse drone models. Emphasizing the computationally demanding nature of integrating reinforcement learning with Computational Fluid Dynamics (CFD) results, the system's scalability in various urban settings and drone capabilities await further exploration. Future research should focus on optimizing the wind simulation methodology to mirror urban environment complexities better. It's also essential to incorporate a broader range of environmental factors, assigning them variable weights as appropriate. Addressing dynamic elements within urban settings remains a key area of focus.

Furthermore, developing a more refined obstacle detection methodology is necessary to enhance the drone's navigation in complex terrains. While our research has demonstrated potential in urban drone navigation using deep reinforcement learning, it is important to acknowledge the limitations imposed by the use of simulated GPS data. Future studies may explore the integration of real GPS data or other advanced positioning technologies to enhance the applicability and accuracy of such navigational systems. Integrating Hardware-in-the-Loop (HIL) simulations could provide a more accurate reflection of and response to real-world GPS signals. Lastly, enhancing the scalability of the proposed system and streamlining the training process to reduce computational demands are critical steps for improving the practical applicability of this approach.

It is essential to acknowledge a limitation that has emerged through our empirical test cases. While these cases were meticulously designed to demonstrate the effectiveness of our proposed methodology in a controlled environment with limited sensor input, their simplified nature may limit the direct applicability to more complex, real-world scenarios. We chose these test cases intentionally, aiming to highlight the core capabilities of our methodology in a clear and controlled setting. This approach was particularly relevant for urban drone navigation, where complex and expensive sensor systems are often impractical. By demonstrating effective performance under these conditions, we aimed to underscore the robustness and practical utility of our methodology. However, we recognize that the simplified nature of these test cases might not capture the full spectrum of challenges and variables present in more intricate urban environments. As such, while these test cases serve as a valuable proof of concept and a foundational framework for our methodology, they do not fully represent the complexity of real-world applications. This limitation does not diminish the value of our findings but rather provides a direction for future research. It underscores the need for further studies that apply our methodology in more diverse and complex scenarios. We believe that extending our research to these settings will not only validate but also enhance the applicability and scalability of our approach.

ACKNOWLEDGEMENTS

This material is supported by the Air Force Office of Scientific Research (AFOSR) under grant FA9550-22-1-0492. Any opinions, findings, conclusions, or recommendations expressed in this article are those of the authors and do not reflect the views of the AFOSR.

REFERENCES

[1] A. Albert, F.S. Leira, L. Imsland. UAV Path Planning using MILP with Experiments. Modeling, Identification and Control: A Norwegian Research Bulletin, 38 (1) (2017), pp. 21-32. https://doi.org/10.4173/mic.2017.1.3

[2] E. Aldao, L.M. González-de Santos, H. González-Jorge. LiDAR Based Detect and Avoid System for UAV Navigation in UAM Corridors. Drones, 6 (8) (2022), pp. 185. https://doi.org/10.3390/drones6080185

[3] O. Andersson, M. Wzorek, P. Rudol, P. Doherty. Model-predictive control with stochastic collision avoidance using bayesian policy optimization. 2016 IEEE International Conference on Robotics and Automation (ICRA), IEEE, (2016), pp. 4597-4604. https://doi.org/10.1109/ICRA.2016.7487661

[4] L. Apvrille, T. Tanzi, J.-L. Dugelay. Autonomous drones for assisting rescue services within the context of natural disasters. 2014 XXXIth URSI General Assembly and Scientific Symposium (URSI GASS), IEEE, (2014), pp. 1-4. https://doi.org/10.1109/URSIGASS.2014.6929384

[5] M.Y. Arafat, M.M. Alam, S. Moh. Vision-Based Navigation Techniques for Unmanned Aerial Vehicles: Review and Challenges. Drones, 7 (2) (2023), pp. 89. https://doi.org/10.3390/drones7020089

[6] 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), pp. 95441-95450. https://doi.org/10.1109/ACCESS.2022.3204876

[7] A.T. Azar, A. Koubaa, N. Ali Mohamed, H.A. Ibrahim, Z.F. Ibrahim, M. Kazim, A. Ammar, B. Benjdira, A.M. Khamis, I.A. Hameed. Drone deep reinforcement learning: A review. Electronics, 10 (9) (2021), pp. 999. https://doi.org/10.3390/electronics10090999

[8] T. Baca, D. Hert, G. Loianno, M. Saska, V. Kumar. Model predictive trajectory tracking and collision avoidance for reliable outdoor deployment of unmanned aerial vehicles. 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, (2018), pp. 6753-6760. https://doi.org/10.1109/IROS.2018.8594266

[9] D. Baskar, A. Gorodetsky. A Simulated Wind-field Dataset for Testing Energy Efficient Path-Planning Algorithms for UAVs in Urban Environment. AIAA AVIATION 2020 FORUM, (2020), p. 2920. https://doi.org/10.2514/6.2020-2920

[10] Y. Bengio, J. Louradour, R. Collobert, J. Weston. Curriculum learning. Proceedings of the 26th annual international conference on machine learning, (2009), pp. 41-48. https://doi.org/10.1145/1553374.1553380

[11] L. Biao, J. Cunyan, W. Lu, C. Weihua, L. Jing. A parametric study of the effect of building layout on wind flow over an urban area. Building and Environment, 160 (2019), pp. 106160. https://doi.org/10.1016/j.buildenv.2019.106160

[12] B. Blocken, T. Stathopoulos, J. Carmeliet. Wind environmental conditions in passages between two long narrow perpendicular buildings. Journal of Aerospace Engineering, 21 (4) (2008), pp. 280-287. https://doi.org/10.1061/(ASCE)0893-1321(2008)21:4(280)

[13] M. Bolognini, L. Fagiano. Lidar-based navigation of tethered drone formations in an unknown environment. IFAC-PapersOnLine, 53 (2) (2020), pp. 9426-9431. https://doi.org/10.1016/j.ifacol.2020.12.2413

[14] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, J.J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. IEEE Transactions on robotics, 32 (6) (2016), pp. 1309-1332. https://doi.org/10.1109/TRO.2016.2624754

[15] A. Capolupo, S. Pindozzi, C. Okello, N. Fiorentino, L. Boccia. Photogrammetry for environmental monitoring: The use of drones and hydrological models for detection of soil

contaminated by copper. Science of the Total Environment, 514 (2015), pp. 298-306. https://doi.org/10.1016/j.scitotenv.2015.01.109

[16] P. Chakravarty, K. Kelchtermans, T. Roussel, S. Wellens, T. Tuytelaars, L. Van Eycken. CNN-based single image obstacle avoidance on a quadrotor. 2017 IEEE international conference on robotics and automation (ICRA), IEEE, (2017), pp. 6369-6374. https://doi.org/10.1109/ICRA.2017.7989752

[17] A. Devo, G. Mezzetti, G. Costante, M.L. Fravolini, P. Valigi. Towards generalization in target-driven visual navigation by using deep reinforcement learning. IEEE Transactions on Robotics, 36 (5) (2020), pp. 1546-1561. https://doi.org/10.1109/TRO.2020.2994002

[18] A. Devos, E. Ebeid, P. Manoonpong. Development of autonomous drones for adaptive obstacle avoidance in real world environments. 2018 21st Euromicro conference on digital system design (DSD), IEEE, (2018), pp. 707-710. https://doi.org/10.1109/DSD.2018.00009

[19] N. El-Sheimy, Y. Li. Indoor navigation: State of the art and future trends. Satellite Navigation, 2 (1) (2021), pp. 1-23. https://doi.org/10.1186/s43020-021-00041-3

[20] T. Elmokadem, A.V. Savkin. A hybrid approach for autonomous collision-free UAV navigation in 3D partially unknown dynamic environments. Drones, 5 (3) (2021), pp. 57. https://doi.org/10.3390/drones5030057

[21] A. Entrop, A. Vasenev. Infrared drones in the construction industry: designing a protocol for building thermography procedures. Energy procedia, 132 (2017), pp. 63-68. https://doi.org/10.1016/j.egypro.2017.09.636

[22] M. Esposito, M. Crimaldi, V. Cirillo, F. Sarghini, A. Maggio. Drone and sensor technology for sustainable weed management: A review. Chemical and Biological Technologies in Agriculture, 8 (1) (2021), pp. 1-11. https://doi.org/10.1186/s40538-021-00217-8

[23] Y. Fan, S. Chu, W. Zhang, R. Song, Y. Li. Learn by observation: Imitation learning for drone patrolling from videos of a human navigator. 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, (2020), pp. 5209-5216. https://doi.org/10.48550/arXiv.2008.13193

[24] D. Floreano, R.J. Wood. Science, technology and the future of small autonomous drones. nature, 521 (7553) (2015), pp. 460-466. https://doi.org/10.1038/nature14542

[25] L.R. García Carrillo, A.E. Dzul López, R. Lozano, C. Pégard. Combining stereo vision and inertial navigation system for a quad-rotor UAV. Journal of intelligent & robotic systems, 65 (1-4) (2012), pp. 373-387. https://doi.org/10.1007/s10846-011-9571-7

[26] S. Hochreiter, J. Schmidhuber. Long short-term memory. Neural computation, 9 (8) (1997), pp. 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

[27] V.J. Hodge, R. Hawkins, R. Alexander. Deep reinforcement learning for drone navigation using sensor data. Neural Computing and Applications, 33 (2021), pp. 2015-2033. https://doi.org/10.1007/s00521-020-05097-x

[28] C. Huang, X. Zhou, X. Ran, J. Wang, H. Chen, W. Deng. Adaptive cylinder vector particle swarm optimization with differential evolution for UAV path planning. Engineering Applications of Artificial Intelligence, 121 (2023), pp. 105942. https://doi.org/10.1016/j.engappai.2023.105942

[29] M.L. Imrane, A. Melingui, J.J.B. Mvogo Ahanda, F. Biya Motto, R. Merzouki. Artificial potential field neuro-fuzzy controller for autonomous navigation of mobile robots. Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering, 235 (7) (2021), pp. 1179-1192. https://doi.org/10.1177/0959651820974831

[30] H. Jasak. OpenFOAM: open source CFD in research and industry. International Journal of Naval Architecture and Ocean Engineering, 1 (2) (2009), pp. 89-94. https://doi.org/10.2478/IJNAOE-2013-0011

[31] K. Javanroodi, M. Mahdavinejad, V.M. Nik. Impacts of urban morphology on reducing cooling load and increasing ventilation potential in hot-arid climate. Applied energy, 231 (2018), pp. 714-746. https://doi.org/10.1016/j.apenergy.2018.09.116

[32] S. Jung, S. Hwang, H. Shin, D.H. Shim. Perception, guidance, and navigation for indoor autonomous drone racing using deep learning. IEEE Robotics and Automation Letters, 3 (3) (2018), pp. 2539-2544. https://doi.org/10.1109/LRA.2018.2808368

[33] L.P. Kaelbling, M.L. Littman, A.W. Moore. Reinforcement learning: A survey. Journal of artificial intelligence research, 4 (1996), pp. 237-285. https://doi.org/10.48550/arXiv.cs/9605103

[34] M.T.R. Khan, M. Muhammad Saad, Y. Ru, J. Seo, D. Kim. Aspects of unmanned aerial vehicles path planning: Overview and applications. International Journal of Communication Systems, 34 (10) (2021), pp. e4827. https://doi.org/10.1002/dac.4827

[35] J. Kwak, Y. Sung. Autonomous UAV flight control for GPS-based navigation. IEEE Access, 6 (2018), pp. 37947-37955. https://doi.org/10.1109/ACCESS.2018.2854712

[36] S. Lee, D. Har, D. Kum. Drone-assisted disaster management: Finding victims via infrared camera and lidar sensor fusion. 2016 3rd Asia-Pacific World Congress on Computer Science and Engineering (APWC on CSE), IEEE, (2016), pp. 84-89. https://doi.org/10.1109/APWC-on-CSE.2016.025

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

[38] J. Linchant, J. Lisein, J. Semeki, P. Lejeune, C. Vermeulen. Are unmanned aircraft systems (UAS s) the future of wildlife monitoring? A review of accomplishments and challenges. Mammal Review, 45 (4) (2015), pp. 239-252. https://doi.org/10.1111/mam.12046

[39] Y. Liu, Q. Wang, H. Hu, Y. He. A novel real-time moving target tracking and path planning system for a quadrotor UAV in unknown unstructured outdoor scenes. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 49 (11) (2018), pp. 2362-2372. https://doi.org/10.1109/TSMC.2018.2808471

[40] Y. Lu, Z. Xue, G.-S. Xia, L. Zhang. A survey on vision-based UAV navigation. Geo-spatial information science, 21 (1) (2018), pp. 21-32. https://doi.org/10.1080/10095020.2017.1420509

[41] K. Minoda, F. Schilling, V. Wüest, D. Floreano, T. Yairi. Viode: A simulated dataset to address the challenges of visual-inertial odometry in dynamic environments. IEEE Robotics and Automation Letters, 6 (2) (2021), pp. 1343-1350. https://doi.org/10.1109/LRA.2021.3058073

[42] B. Mishra, D. Garg, P. Narang, V. Mishra. Drone-surveillance for search and rescue in natural disaster. Computer Communications, 156 (2020), pp. 1-10. https://doi.org/10.1016/j.comcom.2020.03.012

[43] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski. Human-level control through deep reinforcement learning. nature, 518 (7540) (2015), pp. 529-533. https://doi.org/10.1038/nature14236

[44] A. Mousavian, D. Anguelov, J. Flynn, J. Kosecka. 3d bounding box estimation using deep learning and geometry. Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, (2017), pp. 7074-7082. https://doi.org/10.48550/arXiv.1612.00496

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

[46] A. Nagabandi, I. Clavera, S. Liu, R.S. Fearing, P. Abbeel, S. Levine, C. Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. arXiv preprint arXiv:1803.11347 (2018). https://doi.org/10.48550/arXiv.1803.11347

[47] J.C. Nathanael, C.H.J. Wang, K.H. Low. Simulation of Wind Field in a Building Complex for Evaluation of the Wind Effect Along UAS Flight Path. AIAA AVIATION 2023 Forum, (2023), p. 4096. https://doi.org/10.2514/6.2023-4096

[48] A. Patrik, G. Utama, A.A.S. Gunawan, A. Chowanda, J.S. Suroso, R. Shofiyanti, W. Budiharto. GNSS-based navigation systems of autonomous drone for delivering items. Journal of Big Data, 6 (2019), pp. 1-14. https://doi.org/10.1186/s40537-019-0214-3

[49] H.X. Pham, H.M. La, D. Feil-Seifer, L.V. Nguyen. Autonomous uav navigation using reinforcement learning. arXiv preprint arXiv:1801.05086 (2018). https://doi.org/10.1109/SSRR.2018.8468611

[50] V. Puri, A. Nayyar, L. Raja. Agriculture drones: A modern breakthrough in precision agriculture. Journal of Statistics and Management Systems, 20 (4) (2017), pp. 507-518. https://doi.org/10.1080/09720510.2017.1395171

[51] S. Ragi, E.K. Chong. UAV path planning in a dynamic environment via partially observable Markov decision process. IEEE Transactions on Aerospace and Electronic Systems, 49 (4) (2013), pp. 2397-2412. https://doi.org/10.1109/TAES.2013.6621824

[52] A. Ramezani Dooraki, D.-J. Lee. A multi-objective reinforcement learning based controller for autonomous navigation in challenging environments. Machines, 10 (7) (2022), pp. 500. https://doi.org/10.3390/machines10070500

[53] A. Rejeb, K. Rejeb, S.J. Simske, H. Treiblmaier. Drones for supply chain management and logistics: a review and research agenda. International Journal of Logistics Research and Applications (2021), pp. 1-24. https://doi.org/10.1080/13675567.2021.1981273

[54] A. Restas. Drone applications for supporting disaster management. World Journal of Engineering and Technology, 3 (03) (2015), pp. 316. http://dx.doi.org/10.4236/wjet.2015.33C047

[55] M.F. Sani, G. Karimian. Automatic navigation and landing of an indoor AR. drone quadrotor using ArUco marker and inertial sensors. 2017 international conference on computer and drone applications (IConDA), IEEE, (2017), pp. 102-107. https://doi.org/10.1109/ICONDA.2017.8270408

[56] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017). https://doi.org/10.48550/arXiv.1707.06347

[57] A. Shahoud, D. Shashev, S. Shidlovskiy. Visual navigation and path tracking using street geometry information for image alignment and servoing. Drones, 6 (5) (2022), pp. 107. https://doi.org/10.3390/drones6050107

[58] A. Shantia, R. Timmers, Y. Chong, C. Kuiper, F. Bidoia, L. Schomaker, M. Wiering. Two-stage visual navigation by deep neural networks and multi-goal reinforcement learning. Robotics and Autonomous Systems, 138 (2021), pp. 103731. https://doi.org/10.1016/j.robot.2021.103731

[59] S.M. Shavarani, M.G. Nejad, F. Rismanchian, G. Izbirak. Application of hierarchical facility location problem for optimization of a drone delivery system: a case study of Amazon prime air in the city of San Francisco. The International Journal of Advanced Manufacturing Technology, 95 (2018), pp. 3141-3153. https://doi.org/10.1007/s00170-017-1363-1

[60] S.-Y. Shin, Y.-W. Kang, Y.-G. Kim. Reward-driven U-Net training for obstacle avoidance drone. Expert Systems with Applications, 143 (2020), pp. 113064. https://doi.org/10.1016/j.eswa.2019.113064

[61] A. Suleiman, Z. Zhang, L. Carlone, S. Karaman, V. Sze. Navion: A 2-mw fully integrated real-time visual-inertial odometry accelerator for autonomous navigation of nano drones. IEEE Journal of Solid-State Circuits, 54 (4) (2019), pp. 1106-1119. https://doi.org/10.1109/JSSC.2018.2886342

[62] Q. Sun, M. Li, T. Wang, C. Zhao. UAV path planning based on improved rapidly-exploring random tree. 2018 Chinese control and decision conference (CCDC), IEEE, (2018), pp. 6420-6424. https://doi.org/10.1109/CCDC.2018.8408258

[63] G. Tong, N. Jiang, L. Biyue, Z. Xi, W. Ya, D. Wenbo. UAV navigation in high dynamic environments: A deep reinforcement learning approach. Chinese Journal of Aeronautics, 34 (2) (2021), pp. 479-489. https://doi.org/10.1016/j.cja.2020.05.011

[64] F. Vanegas, K.J. Gaston, J. Roberts, F. Gonzalez. A framework for UAV navigation and exploration in GPS-denied environments. 2019 ieee aerospace conference, IEEE, (2019), pp. 1-6. https://doi.org/10.1109/AERO.2019.8741612

[65] D. Ventura, M. Bruno, G.J. Lasinio, A. Belluscio, G. Ardizzone. A low-cost drone based application for identifying and mapping of coastal fish nursery grounds. Estuarine, Coastal and Shelf Science, 171 (2016), pp. 85-98. https://doi.org/10.1016/j.ecss.2016.01.030

[66] L. von Stumberg, V. Usenko, J. Engel, J. Stückler, D. Cremers. From monocular SLAM to autonomous drone exploration. 2017 European Conference on Mobile Robots (ECMR), IEEE, (2017), pp. 1-8. https://doi.org/10.1109/ECMR.2017.8098709

[67] S. Wang, Z. Mao, C. Zeng, H. Gong, S. Li, B. Chen. A new method of virtual reality based on Unity3D. 2010 18th international conference on Geoinformatics, IEEE, (2010), pp. 1-5. https://doi.org/10.1109/GEOINFORMATICS.2010.5567608

[68] Z. Wang, Y. Wu, Q. Niu. Multi-sensor fusion in automated driving: A survey. Ieee Access, 8 (2019), pp. 2847-2868. https://doi.org/10.1109/ACCESS.2019.2962554

[69] N. Wen, X. Su, P. Ma, L. Zhao, Y. Zhang. Online UAV path planning in uncertain and hostile environments. International journal of machine learning and cybernetics, 8 (2017), pp. 469-487. https://doi.org/10.1007/s13042-015-0339-4

[70] Z. Xue, T. Gonsalves. Vision based drone obstacle avoidance by deep reinforcement learning. AI, 2 (3) (2021), pp. 366-380. https://doi.org/10.3390/ai2030023

[71] M. Zhang, M. Zhang, Y. Chen, M. Li. IMU data processing for inertial aided navigation: A recurrent neural network based approach. 2021 IEEE International Conference on Robotics and Automation (ICRA), IEEE, (2021), pp. 3992-3998. https://doi.org/10.1109/ICRA48506.2021.9561172