

# Deep Reinforcement Learning for Autonomous Map-Less Navigation of a Flying Robot

QUALID DOUKHI AND DEOK JIN LEE

Department of Mechanical Design Engineering, Jeonbuk National University, Jeonju 54896, South Korea

Corresponding author: Deok Jin Lee (deokjlee@jbnu.ac.kr)

This work was supported by the Unmanned Vehicles Core Technology Research and Development Program through the National Research Foundation of Korea (NRF), Unmanned Vehicle Advanced Research Center (UVARC), funded by the Ministry of Science Information and Communications Technologies (ICT), the Republic of Korea, under Grant 2020M3C1C1A01082375; in part by the DNA+ Drone Technology Development Program through NRF funded by the Ministry of Science ICT under Grant NRF-2020M3C1C2A01080819; and in part by the research fund for newly appointed professors of Jeonbuk National University, in 2021.

**ABSTRACT** Flying robots are expected to be used in many tasks, such as aerial delivery, inspection inside dangerous areas, and rescue. However, their deployment in unstructured and highly dynamic environments has been limited. This paper proposes a novel approach for enabling a micro-aerial vehicle (MAV) system equipped with a laser rangefinder and depth sensor to autonomously navigate and explore an unknown indoor or outdoor environment. We built a modular deep-Q-network architecture to fuse information from multiple sensors mounted onboard a vehicle. The developed algorithm can perform collision-free flights in the real world, while being trained entirely on a 3D simulator. The proposed method does not require prior expert demonstrations, 3D mapping, or path planning. It transforms fused sensory data into a velocity control input for a robot through an end-to-end convolutional neural network (CNN). The obtained policy was compared to a simulation using the conventional potential-field method. Our approach achieves zero-shot transfer from simulation to real-world environments that were never experienced during training by simulating realistic sensor data. Several intensive experiments were conducted to demonstrate the effectiveness of our system for safely flying in dynamic outdoor and indoor environments. The supplementary videos for the actual flight tests can be accessed at <http://bit.ly/2SEw8dQ>.


**INDEX TERMS** Autonomous navigation, sensor-fusion, collision-free, deep Q-network, zero-shot transfer, micro aerial vehicle.

## I. INTRODUCTION

Micro aerial vehicles (MAVs) are widely used in various applications in both the military and civilian domains. Owing to their agility and maneuverability, multirotors can hover and move freely in 3D space, making them suitable for many applications, such as surveillance and rescue, aerial photography, and precision agriculture [1].

Autonomous navigation and collision avoidance are fundamental requirements for robotic aerial systems that operate in unstructured and unknown open-world environments. Designing an autonomous navigation system with the ability to avoid obstacles in MAVs is a long-established research problem [2]. It is difficult to autonomously navigate an under-actuated aerial robot that performs specific missions during an unstructured and unknown environment without

colliding with obstacles. Classical approaches are primarily based on 3D mapping, relative state estimation [3], [4], trajectory optimization, and path planning [5], [6]. Nevertheless, conventional methods have significant limitations and are prone to failures, particularly in unstructured, dynamic environments where reliable state feedback is unavailable for aerial robots. In addition, building an accurate 3D map of an environment requires considerable computational power. To address these problems, researchers have proposed Deep Learning (DL) based solutions that can be realized effectively and efficiently using current computer systems and graphical processing units (GPUs) [7]. The success of deep learning in solving problems in artificial intelligence [8]–[10] motivates researchers in control and robotics to apply the recent algorithms to common aerial robotics problems such as flight control [11] and obstacle avoidance. End-to-end convolutional neural network (CNN) approaches have been proposed to directly generate control inputs from

The associate editor coordinating the review of this manuscript and approving it for publication was Ehsan Asadi .

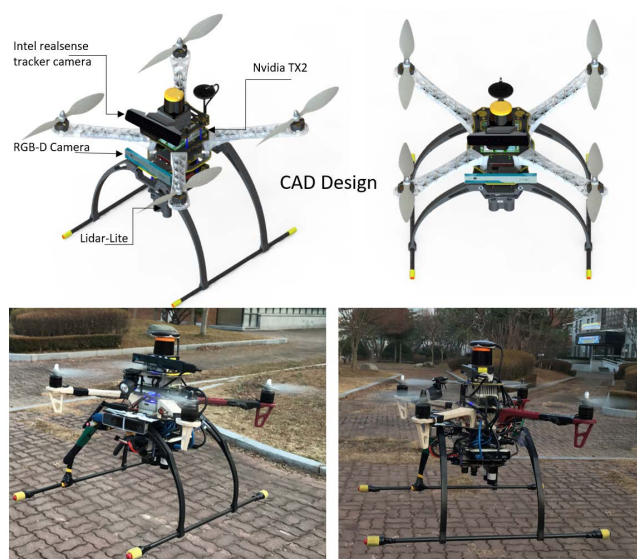


FIGURE 1. The developed MAV platform.

raw sensory data [12], thereby reducing the complexity of classical methods. However, most applied DL algorithms are supervised, and structured datasets are used to train the models. These approaches require a large amount of data and manual labelling, which is time-consuming. To address these limitations, reinforcement learning (RL) has been merged with DL over the past few years, leading to a new research area called deep reinforcement learning (DRL). Examples are the recent Deep-Mind algorithms, deep-Q-network (DQN), and its generic versions, such as the double DQN and dueling DQN. More recently, policy gradient algorithms such as proximal policy optimization (PPO) and the asynchronous advantage actor-critic (A3C) have been proposed, for a continuous action space [13]–[15].

These algorithms have been successfully tested in gaming environments and have shown better performance than that of humans. DRL algorithms are considered powerful and promising tools for automatically mapping high-dimensional sensor information to robot motion commands without referencing the ground truth. They require only a scalar reward function to motivate the learning agent through trial-and-error interactions with the environment to find the best action for each given state.

This paper presents a complete learning system for MAV for autonomous navigation and obstacle avoidance in real indoor and outdoor environments. The system uses a DQN with a new CNN architecture to control the heading and forward motions of the MAV. A collision-free policy was learned in a simulation environment designed using the Gazebo simulator. It was then deployed directly on a real MAV, without any tuning.

Our system uses a Pixhawk autopilot for flight control, two-dimensional LiDAR data, and a depth camera as the input for the algorithm. Our algorithm allows the MAV to follow

predefined target points in the outdoor environment and avoid obstacles by switching between mission flight and DQN modes. All algorithms run in real time onboard the MAV using an NVIDIA Jetson TX2 GPU. Testing was performed in indoor and outdoor environments in simulations and the real world.

The remainder of this paper is organized as follows. Section II introduces related work. Section III describes the aerial robot platform and software architecture. Section IV describes the developed algorithms and the modular architecture. Section V presents the simulation results. Section VI presents real-time experiments, and Section VII concludes the paper.

## II. RELATED WORK

Techniques for autonomous navigation can be classified into map-based and mapless methods. Map-based methods require a global or local map of the environment during flight to make navigation decisions (e.g., [16]–[19]). Mapless navigation primarily concentrates on autonomous navigation, without building a map of the environment. It uses computer vision techniques such as optical flow detection, feature matching using an input image, or an image fused with other sensors such as LiDAR (e.g., [20]–[22]). A survey of the visual navigation methods can be found in [23]. Our methodology is mapless. Unlike other methods, it does not require a prior map of the environment or supervised training or human demonstrations [24]–[26].

Deep neural networks (DNNs) have shown results for solving complex autonomous navigation problems with obstacle avoidance ability using collected datasets for training neural networks. DNNs have been successfully applied to autonomously following trials in an unstructured forest environment using a monocular camera [27], [28]. A stereo-vision algorithm was presented for high-speed navigation in cluttered environments [29]. A CNN was used to learn a navigation strategy by imitating expert demonstrations and learning an end-to-end policy [30], [31]. Autonomous flight was performed in an indoor environment for micro air vehicles (MAV) using a single image by classifying the environment using deep learning modules and then estimating the desired direction to fly (left, right, center) [32], [33].

However, these studies require a vast dataset (data-driven approaches) or human expert demonstration to obtain the desired policy that accomplishes the predefined task. Moreover, the complexity of acquiring a dataset limits its adoption in aerial robotics. Researchers have merged deep learning with reinforcement learning to overcome these problems, which has led to DRL, in which interest is growing because it shows excellent results in many video games [34]. It does not need a dataset or human demonstration, and the agent always tries to maximize the accumulated reward from the interaction with the environment by trial and error.

Many studies have applied DRL to autonomous navigation tasks on different robotic platforms [35]–[37]. Lie [38]

showed that a mobile robot can be trained end-to-end using an asynchronous DRL mapless motion planner. Hong [39] used the A3C algorithm [40] and semantic segmentation to close the gap between simulation and the real world. The obtained control policy was successfully tested on a ground robot. Mirowski [41] presented goal-driven navigation with auxiliary depth prediction and loop closure classification tasks as a reinforcement learning problem. Their approach showed that the robot could avoid obstacles in a complex 3D maze environment. Another study [42] addressed the problem of autonomous mapless navigation in crowded scenes. A policy gradient based reinforcement learning algorithm was used to accomplish this task. The obtained collision-free policy was tested using different mobile robotics platforms to perform free-collision navigation in the real world.

Very few studies have been conducted on the application of DRL for aerial robotics. A previous study [43] addressed the autonomous landing of a UAV on a moving object. Another study in [44], [45] presented a method for controlling a quadrotor with a neural network trained using reinforcement learning technique. Singla *et al.* [46] presented a deep recurrent-Q-network with temporal attention for obstacle avoidance in an indoor environment. The end-to-end DRL was merged with expert data for obstacle avoidance based on monocular images [47]. The work in [48] presented an actor-critic method that allows UAVs to execute navigation tasks in large-scale complex environments. However, only the method was tested in simulation environments, there is no guarantee that it will work in the real world.

Recent research works such as [49], [50] are still limited and only work in specific conditions like light conditions, such as (day/night), or they require some data to be collected by an expert pilot. These limitations motivated us to develop a new system that could operate in different environments. We believe that this is the first time that sensor-fusion information has been used to develop a strategy for autonomous navigation of an aerial robot through end-to-end DRL.

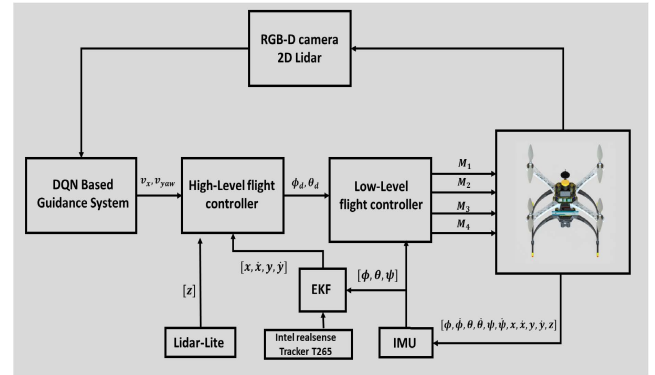
### III. ROBOT PLATFORM AND SYSTEMS DESCRIPTION

In this work, a quadrotor MAV was used as an experimental platform to validate our algorithms. This system is vulnerable to many physical effects, such as aerodynamic effects, gravity, gyroscopic effects, friction, and moment of inertia. The quadrotor configuration is shown in figure 1.

A description of the main mathematical equations of motion of the system is required to understand the main dynamics of the MAV. The derivation of these equations was performed by relying on the following hypotheses [51]:

- 1 The quadrotor body is rigid and symmetric
- 2 The propellers are rigid
- 3 Thrust and drag forces are proportional to the square of the rotors speed
- 4 The center of mass and origin of the coordinate system of the quadrotor structure coincide

Two reference frames are used in this study. The earth reference frame  $R^E$  is defined by axes  $x_E, y_E,$  and  $z_E$  with the  $z_E$



**FIGURE 2. MAV's control system architecture: An EKF is used for accurate state estimation. The DQN guidance system sends velocity commands to the high-level flight controller to adjust the MAV's heading and position. The low-level controller converts the desired attitude to motor commands ( $M_1, M_2, M_3, M_4$ ).**

axis pointing upward, and the body-fixed frame  $R^b$  is defined by axes  $x_b, y_b,$  and  $z_b$ . The attitude of the quadrotor is defined by the orientation of  $R^b$  with respect to  $R^E$  by the rotational matrix  $R$  presented in the equation 1.

$$R = \begin{bmatrix} c\phi c\theta & s\phi s\theta c\psi - s\psi c\phi & c\phi s\theta c\psi + s\psi s\phi \\ s\phi c\theta & s\phi s\theta s\psi + c\psi c\theta & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \quad (1)$$

where  $s(\cdot)$  and  $c(\cdot)$  stand for  $\sin(\cdot)$  and  $\cos(\cdot)$  functions, respectively. Based on the Newton-Euler formulation, the rotational and translational dynamics can be expressed as equation 2

$$\begin{cases} \ddot{x} = (c\phi s\theta c\psi + s\phi s\psi) \frac{1}{m} U_1 \\ \ddot{y} = (c\phi s\theta s\psi - s\phi c\psi) \frac{1}{m} U_1 \\ \ddot{z} = -g + (c\phi c\theta) \frac{1}{m} U_1 \\ \ddot{\phi} = \dot{\theta} \dot{\psi} \left( \frac{I_{yy} - I_{zz}}{I_{xx}} \right) + \frac{1}{I_{xx}} U_2 \\ \ddot{\theta} = \dot{\phi} \dot{\psi} \left( \frac{I_{zz} - I_{xx}}{I_{yy}} \right) + \frac{1}{I_{yy}} U_3 \\ \ddot{\psi} = \dot{\theta} \dot{\phi} \left( \frac{I_{xx} - I_{yy}}{I_{zz}} \right) + \frac{1}{I_{zz}} U_4 \end{cases} \quad (2)$$

where  $U_i$  ( $i = 1, 2, 3, 4$ ) are the altitude, roll, pitch, and yaw control input respectively.  $I_{(xx,yy,zz)}$  is the moment of inertia along each axis. The complete architecture of the control system is shown in Fig. 2. The deep-Q-network plays a key role in intelligent guidance and decision-making by sending a suitable heading rate  $v_{yaw}$  and the forward velocity  $v_x$  to the flight controller. To guarantee a flexible and safe hardware platform, we designed a system using 3D computer-aided design software, including all sensory systems. This reduces the time and cost and increases the design accuracy and reliability. The hardware configuration is shown in figure 1. It includes a quadrotor MAV equipped with an open-source Pixhawk autopilot and an Nvidia Jetson TX2 with an Auvideo J120 carrier board as a companion computer for the autopilot.

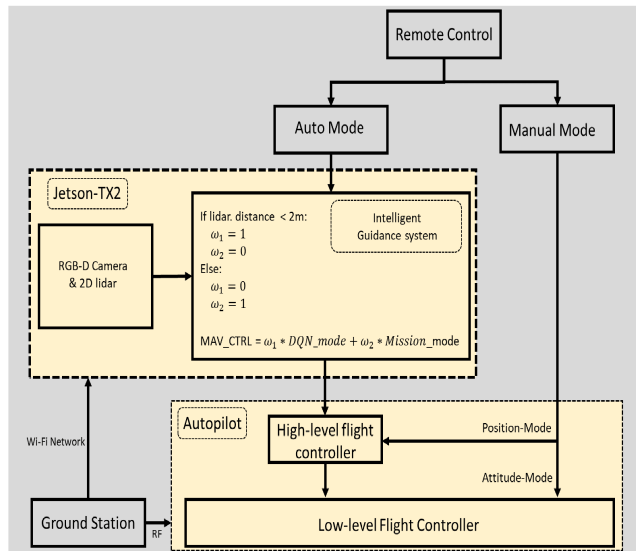


FIGURE 3. Schematic block diagram of the real-time implementation.

The developed algorithm uses a forward-facing Intel Realsense R200 USB RGB-D camera, that provides depth information with a maximum distance of 4 m at 30 Hz. The Hokuyo UST-10LX scanning laser range-finder has a field of view of 270°, a maximum detection distance of 30 m, and an update rate of 30 Hz. The proposed algorithm runs on an NVIDIA Jetson TX2.

The developed MAV can be operated in both indoor and outdoor environments. It is equipped with a forward-facing fish-eye Intel tracking camera T265 module that provides a V-SLAM algorithm for robust state estimation, where the visible features of the ground are used to determine the MAV's position, ground velocity, and orientation, which is sent to an extended Kalman filter (EKF) that runs on the Pixhawk to be fused with other sensor data (e.g., IMU data). We used a LiDAR-Lite V3 laser range finder to obtain accurate altitude feedback in an indoor environment. The Jetson TX2 on-board companion computer was flashed with JetPack 3.1, and the robot operating system (ROS Kinetic) [52] was also installed for easy hardware interface. We adopted Keras as a deep-learning library with TensorFlow in the backend. OpenCV was used for depth image processing. Our system uses the following ROS drivers: the RealSense camera package for interfacing the R200 and T265 USB cameras, the URG node for reading Hokuyo LiDAR data, and MAVROS for communicating with the autopilot.

The human pilot can select two flight modes: auto or manual using a radio transmitter and toggling a switch button. If the auto mode is selected and there are no obstacle within 2 m of the MAV in an outdoor environment, the mission flight mode will be activated to reach specific target points. If there is an obstacles near the MAV within a distance of 2m, the DQN flight mode will be activated. This mode uses fused sensor data from the depth camera and LiDAR measurements, making the drone fully autonomous and avoiding possible obstacles.

After the obstacle avoidance maneuver is completed, the mission flight mode is activated again to correct the MAV path and make it head towards the target points. In the case of an emergency, the pilot can intervene at any time by switching to manual flight mode, as shown in figure 3. The simulation environments were developed using the Gazebo simulator to train the agent realistically. All the sensors were simulated using real specifications. In addition, PX4 provides a software in the loop (SITL) simulation [53], which we used for training.

#### IV. PROPOSED METHOD

Many classical autonomous navigation methods require prior knowledge of the obstacle's location or the environment map. Our algorithm does not require mapping for navigation. This section describes a mapless autonomous navigation technique for MAV. The goal is to determine an optimal control input. The goal is to find an optimal control input  $U_t = \pi_{cf}(s_t)$ , where  $s_t$  is the observation from the fused sensory data at each time step  $t$ . The control input  $U_t$  allows the MAV to avoid obstacles during flight by following the optimal policy,  $\pi_{cf}$ .

##### A. PROBLEM FORMULATION

The MAV navigation problem was formulated as a Markov decision process (MDP), where the MAV interacts with the environment using an RGB-D camera and a LiDAR range finder. A (DQN) with a new CNN architecture is used to solve this problem. The proposed modular architecture contains a collision awareness (CAM) and a collision-free control policy module (CFCPM). The CAM is responsible for sensor fusion and generates a robot observation  $s_t$ . The CFCPM takes the observation  $s_t$  as input and chooses an action  $a \in A$  according to the collision-free policy  $\pi_{cf}$ .

##### B. COLLISION AWARENESS MODULE

The main objective of CAM is to process and fuse sensory data. It then generates observation  $s_t$ , which is passed to the CFCPM, as shown in Figure 4. To reduce the processing time, the field of view of the LiDAR was limited to 90°, which provided a 360 laser beam ray. Using these rays, a binary image with a size of (90, 90) pixels is created by concatenating the rays vertically. This image contains useful two-dimensional (2D) distance information from obstacles within a selected field of view. The depth camera detects tiny obstacles from which LiDAR rays are not reflected. The raw depth image was resized to (90, 90) pixels and then converted to a binary image to close the simulation and real-world gaps. By combining both the sensors, small and distant obstacles can be detected. The obtained images were fused to generate the observation  $s_t$  by concatenating two consecutive images from LiDAR and two successive images from the processed depth. Finally, the total observation  $s_t$  with a size of (90, 90, 4) is obtained and can be forwarded to the CFCPM.



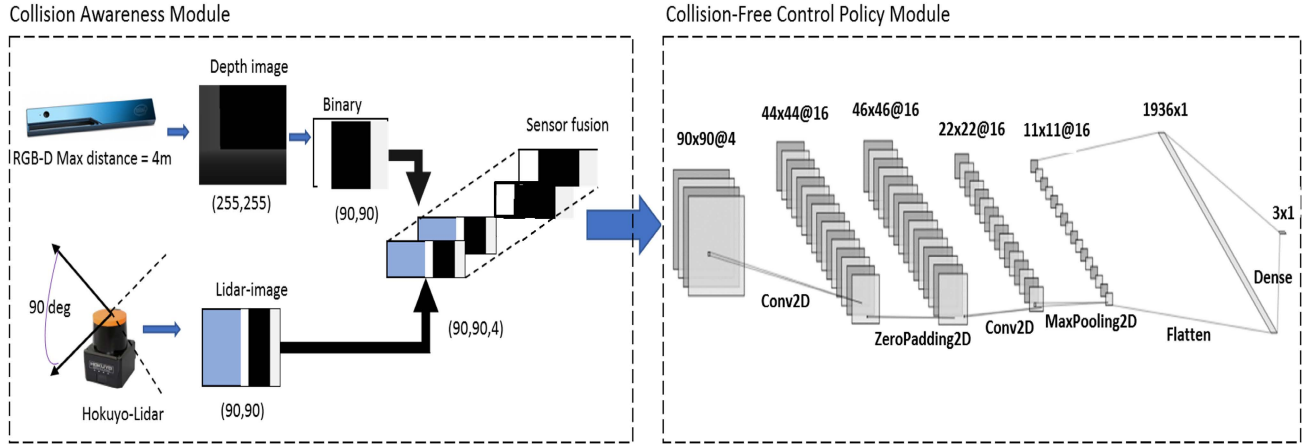


FIGURE 4. The neural networks architecture for the sensor fusion based obstacle avoidance.

C. COLLISION-FREE CONTROL POLICY MODULE

The CFCPM uses the deep Q-network algorithm to find the optimal collision-free policy  $\pi_{cf}^*$ , which allows the MAV to select the best action  $a \in A$  in a given observation  $s_t \in S$  to maximize the total future reward  $R_t = \sum_{\tau=t}^T \gamma^{\tau-t} r_\tau$ , where  $\gamma$  is the discount factor.

The interaction between the MAV and the environment includes a series of actions  $a$ , including three moving commands (right, left, and forward), and observed rewards  $r$  at time  $t = 1, 2, \dots, T$ . During the learning process, the MAV collects information about the environment and learns the optimal collision-free policy  $\pi_{cf}^*$ . These interactions can be represented by a tuple  $(s_1, a_1, r_2, s_2, a_2, \dots, s_T)$ , where  $s_T$  is the terminal state. In our case, the MAV reached the terminal state when the distance from the obstacle was less than 2m, or the maximum step size is exited. The DQN algorithm approximates the action-value function (Q-value) using a deep CNN, as shown in figure 4. It contained two convolution layers with a filter size of (3, 3, 16), stride size of (2, 2), and zero-padding layer to avoid data loss. A ReLu activation function follows each convolution layer, and then the feature maps are transferred to the max-pooling layer to downsample and extract the essential features. The output is forwarded to a fully connected layer to generate three velocity commands in the body frame  $R^b$  of the MAV. Given a collision-free policy  $\pi_{cf}(s_t) = a_t$ , the Q-value can be represented as follows:

$$Q(s_t, a_t)^{\pi_{cf}} = \mathbb{E}[R_t | s_t, a_t, \pi_{cf}] \quad (3)$$

The main objective is to maximize the total future reward  $R_t$ , which can be achieved by maximizing the Q-value function:

$$Q^*(s_t, a_t) = \max_{\pi_{cf}} \mathbb{E}[R_t | s_t, a_t] \quad (4)$$

The optimal Q-value can be decomposed into a Bellman equation as follows:

$$Q^*(s_t, a_t) = \mathbb{E}_{t+1}[r + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) | s_t, a_t] \quad (5)$$

The collision awareness module provides a large observation space (90, 90, 4), which is used to approximate the Q-value. The iterative approximation method presented in equation 5 is not feasible in this case. To overcome this problem, the Q-value was approximated using a deep CNN such that  $Q(s_t, a_t, \theta) \approx Q^*(s_t, a_t)$ .  $\theta$  is the network weight which is updated using reward feedback  $r$  from the environment.

Algorithm 1: Reward Function Definition

```

Read the LiDAR data
Resize the LiDAR data
LiDAR = [1 ... 360]
if Min(LiDAR) > 2m then
    if action = Forward then
        r = 10 * (1/360 * sum_{i=1}^{360} LiDAR_i)
    else
        r = -0.1
else
    r = -10
    
```

The designed reward function is presented in Algorithm 1, based on LiDAR measurements, ensuring safety and fast learning. It also motivates the MAV to move forward in cases where there are no obstacles in front. Suppose that the minimum distance from any obstacle exceeds 2m, and the selected action moves forward. In such a case, a positive reward will support the MAV in moving forward, rather than rotating left or right. A small negative reward will be given if the moving forward is not selected. However, if the minimum LiDAR distance is lower than 2m, a significant negative reward is assigned, the training episode finishes, and the MAV position is reinitialized. The obtained reward  $r$  is used to calculate the target optimal Q-value  $Y_t$ , as follows:

$$Y_t = r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}, \theta_{t-1}) \quad (6)$$

The neural network parameter  $\theta$  is updated by performing stochastic gradient descent on the DQN based on the loss

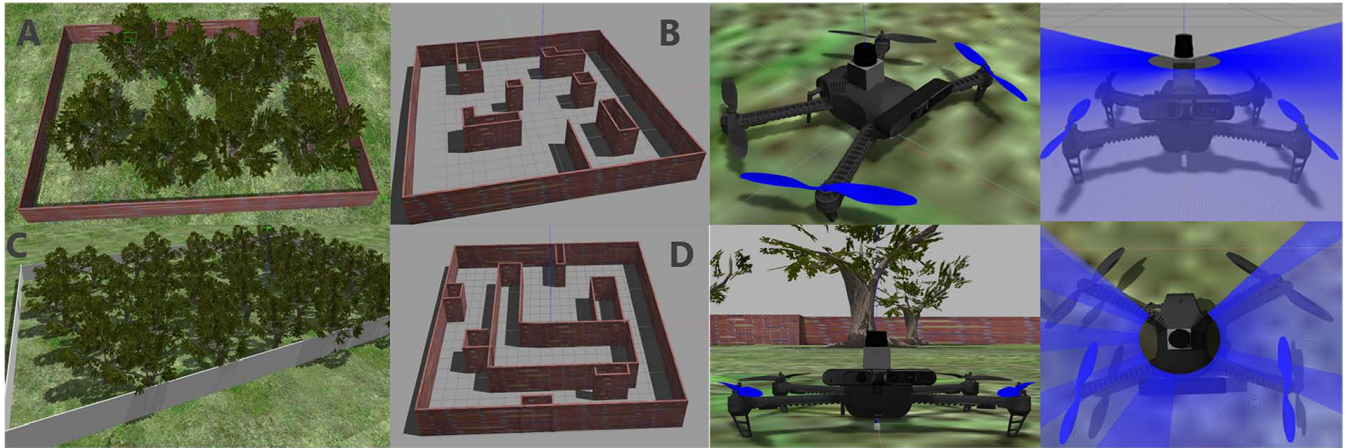


FIGURE 5. Simulation environments: A, B Training environments, C, D testing environments.

function:

$$L_t(\theta_t) = (Y_t - Q(s, a, \theta_t)) \quad (7)$$

We can obtain the gradient of the loss function, as shown below:

$$\nabla_{\theta_t} L_t(\theta_t) = (Y_t - Q(s, a, \theta_t)) \nabla_{\theta_t} Q(s, a, \theta_t) \quad (8)$$

The total workflow of our approach is presented in Algorithm 2. The experience replays from memory, which contains state transitions  $(s_t, a_t, s_{t+1}, r_{t+1})$  were used by taking a random sample with a batch size of 32. Experience replay provides batch updating in an online learning fashion, as follows:

- This tuple  $(s, a, s_{t+1}, r_{t+1})$  is saved in a list to a specific length (64)
- If the replay memory is filled, randomly select a sample size of 32.
- Calculate value update for each sample.

The  $\epsilon$ -greedy training strategy was used by selecting a random action under a certain probability, which allowed the exploitation of the best actions most of the time and kept exploring other actions from time to time. Another target network was used, as follows:

- Initialize the Q-network with parameters (weights)  $\theta$
- Initialize the target network as a copy of the Q-network with different parameters  $\theta_T$ .
- Use the  $\epsilon$ -greedy strategy with the Q-network's Q-values to select action  $a$ .
- Get the reward and new observation  $r_{t+1}, s_{t+1}$ .
- The Q-value of the target network is set to  $r_{t+1}$  if the episode has just been terminated or to  $r_{t+1} + \gamma \max_a Q_{\theta_T}(s_{t+1})$
- Backpropagate the target network's Q-value through the Q-network
- Every C number of iterations, set  $\theta_T = \theta$

---

**Algorithm 2:** DQN for the Autonomous Map-Less Navigation of MAV

---

**initialization :**

- Initialize the CFPCM and CAM
- Initialize replay memory D with size 10000
- Initialize Q-value  $Q(s, a; \theta)$  with random weights  $\theta$
- Initialize the target Q-value  $Y_t$  with weights  $\theta_T = \theta$
- Set minimal distance from any obstacle  $MD = 2m$ .
- Arm the drone
- Switch to manual flight mode
- Takeoff to a fixed altitude 1.5m
- Switch back to auto-mode

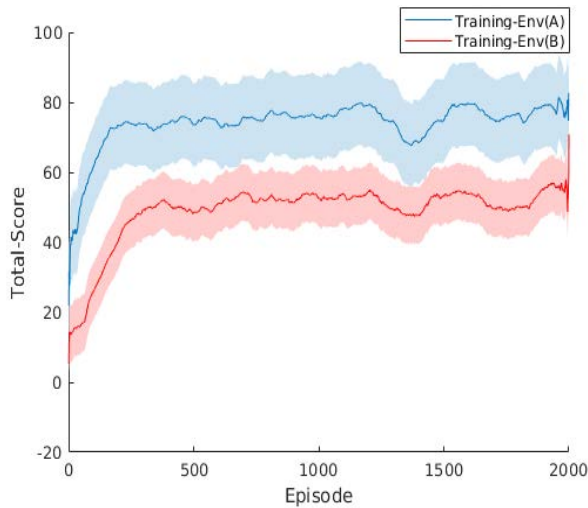
**for**  $episode = 1, M$  **do**

- Set MAV initial position
- Predict the distance from obstacles  $d$

**while**  $d > MD$  **do**

- Get the current observation  $s_t$  using collision awareness module(CAM)
- With probability  $\epsilon$  select a random action  $a_t$
- Otherwise select  $a_t = \operatorname{argmax}_a Q(s_t, a; \theta)$
- Perform the action  $a_t$  get the next observation  $s_{t+1}$  and the reward  $r$
- Store the transition  $(s_t, a_t, r, s_{t+1})$  in D
- Sample random minibatch transitions  $(s_k, a_k, r, s_{k+1})$  with a size of 32 from D
- if**  $d \leq MD$  **then**
  - |  $\bar{Y}_k = r_k$
- else**
  - |  $\bar{Y}_k = r_k + \gamma \max_{a_{t+1}} Q(s_{k+1}, a_{t+1}; \theta_{t-1})$
- Update the neural network weights  $\theta$  using stochastic gradient descent
- Update target network weights every X step

- Save the neural network weights  $\theta$
-



**FIGURE 6.** The learning curves: The blue for indoor and the red outdoor environment.

**TABLE 1.** Training parameters.

parameters	value
Learning rate	0.001
Discount factor $\gamma$	0.99
Epsilon greedy $\epsilon$	1 to 0.1
Replay memory size	10000
Mini-Batch Size	32
Forward speed	0.7 to 1 m/s
Angular speed	0.7 rad/s
Altitude	1.5 m

## V. TRAINING DEEP REINFORCEMENT LEARNING

### A. TRAINING PLATFORM

Several intensive experiments were performed in simulated and real-world environments to evaluate the feasibility of the proposed algorithm. Four simulation environments were built: two to mimic an indoor scenario and two for outdoor scenes based on the Gazebo simulator and SITL, as shown in Figure 5. Environments A and B were used for training, and C and D were used for testing.

Outdoor environments contain randomly placed trees with high densities and walls to limit the training area. An indoor corridor is a corridor with different distances between the walls. The training was performed in both environments A and B using a desktop computer with an Nvidia GTX 1080 GPU, Intel i7-6700 3.40 GHz x 8 CPU, and 16 GB of RAM. A low altitude of 1,5 m and a forward speed of 1 m/s were used in environment A (outdoors). For indoor environment B, the forward speed was 0,7 m/s, and the angular velocity was 0,7 rad/s. All the models were trained using the Keras framework, CUDA 8, and CuDNN 6. The training parameters are listed in Table 1. During the training process, only DQN flight mode is activated. We trained for 2000 episodes within 168 h. The accumulated discount convergence curves are shown in figure 6 for both scenarios (A, B).

The discounted total reward curves were plotted versus the training episodes. Each episode contained 10,000 training

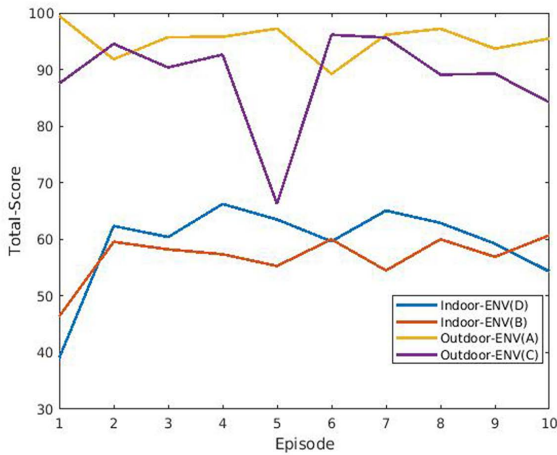
steps. Figure 6 shows that both curves converge to different values. In the outdoor environment, the total discounted reward converged to a higher value of approximately 73 within 250 episodes, whereas in the indoor scenario, the overall score to approximately 49 within 400 episodes. This indicates that the MAV performs better outdoors than indoors in terms of obstacle avoidance and navigation because of the larger open space. During the training process, both the convergence time and total discounted reward depended on the CNN architecture, the observation size, and MAV's speed. Different architectures can lead to different performance, and training at low speeds allows the learning of stable policies. Frame stacking also has a significant impact on performance. The modular architecture was designed to maintain a balance between the learning speed and the success rate. After the training was completed, the obtained models were saved for testing in the same environment and different new unseen environments.

### B. SIM-TO-SIM: TRAINING RESULTS VERIFICATION

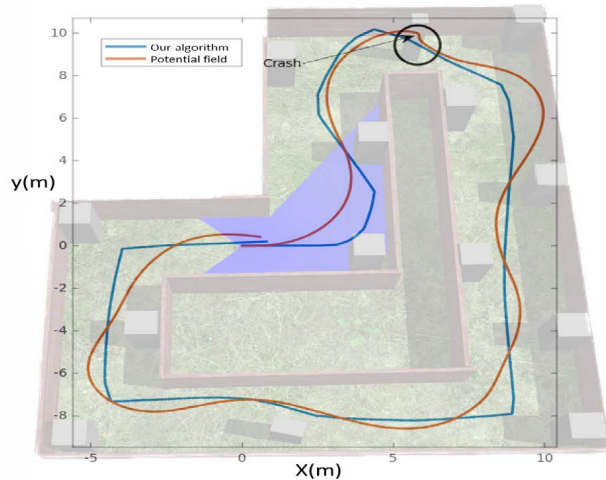
During the testing phase, only DQN flight mode was activated. First, different environments were built to verify the generalization capability of our algorithm for new inexperienced scenarios. The same desktop for the training phase was used for testing. First, we tested the obtained models in an indoor scenario. Two environments are used in this study. The first is the same as that used for training (B), and the second is a different one with a high obstacle density (D). Subsequently, we tested them in an outdoor environment. Similar to the indoor case, two scenarios were used: training environment (A) and a new one (C). Throughout the testing phase, ten episodes of 10,000 steps were used. The total discounted reward for the different cases is shown in Figure 7a. In the outdoor scenarios (A, C), the obtained reward was higher than that in the indoor scenarios (B, D). In outdoor training environment (A), the average total score was positive, approximately 95, which indicates that the MAV had good performance in obstacle avoidance because all the features were already seen and learned in the training phase. However, in the new unseen outdoor environment (C), the average total score is approximately 85. The average overall score for the indoor environment was approximately 60, and the MAV avoided obstacles. To verify the generalization of our algorithm in new environments, more detailed simulations were performed and compared with the traditional potential field method, which uses only laser scan data as input [54].

Figures 7b and, 7c show the simulated environments and the flight paths obtained by the MAV. Example scenario 1, presented in figure 7b, shows that the potential-field method fails. The MAV collided with the obstacle, as indicated by the black circle. Our method shows more robustness for obstacle avoidance and smoother motion, which is not the case for the potential-field approach. Another example, Scenario 2, was presented to confirm the superiority of the proposed algorithm. Figure 7c shows a simulated corridor with different obstacles placed randomly, the MAV starts from an initial

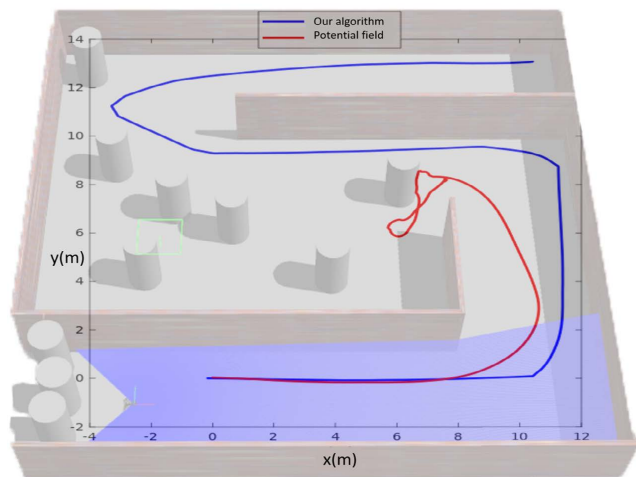




(a) The total testing score of 10 episodes in different environments using the trained models



(b) Paths taken by MAV in example scenario 1



(c) Paths taken by MAV in example scenario 2

FIGURE 7. Total testing score and paths taken by MAV in example scenarios.

position (0, 0) after take-off, it starts to move forward and explore the unknown environment with fixed altitude, the

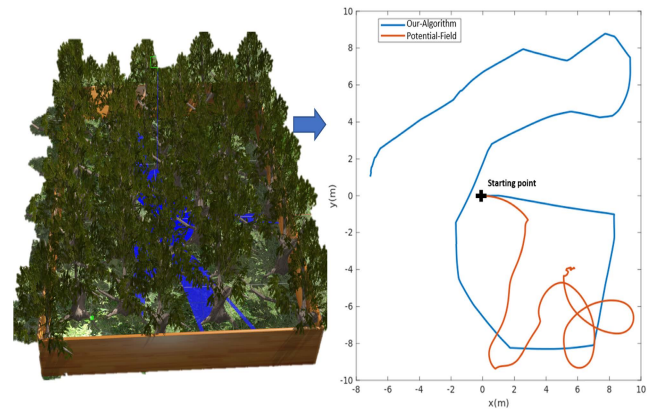


FIGURE 8. Simulated forest environment and the path taken by MAV in example scenarios 3.

reader can notice that both algorithms perform well at the beginning when there are no obstacles in the middle of the corridor, after that at the point coordinate (7, 7) the potential field method failed due to the obstacle’s central location. However, the proposed algorithm can guide the MAV to the end of the corridor. Another simulation in a forest environment was performed. Figure 8 presents the obtained results; the red trajectory shows that the potential field algorithm fails in an open space scenario with no solid walls. Our approach guided the MAV autonomously, as presented with a blue trajectory, using the same models tested in previous scenarios 1,2.

Finally, fully autonomous mission scenarios are developed. The main objective is to ensure that the MAV reaches a pre-defined target point while avoiding collisions with obstacles. Figure 9 shows the first simulated outdoor scenario where the MAV takes off from an initial position (0, 0) and attempts to find a collision-free path to arrive at the target point indicated by a blue circle. Both the DQN flight mode and mission flight mode will be running. When there is no obstacle in front within a distance of 2m, the mission mode lets the MAV head toward the target point and move forward until it detects an obstacle within the predefined range. The DQN-mode executes the obstacle avoidance operation. This maneuver guarantees full autonomy without crashes, as indicated by the blue line. The second scenario simulated an indoor corridor with randomly placed pillars, as shown in Figure 10. This scenario aims to reach multiple target points (T<sub>1</sub>, T<sub>2</sub>, T<sub>3</sub>). The path taken by the MAV is indicated by the blue line, starting from the initial position (0, 0). The MAV reaches the first point, T<sub>1</sub>, without collision, and then it heads back towards points T<sub>2</sub> and T<sub>3</sub>. The reader is recommended to watch the supplementary videos for a better understanding of the simulated scenarios.

VI. SIM-TO-REAL: REAL TIME EXPERIMENTAL VERIFICATION

Several real-time tests were performed to validate our system by deploying trained models from a simulation in a real



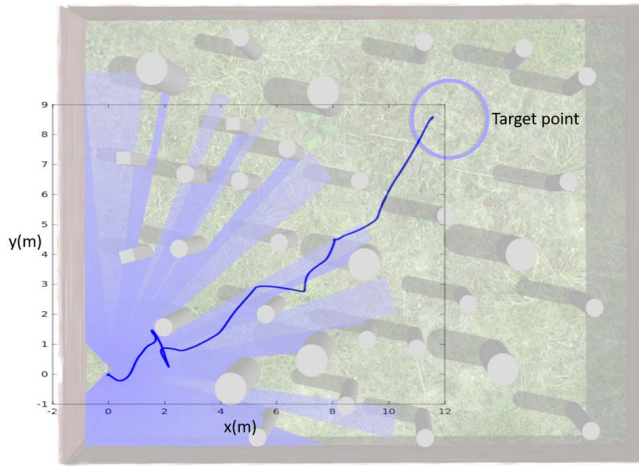


FIGURE 9. Fully autonomous mission in obstacle field 1.

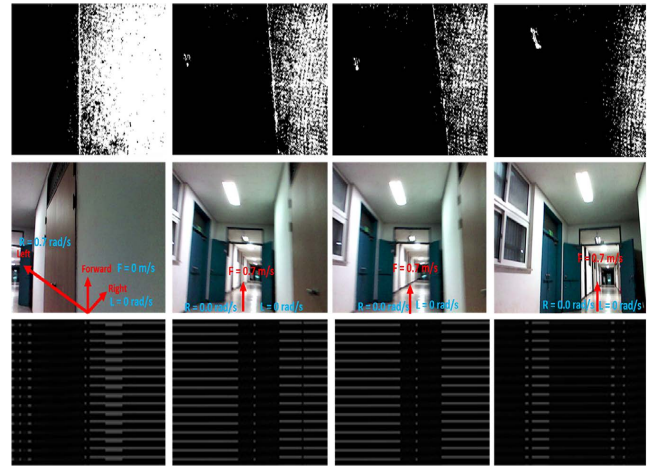


FIGURE 12. Images of the corridor at different possible positions of the MAV. The first, second, and third rows show the images obtained from depth, RGB, and LiDAR, respectively.

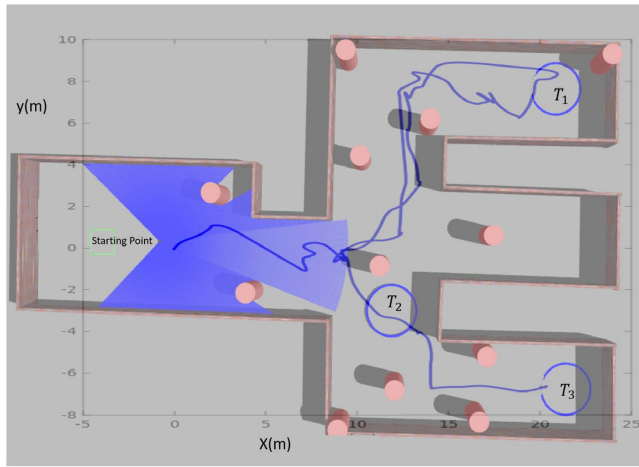


FIGURE 10. Fully autonomous mission in obstacle field 2.

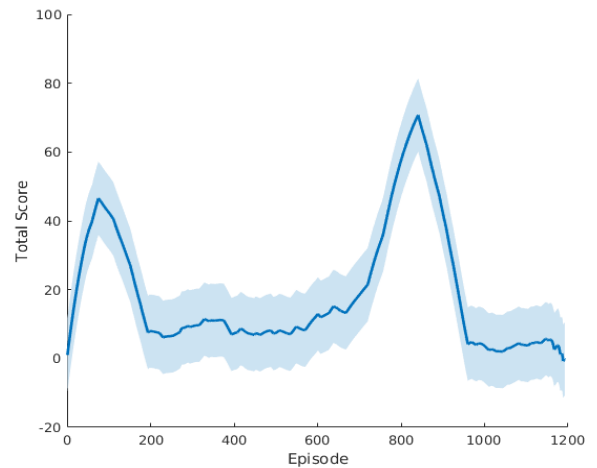


FIGURE 13. The obtained total score during the real test in an indoor corridor.

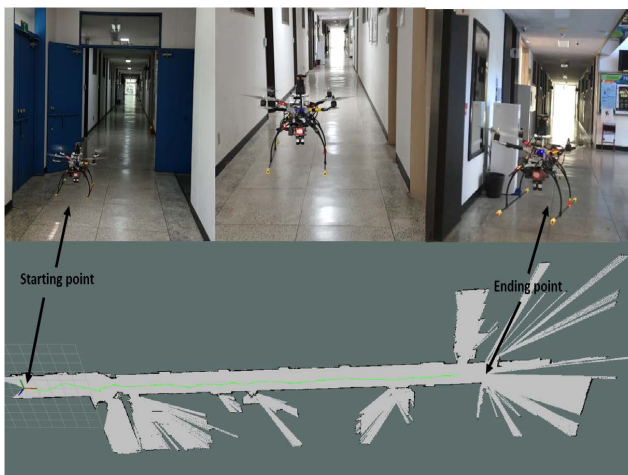


FIGURE 11. Scenario 1: Real experiments in indoor corridor.

MAV. First, only the DQN flight mode was used to explore unknown indoor corridors autonomously. Two case scenarios were investigated (straight and L-shaped corridors). Second,

the experiment was performed in an outdoor environment to validate the fully autonomous mission across a forest trial in the presence of dynamic and static obstacles. The accompanying videos were available for the real-time experiments.

**A. REAL TIME INDOOR VERIFICATION FOR THE DQN-FLIGHT MODE**

The trained DQN model was initially tested in a straight indoor corridor, as shown in figure 11. Figure 12 shows the controlled yaw rates right and left (R, L), linear forward speed (F), some sample images from the depth and LiDAR, and the corresponding RGB from the actual test. In the binarized depth and LiDAR images, the walls were detected as an obstacles with a white color, and the empty space was black. The images obtained from the simulation were similar to real images. The accumulated scores during this test are presented in figure 13. The total score increased before the first 100 episodes because the MAV had landed. After taking off, the drone started to move forward to explore the unknown

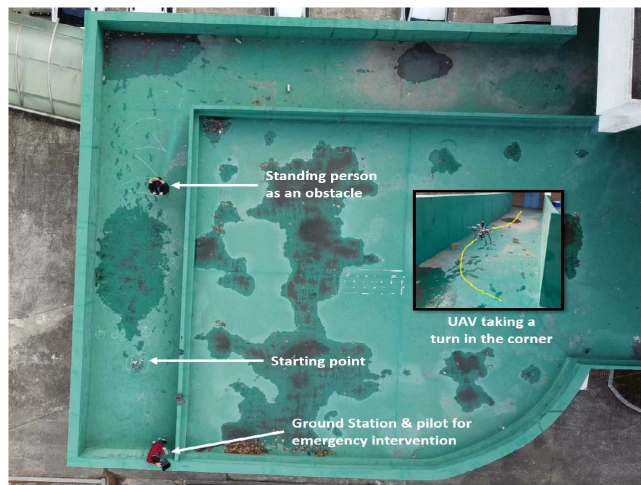


FIGURE 14. Scenario 2: Real experiments in the corridor like environment.

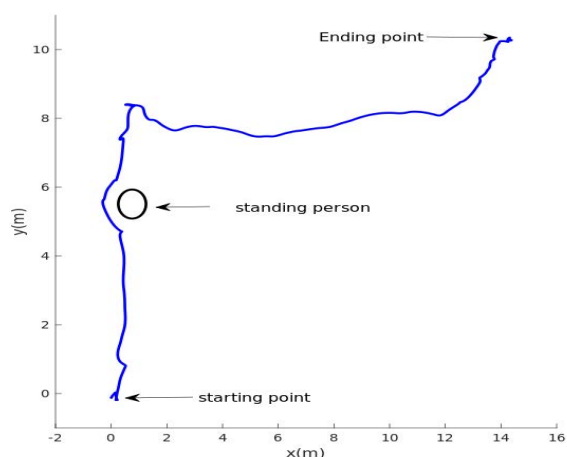


FIGURE 15. Scenario 2: The MAV's current path while avoiding colliding with the walls and the standing person.

corridor, and the overall score was approximately 10 from episode 200 to 700 because the environment was very narrow, and the MAV always tried to find an open space to maximize the accumulated reward. From episodes 800 to 1000, the score increased to 80 because the drone arrived in the open space where the ending point was located. The total positive accumulative reward indicates that the MAV successfully passes through the corridor from the starting point to the ending point and avoids collision with the walls.

Another experimental test scenario two was performed in a corridor-like environment with an L shape as presented in figure 14, after takeoff the MAV was ordered to autonomously explore the corridor within a forward speed of 0.6 m/s. Figure 15 shows the current path taken by the MAV while avoiding the obstacles. The controlled forward velocity and the heading rate are shown in 16. From the second row, we can see that the desired yaw rate presented in red color is switching between two values (+0.8, -0.8) rad/s which correspond to left and right discrete actions, the blue color preset the current heading rate, which is tracking the desired

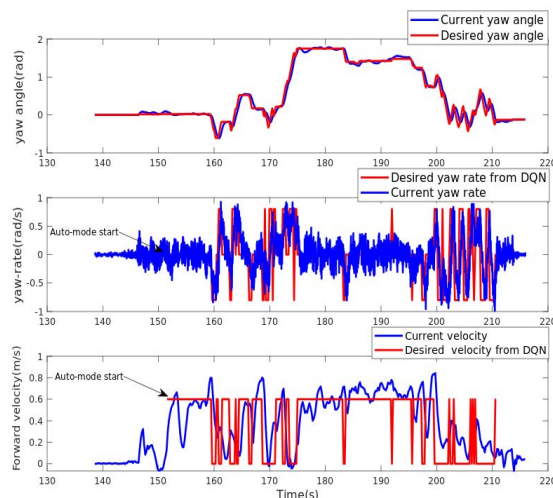


FIGURE 16. Case scenario 2: the first row presents the desired and current yaw angles, the second row shows the desired yaw rate output from DQN-mode, and the current one, the last row shows the desired forward velocity controlled by DQN-mode and the current one.

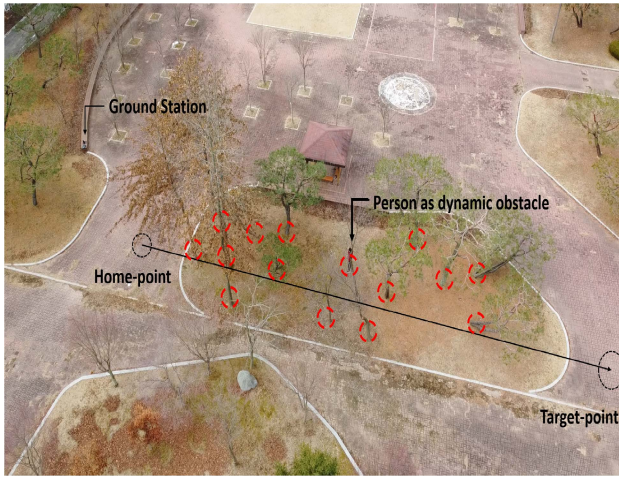
one, making it more clear the yaw angle was plotted in the first row, the MAV follows the desired yaw angle properly which allows it to avoid the obstacles. The last row presents the desired forward linear velocity generated by the DQN mode, corresponding to the moving forward discrete action (0.6 m/s). The actual velocity is indicated in blue color. The MAV attempted to track the desired direction speed within a certain overshoot of 0.2 m/s.

To verify the capabilities of our algorithm, a flight mission was conducted in a forest environment, as shown in Fig.17. Different trees were placed randomly, which acted as static obstacles, as indicated by the red circle in figure 17a. In addition, a walking person is considered to be a dynamic obstacle. The objective of this mission is to reach a specific target point in the world frame and then return to the home point by taking the shortest path while avoiding possible obstacles. The starting and target points are shown in figure 17a with black circles.

### B. REAL TIME FULLY AUTONOMOUS MISSION

In figure 17a, the black line represents the mission's shortest path connecting the starting point to the desired target point across different obstacles. This path requires less time to complete the mission, with less power consumption. However, there are many obstacles. After takeoff, the MAV starts to head toward the target point at a low altitude of 1.5 m, and if there are any static or dynamic obstacles in front, the MAV switches automatically to the DQN flight mode, which allows it to avoid the obstacles after the obstacle avoidance maneuver finishes the MAV head back toward the target using mission flight mode. This operation ensures path shortness. For safety reasons, there is always a human pilot for emergency intervention that toggles a switch button. Figure 18





(a) Aerial view for the test environment



(b) MAV during flight test

FIGURE 17. Case scenario 3: The real environment selected for the full mission flight test.

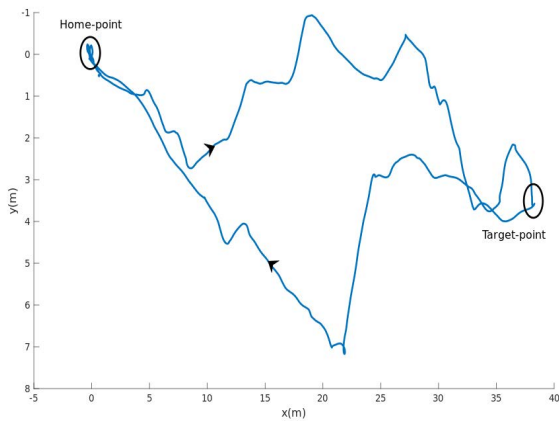


FIGURE 18. The current path taken by the MAV while trying to reach the target point and avoid colliding with obstacles.

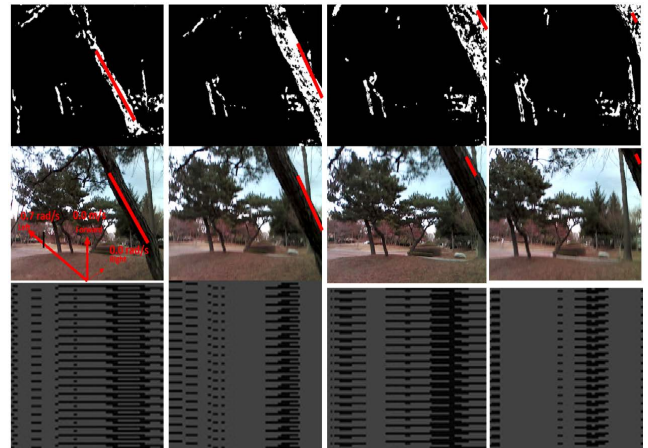


FIGURE 20. Forest trail images at different possible positions of the MAV. The first, second, and third rows show the obtained images from the depth, RGB and LiDAR.

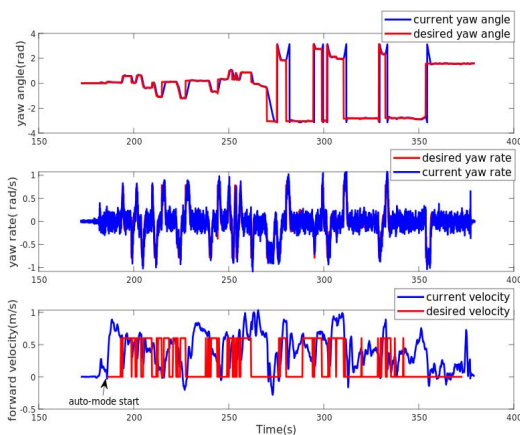


FIGURE 19. Case scenario 3: the first row presents the desired and current yaw angles, the second row shows the desired yaw rate output and the current one, the last row shows the desired forward velocity and the current one.

shows the path taken by the MAV during the entire mission. The MAV reached the target point safely without colliding

with the trees or the walking person; after that, it returned to the starting point. The entire mission lasted approximately 115 s. The controlled heading rate and forward velocity outputs are shown in Fig. 19. After activating the auto-mode, the mission starts, and the MAV heading and forward velocity start to track the desired velocity generated from the auto-mode.

Figure 20 shows sample images collected during the flight test the trees are detected as obstacles with white color, and the images contain strong features of the surrounding environment. After concatenating the depth and LiDAR images, they are passed to the neural network model.

## VII. CONCLUSION

This study presents a novel approach for integrated autonomous navigation and obstacle avoidance for an MAV quadrotor using a modular deep Q-network architecture. The inputs of the proposed algorithm are the LiDAR range finder



distance measurements and a depth image, and it fuses the sensory data through an end-to-end CNN. To determine the optimal collision-free policy, we built different virtual indoor and outdoor simulation environments with realistic simulations of the sensor data. The collision-free policy was entirely trained in the simulation, and then deployed on a real MAV platform. The simulation and real-world experiments show that the proposed method significantly improves the generalization capability of the trained DRL policies. It also shows outstanding performance in terms of the success rate and collision avoidance. Future work will address the application of this approach to navigation problems in 3D space.

## REFERENCES

- [1] F. Nex and F. Remondino, "UAV for 3D mapping applications: A review," *Appl. Geomatics*, vol. 6, no. 1, pp. 1–15, 2014.
- [2] S. Hrabar, "3D path planning and stereo-based obstacle avoidance for rotorcraft UAVs," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2008, pp. 807–814.
- [3] J.-C. Trujillo, R. Munguia, E. Guerra, and A. Grau, "Cooperative monocular-based SLAM for multi-UAV systems in GPS-denied environments," *Sensors*, vol. 18, no. 5, p. 1351, 2018.
- [4] R. Mascaro, L. Teixeira, T. Hinzmann, R. Siegwart, and M. Chli, "GOMSF: Graph-optimization based multi-sensor fusion for robust UAV pose estimation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 1421–1428.
- [5] P. Quillen, J. Muñoz, and K. Subbarao, "Path planning to a reachable state using minimum control effort based navigation functions," *J. Astron. Sci.*, vol. 66, no. 4, pp. 554–581, Dec. 2019.
- [6] M. Radmanesh, M. Kumar, P. H. Guentert, and M. Sarim, "Overview of path-planning and obstacle avoidance algorithms for UAVs: A comparative study," *Unmanned Syst.*, vol. 6, no. 2, pp. 95–118, Apr. 2018.
- [7] (May 2021). *Cuda Toolkit*. [Online]. Available: <http://www.nvidia.com/CUDA>
- [8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.
- [9] Y. Wu *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," 2016, *arXiv:1609.08144*.
- [10] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [11] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, "Learning deep control policies for autonomous aerial vehicles with MPC-guided policy search," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 528–535.
- [12] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Müller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," 2016, *arXiv:1604.07316*.
- [13] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [14] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.
- [15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
- [16] S. H. Vemprala and S. Saripalli, "Collaborative localization for micro aerial vehicles," *IEEE Access*, vol. 9, pp. 63043–63058, 2021.
- [17] J. Kim and S. Sukkarieh, "6DoF SLAM aided GNSS/INS navigation in GNSS denied and unknown environments," *J. Global Positioning Syst.*, vol. 4, nos. 1–2, pp. 120–128, Dec. 2005.
- [18] S. Weiss, D. Scaramuzza, and R. Siegwart, "Monocular-SLAM-based navigation for autonomous micro helicopters in GPS-denied environments," *J. Field Robot.*, vol. 28, no. 6, pp. 854–874, 2011.
- [19] I. Dryanovski, R. G. Valenti, and J. Xiao, "An open-source navigation system for micro aerial vehicles," *Auto. Robots*, vol. 34, no. 3, pp. 177–188, Apr. 2013.
- [20] L. Ma, Y. Liu, and J. Chen, "Using RGB image as visual input for mapless robot navigation," 2019, *arXiv:1903.09927*.
- [21] Y. He and X. Qing, *Automatic Control, Mechatronics and Industrial Engineering: Proceedings of the International Conference on Automatic Control, Mechatronics and Industrial Engineering (ACMIE 2018), October 29-31, 2018, Suzhou, China*. Boca Raton, FL, USA: CRC Press, 2019.
- [22] G. N. DeSouza and A. C. Kak, "Vision for mobile robot navigation: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 2, pp. 237–267, Feb. 2002.
- [23] Y. Lu, Z. Xue, G. S. Xia, and L. Zhang, "A survey on vision-based UAV navigation," *Geo-Spatial Inf. Sci.*, vol. 21, no. 2, pp. 21–32, 2018.
- [24] G. Li, M. Müller, V. Casser, N. Smith, D. L. Michels, and B. Ghanem, "OIL: Observational imitation learning," 2018, *arXiv:1803.01129*.
- [25] M. Müller, V. Casser, N. Smith, D. L. Michels, and B. Ghanem, "Teaching UAVs to race: End-to-end regression of agile controls in simulation," in *Proc. Eur. Conf. Comput. Vis. (ECCV) Workshops*, 2018, pp. 1–18.
- [26] D. Gandhi, L. Pinto, and A. Gupta, "Learning to fly by crashing," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 3948–3955.
- [27] N. Smolyanskiy, A. Kamenev, J. Smith, and S. Birchfield, "Toward low-flying autonomous MAV trail navigation using deep neural networks for environmental awareness," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 4241–4247.
- [28] A. Giusti, J. Guzzi, D. C. Cireşan, F.-L. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Di Caro, D. Scaramuzza, and L. M. Gambardella, "A machine learning approach to visual perception of forest trails for mobile robots," *IEEE Robot. Autom. Lett.*, vol. 1, no. 2, pp. 661–667, Jul. 2016.
- [29] A. J. Barry and R. Tedrake, "Pushbroom stereo for high-speed navigation in cluttered environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 3046–3052.
- [30] A. Loquercio, A. I. Maqueda, C. R. del Blanco, and D. Scaramuzza, "DroNet: Learning to fly by driving," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 1088–1095, Apr. 2018.
- [31] A. Carrio, C. Sampedro, A. Rodriguez-Ramos, and P. Campoy, "A review of deep learning methods and applications for unmanned aerial vehicles," *J. Sensors*, vol. 2017, pp. 1–13, Aug. 2017.
- [32] R. P. Padhy, S. Verma, S. Ahmad, S. K. Choudhury, and P. K. Sa, "Deep neural network for autonomous UAV navigation in indoor corridor environments," *Proc. Comput. Sci.*, vol. 133, pp. 643–650, Jan. 2018.
- [33] D. Ki Kim and T. Chen, "Deep neural network for real-time autonomous indoor navigation," 2015, *arXiv:1511.04668*.
- [34] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.
- [35] A. R. Dooraki and D.-J. Lee, "An end-to-end deep reinforcement learning-based intelligent agent capable of autonomous exploration in unknown environments," *Sensors*, vol. 18, no. 10, p. 3575, Oct. 2018.
- [36] E. Cetin, C. Barrado, G. Muñoz, M. Macias, and E. Pastor, "Drone navigation and avoidance of obstacles through deep reinforcement learning," in *Proc. IEEE/AIAA 38th Digit. Avionics Syst. Conf. (DASC)*, Sep. 2019, pp. 1–7.
- [37] S.-Y. Shin, Y.-W. Kang, and Y.-G. Kim, "Automatic drone navigation in realistic 3D landscapes using deep reinforcement learning," in *Proc. 6th Int. Conf. Control, Decis. Inf. Technol. (CoDIT)*, Apr. 2019, pp. 1072–1077.
- [38] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 31–36.
- [39] Z.-W. Hong, C. Yu-Ming, S.-Y. Su, T.-Y. Shann, Y.-H. Chang, H.-K. Yang, B. H.-L. Ho, C.-C. Tu, Y.-C. Chang, T.-C. Hsiao, H.-W. Hsiao, S.-P. Lai, and C.-Y. Lee, "Virtual-to-real: Learning to control in visual semantic segmentation," 2018, *arXiv:1802.00285*.
- [40] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.
- [41] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, D. Kumaran, and R. Hadsell, "Learning to navigate in complex environments," 2016, *arXiv:1611.03673*.
- [42] T. Fan, X. Cheng, J. Pan, D. Manocha, and R. Yang, "Crowd-Move: Autonomous mapless navigation in crowded scenarios," 2018, *arXiv:1807.07870*.

- [43] A. Rodriguez-Ramos, C. Sampedro, H. Bavle, P. D. L. Puente, and P. Campoy, "A deep reinforcement learning strategy for UAV autonomous landing on a moving platform," *J. Intell. Robot. Syst.*, vol. 93, nos. 1–2, pp. 351–366, Feb. 2019.
- [44] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robot. Autom. Lett.*, vol. 2, no. 4, pp. 2096–2103, Oct. 2017.
- [45] A. Molchanov, T. Chen, W. Hönig, J. A. Preiss, N. Ayanian, and G. S. Sukhatme, "Sim-to-(multi)-real: Transfer of low-level robust control policies to multiple quadrotors," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 59–66.
- [46] A. Singla, S. Padakandla, and S. Bhatnagar, "Memory-based deep reinforcement learning for obstacle avoidance in UAV with limited environment knowledge," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 1, pp. 107–118, Jan. 2021.
- [47] M. A. Anwar and A. Raychowdhury, "NavREn-RL: Learning to fly in real environment via end-to-end deep reinforcement learning using monocular images," in *Proc. 25th Int. Conf. Mechatronics Mach. Vis. Pract. (M2VIP)*, Nov. 2018, pp. 1–6.
- [48] C. Wang, J. Wang, Y. Shen, and X. Zhang, "Autonomous navigation of UAVs in large-scale complex environments: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2124–2136, Mar. 2019.
- [49] K. Kang, S. Belkhale, G. Kahn, P. Abbeel, and S. Levine, "Generalization through simulation: Integrating simulated and real data into deep reinforcement learning for vision-based autonomous flight," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 6008–6014.
- [50] F. Sadeghi and S. Levine, "CAD2RL: Real single-image flight without a single real image," 2016, *arXiv:1611.04201*.
- [51] O. Doukhi, A. R. Fayjie, and D. J. Lee, "Intelligent controller design for quad-rotor stabilization in presence of parameter variations," *J. Adv. Transp.*, vol. 2017, pp. 1–10, Jan. 2017.
- [52] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source robot operating system," in *Proc. ICRA Workshop Open Source Softw.*, Kobe, Japan, 2009, vol. 3, no. 3.2, p. 5.
- [53] L. Meier, D. Honegger, and M. Pollefeys, "PX4: A node-based multi-threaded open source robotics framework for deeply embedded platforms," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 6235–6240.
- [54] B. M. Sathiyaraj, L. C. Jain, A. Finn, and S. Drake, "Multiple UAVs path planning algorithms: A comparative study," *Fuzzy Optim. Decis. Making*, vol. 7, no. 3, p. 257, 2008.



**OUALID DOUKHI** received the Ph.D. degree in mechanical engineering from Kunsan National University, in February 2020. Currently, he is a Postdoctoral Researcher with the Center of Artificial Intelligence and Autonomous Systems (CAIAS), Jeonbuk National University, South Korea. His research interests include state estimation and control, computer vision and machine learning, and deep reinforcement learning.



**DEOK JIN LEE** received the Ph.D. degree in aerospace engineering from Texas A&M University, in May 2005. He worked at the Agency for Defense Development (ADD), from 2006 to 2007, and the Korean Air Research and Development Center, from 2009 to 2011. He was also a Research Professor at the Center for Autonomous Vehicle Research (CAVR), Naval Postgraduate School, Monterey, CA, U.S.A. His research interests include intelligent autonomous systems, machine learning, deep reinforcement learning, sensor fusion and sensor networks, adaptive estimation and control, integrated navigation and localization, and multiagent control.

• • •