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

OUALID DOUKHI, DEOK JIN LEE

1Jeonbuk National University, Department of Mechanical Design Engineering, South Korea (e-mail: doukhisualid@jbnu.ac.kr)
2Jeonbuk National University, Department of Mechanical Design Engineering, South Korea (e-mail: deokjlee@jbnu.ac.kr)

Corresponding author: Deok Jin Lee (e-mail: 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) and Unmanned Vehicle Advanced Research Center (UV ARC) funded by the Ministry of Science and ICT, the Republic of Korea, grants number (2020M3C1C1A01083161), and (2020M3C1C1A0208477212). This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government(MSIT) (2019R1F1A104971113)

ABSTRACT

Flying robots are expected to be used in many tasks, like 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 the vehicle. The developed algorithm can perform collision-free flight in the real world while trained entirely on a 3D simulator. Our method does not require prior expert demonstration or 3D mapping and path planning. It transforms the fused sensory data to a velocity control input for the robot through an end-to-end Convolutional Neural Network (CNN). The obtained policy was compared in simulation with the conventional potential field method. Our approach attains 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 show our system’s effectiveness in flying safely in dynamic outdoor and indoor environments. The supplementary videos for the actual flight tests can be accessed at the following link: 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) were used widely in different applications in both military and civilian domains. Due to their agility and maneuverability, multirotor 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 must operate in unstructured and unknown open-world environments. Designing an autonomous navigation system with the ability to avoid obstacles for MAVs is a long-established research problem [2]. It is difficult to autonomously navigate an under-actuated aerial robot performing specific missions in an unstructured and unknown environment without colliding with any obstacle. Classical approaches are mainly 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, especially in unstructured, dynamic environments where reliable state feedback is unavailable for aerial robots. In addition, building an accurate 3D map of the environment takes considerable computational power. To address these problems and challenges, 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]. Deep learning’s success in solving the problems in artificial intelligence [8]–
It was then deployed directly on a real MAV without any simulation environment designed in the Gazebo simulator. The collision-free policy was learned in a new CNN architecture for controlling the MAV's heading and outdoor environments. The system uses a DQN for autonomous navigation and obstacle avoidance in real-time for each given state.

DRL algorithms are considered a powerful and promising tool for autonomous navigation to overcome these problems, which led to DRL, in which interest is growing since it shows excellent results in many video games [34]. It does not need a map of the environment and does not require supervised training or human demonstration [24]–[26].

Deep Neural Networks (DNNs) showed some results for solving complex autonomous navigation problems with obstacle avoidance using collected data-sets for training the neural networks. DNNs have been successfully applied for autonomously following trails 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]. CNN was used to learn a navigation strategy by imitating expert demonstrations and learning an end-to-end policy [30], [31]. The autonomous flight was done 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 works need a vast data-set (data-driven approaches) or human expert demonstration to obtain the desired policy that accomplishes the predefined task. Moreover, the complexity of acquiring a data-set makes its adoption for aerial robotic still limited. Researchers merged deep learning with reinforcement learning to overcome these problems, which led to DRL, in which interest is growing since it shows excellent results in many video games [34]. It does not need a data-set or human demonstration, and the agent always tries to maximize the accumulated reward from the interaction.
Many studies applied DRL to autonomous navigation tasks for different robotics platforms [35]–[37]. Lie [38] showed that a mobile robot could be trained end-to-end through an asynchronous DRL map-less motion planner. Zhang [39] used the A3C algorithm [40] and semantic segmentation to close the gap between simulation and the real world. The obtained control policy was tested successfully on a ground robot. Mirowski [41] presented goal-driven navigation with auxiliary depth prediction and loop closure classification tasks as a reinforcement learning problem. His approach showed that the robot could avoid obstacles in a complex 3D maze environment. Another study [42] addressed the problem of autonomous map-less navigation in a crowded scene. A policy-gradient-based reinforcement learning algorithm was used to accomplish the task. The obtained collision-free policy was tested with different mobile robotics platforms to perform free collision navigation in the real world.

There are very few works on applications of DRL for aerial robotics. One study [43] addressed the autonomous landing of a UAV on a moving object. Another study in [44], [45] presented a method to control a quadrotor with a neural network trained using reinforcement learning techniques. Abhik Singla [46] presented a deep recurrent Q-network with temporal attention for obstacle avoidance for a UAV in an indoor environment. End-to-end DRL was merged with expert data for obstacle avoidance based on monocular images [47]. The work in [48] presents an actor-critic method that allows UAVs to execute navigation tasks in large-scale complex environments. Although the method was tested only 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 (day/night), or they need some data to be collected by an expert pilot. These limitations motivated us to develop a new system that can work in different environments. We believe it is the first time that sensor-fusion information was used to make a strategy for autonomous navigation of an aerial robotic 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 configuration of the quadrotor is shown in figure 1.

A description of the system’s main mathematical equations of motion is required to understand the MAV’s main dynamics. The derivation of such equations was done by relying on the hypotheses below [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

![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 (M1, M2, M3, M4).](image-url)

Two reference frames were used. The earth reference frame \( R^E \) is defined by axes \( x_E, y_E, z_E \) with \( z_E \) axis pointed upward, and the body-fixed frame \( R^b \) is defined by axes \( x_b, y_b, z_b \). The attitude of the quadrotor is defined by the orientation of \( R^b \) with respect to \( R^E \) by the rotational matrix \( R \):

\[
R = \begin{bmatrix}
  c\phi c\theta c\psi - s\phi s\psi & c\phi s\theta c\psi + s\phi c\psi & c\phi s\psi + s\phi c\psi \\
  s\phi c\theta c\psi + c\phi s\psi & c\phi s\theta c\psi - s\phi c\psi & c\phi s\psi - s\phi c\psi \\
  -s\phi c\theta & s\phi s\theta & c\phi
\end{bmatrix}
\]

(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 are written below in equation 2:

\[
\begin{align*}
\dot{x} &= (c_\phi s_\theta c_\psi + s_\phi s_\psi) \frac{1}{m} U_1 \\
\dot{y} &= (c_\phi s_\theta s_\psi - s_\phi c_\psi) \frac{1}{m} U_1 \\
\dot{z} &= -g + (c_\phi c_\theta) \frac{1}{m} U_1 \\
\dot{\phi} &= \frac{1}{I_{xx} - I_{xy}} \left( I_{yy} \dot{\psi} - I_{yz} \dot{\theta} \right) + \frac{1}{I_{xx}} U_2 \\
\dot{\theta} &= \frac{1}{I_{yy} - I_{yz}} \left( I_{xx} \dot{\psi} - I_{xz} \dot{\phi} \right) + \frac{1}{I_{yy}} U_3 \\
\dot{\psi} &= \frac{1}{I_{xz} - I_{yz}} \left( I_{xy} \dot{\theta} - I_{xx} \dot{\phi} \right) + \frac{1}{I_{xz}} U_4
\end{align*}
\]

(2)

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

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

The developed MAV can be operated in 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 ground’s visible features 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). For accurate altitude feedback in an indoor environment, we used a Lidar-Lite V3 laser range finder. The Jetson TX2 onboard companion computer was flashed with JetPack 3.1 (the Jetson SDK), and the robot operating system (ROS kinetic) [52] was also installed for easy hardware interfacing. We adopt 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: RealSense camera package for interfacing the R200 and T265 USB cameras, 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 auto mode is selected and there is no obstacle within 2m of the MAV in an outdoor environment, mission flight mode will be activated to reach specific target points. If there is an obstacle 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 finishes, mission flight mode will be activated again to correct the MAV path and make it head towards the target-points. In case of an emergency, the pilot can intervene at any time by switching to manual flight mode, as shown in figure 3. Simulation environments were developed using the Gazebo simulator to train the agent realistically. All the sensors were simulated using the real specifications. Besides, PX4 provides a Software In The Loop (SITL) simulation [53], which we used for training.

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

IV. PROPOSED METHOD

Many classical autonomous navigation methods require prior knowledge of the obstacle’s location or environment map. Our algorithm does not require mapping for navigation. This section describes a map-less autonomous navigation technique for a MAV. 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 Lidar range finder. A (DQN) with a new CNN architecture was used to solve this problem. The proposed modular architecture contains a Collision Awareness Module (CAM) and Collision-Free Control Policy Module (CFCPM). The CAM is responsible for sensor fusion and generates the 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 the CAM is to process sensory data and fuse it. It then generates the observation \( s_t \), which is passed to the CFCPM, as presented in figure 4. To reduce the processing time, the field of view for the Lidar was limited to 90 degrees, which provides a 360 laser beam ray. Using these rays, a binary image with a size of (90,90) pixels has been created by concatenating the rays vertically. This image contains useful 2D distance information from obstacles within the selected field of view. The depth camera detects tiny obstacles from which the Lidar rays do not reflect. The raw depth image was resized to (90,90) pixels and then converted to a binary image to close the simulation and real-world gap. By combining both sensors, the small and far obstacles can be detected. The obtained images were fused to generate the observation \( s_t \) by concatenating two consecutive images from the Lidar and two successive from the processed depth. Finally, the total observation \( s_t \) with a
the Q-value function:

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

The main objective is to maximize the total future reward \( R_t \), which we can do by maximizing the Q-value function:

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

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

\[ Q^*(s_t, a_t) = E_{t+1}[r + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) | s_t, a_t] \]  (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 with deep CNN so that \( Q(s_t, a_t) \approx Q^*(s_t, a_t) \). \( \theta \) is the network weights to be 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
    \[ r = 10 \times \left( \frac{1}{360} \sum_{i=1}^{360} \text{Lidar}_i \right) \]
  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 is no obstacle in front. Suppose the minimum distance from any obstacle exceeds \( 2m \), and the selected action moves forward. In that case, a positive reward will support the MAV to move forward rather than rotating left or right. A small negative reward will be given if moving forward is not selected. Still, if the minimum Lidar distance is lower than \( 2m \), a significant negative reward is assigned, the training episode finishes, and the MAV position will be reinitialized. The obtained reward \( r \) is used to calculate the
We can obtain the gradient of the loss function, as shown in Algorithm 2. The experience replays from memory which contains state transitions \((s_t, a_t, s_{t+1}, r_{t+1})\) was used, by taking a random sample with a batch size of 32. Experience replay provides batch updating in an online learning fashion as follows.

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

The \(\epsilon\)-greedy training strategy was used by selecting a random action under a certain probability, this allows exploiting the best actions most of the time and keeping exploring other actions from time to time. Also, 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 \(a_t\).
- Get the reward and new observation \(r_{t+1}, s_{t+1}\).
- The target network’s Q-value will be set to \(r_{t+1}\) if the episode has just been terminated or to \(r_{t+1} + \gamma\max_{a} Q_{\theta_t}(s_{t+1})\).

The neural network parameters \(\theta\) are updated by performing stochastic gradient descent on the DQN based on the loss function:

\[
L_t(\theta_t) = (Y_t - Q(s, a, \theta_t)) \tag{7}
\]

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

\[
\nabla \theta_t L_t(\theta_t) = (Y_t - Q(s, a, \theta_t)) \nabla \theta_t Q(s, a, \theta_t) \tag{8}
\]

The total workflow for our approach is presented in Algorithm 2. Experience replay provides batch updating in an online learning fashion as follows.

- Initialize the CFCPM and CAM
- Initialize reply 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 \(a_t = \arg\max_{a} Q(s_t, a; \theta)\)
- 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 size of 32 from D

if \(d \leq MD\) then

\[
Y_k = r_k
\]

else

\[
Y_k = r_k + \gamma\max_{a} Q_{\theta_t}(s_{k+1}, a_{t+1}; \theta_{t-1})
\]

- Update the neural network weights \(\theta\) using stochastic gradient descent
- Update target network weights every X step
- Save the neural network weights \(\theta\)
Several intensive experiments were performed in simulated and real-world environments to evaluate our algorithm’s feasibility. Four simulation environments have been built: two to mimic an indoor scenario and two for outdoor scenes based on the Gazebo simulator and SITL, as presented in figure 5. Environments A and B are for training, and C and D are for testing.

The outdoor environments contain randomly placed trees with high density and walls to limit the training area. The indoor one 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 (outdoor). For indoor environment B, the forward speed was 0.7 m/s, and the angular velocity was 7 rad/s. All models were trained with the Keras framework, CUDA 8, and CuDNN 6. Table 1 shows the training parameters in detail. During the training process, only the DQN-flight mode is activated. We trained for 2000 episodes within 168 hours. The accumulated discounted reward converging curves are shown in figure 6 for both scenarios (A, B).

**V. TRAINING DEEP REINFORCEMENT LEARNING**

**A. TRAINING PLATFORM**

The discounted total reward curves are plotted versus the training episodes. Each episode contains a 10,000 training steps. Figure 6 shows that both curves converge to different values. In the outdoor environment, the total discounted reward converges to a higher value around 73 within 250 episodes, while in the indoor scenario, the overall score converges around 49 within 400 episodes. This indicates that the MAV has better performance outdoors than indoors in terms of obstacle avoidance and navigation because of the bigger open space. During the training process, both the convergence time and total discounted reward depend on the CNN architecture, the observation size, and MAV’s speed.

| Parameters       | Value  |
|------------------|--------|
| Learning rate    | 0.001  |
| Discount factor γ| 0.99   |
| Epsilon greedy ε | 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  |

**TABLE 1: Training Parameters**

Different architectures can lead to different performances; training at low speeds allows learning stable policies. The frame stacking also has a significant impact on the performance. The modular architecture was designed to keep the balance between the learning speed and the success rate. After the training finished, the obtained models were saved for testing in the same environments and different new unseen environments.

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

During the testing phase, only the DQN-flight mode is activated. First, different environments have been built to check the generalization capability of our algorithm to new inexperienced scenarios. The same desktop for the training phase was used for testing. Firstly, we tested the obtained models in an indoor scenario. Two environments were used. The first is the same one used for training (B), and the second is a different one with a high obstacle density (D). After that, we tested them in outdoor environments. Like the indoor case, two scenarios were used: the training environment (A) and a new one (C). Throughout the testing phase, 10 episodes with 10,000 steps were used. The total discounted reward for different cases are presented in figure 7a. In the outdoor cases (A, C), the obtained reward is higher than in the indoor scenarios (B, D). In the outdoor training environment (A), the average total score was positive, about 95, which indicates that the MAV has 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 was about 85. The average overall score in the indoor environments was about 60, and the MAV avoided obstacles. To verify our algorithm’s generalization in new environments, more detailed simulations have been performed and compared with...
The total testing score of 10 episodes in different environments using the trained models.

(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

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

The traditional potential field method, which uses only laser scan data as input [54].

Figures 7b, 7c show the simulated environments and the flight paths taken by the MAV. An example scenario 1 presented in figure 7b shows that the potential field method fails. The MAV has collided with the obstacle as pointed in a black circle. Our method shows more robustness for obstacle avoidance and more smooth motion, which is not the case for the potential field approach. Another example, scenario 2 was presented to confirm the superiority of our algorithm; figure 7c shows a simulated corridor with different obstacles placed randomly, the MAV starts from an initial position (0, 0) after take-off, it starts to move forward and explore the unknown environment with fixed altitude, the 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, our algorithm was able to guide the MAV until the end of the corridor. Another simulation in a forest environment has been 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.

At last, fully autonomous mission scenarios have been performed. The mean objective is to make the MAV reach a predefined target point while avoiding colliding with the obstacles. Figure 9 shows the first simulated outdoor scenario where the MAV takeoff from an initial position (0, 0) and tries to find a collision-free path to arrive at the target point indicated with a blue circle. Both 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 the blue path...
FIGURE 9: Fully autonomous mission in obstacle field 1

FIGURE 10: Fully autonomous mission in obstacle field 2

show. The second scenario simulates an indoor corridor with pillars placed randomly, as figure 10 presents. This scenario aims to reach multitarget points \( (T_1, T_2, T_3) \). The path taken by the MAV is shown in the blue line, starting from an initial position \((0, 0)\). The MAV reaches the first point, \( T_1 \), without colliding, then it heads back towards points \( T_2 \) and \( T_3 \). The reader is recommended to watch the supplementary videos for more 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 the trained models from the simulation in a real MAV. Firstly, only the DQN-flight mode is used to explore unknown indoor corridors autonomously. Two case scenarios were investigated (straight and L shape corridors). Secondly, the experiment has been performed in an outdoor environment to validate the fully autonomous mission across a forest trail in the presence of dynamic and static obstacles. Accompanying videos are available for real-time experiments.

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

The trained DQN model was initially tested in an indoor straight 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 are detected as an obstacle with white color, and the empty space is black. The obtained images from the simulation are similar to the real images. The accumulated score during this test is presented in figure 13. The total score increased before the first 100 episodes because the MAV was landed. After taking off, the drone starts to move forward to explore the unknown corridor, the overall score was around 10 from episode 200 to 700 because the environment was very narrow, the MAV always tries 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 an open space where the ending point is located. The total positive accumulative reward indicates that the MAV successfully passes the corridor from the starting point to the ending point and avoids colliding 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 rad/s, −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 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 shows 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 shown in blue color. The MAV tries to track the desired direction speed within a certain overshot of 0.2 m/s.

To verify the capabilities of our algorithm, a flight mission has been conducted in a forest environment, see figure 17. Different trees are placed randomly, which act as a static ob-
stactle, as indicated by a red circle in figure 17a. Additionally, a walking person was considered a dynamic obstacle. This mission's mean objective is to reach a specific target point given in the world frame then go back 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 finish the mission with less power consumption. However, it contains many obstacles. After take-off, the MAV starts to head toward the target-point with a low altitude of 1.5m, and if there is any static or dynamic obstacle in front, the MAV switches automatically to 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 the path shortness. For safety reasons, there is always a human pilot for emergency intervention by toggling a switch button. Figure 18 presents the path taken by the MAV during the whole mission. The MAV reached the target point safely without colliding with the trees and the walking person; after that, it returned to the starting point. The whole mission took around 115 second. The controlled heading rate and the forward velocity outputs are presented in figure 19. After activating the auto-mode, the mission starts, and the MAV heading and forward velocity

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 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.
start to track the desired one 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 paper presents a novel approach for integrated autonomous navigation and obstacle avoidance for a MAV quadrotor using a modular Deep Q-Network architecture. The proposed algorithm’s inputs are Lidar range finder distance measurements and a depth image, and it fuses the sensory data through an end-to-end CNN. To find 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 simulation and then deployed on a real MAV platform. The simulation and real-world experiments show that the proposed method significantly improves the trained DRL policies’ generalization capability. It also shows outstanding performance in terms of the success rate and collision avoidance. Future work will address the applications of this approach to the navigation problem in a 3D space.

REFERENCES

[1] F. Nex and F. Remondino, “Uav for 3d mapping applications: a review,” Applied geomatics, vol. 6, no. 1, pp. 1–15, 2014.
[2] S. Harbar, “3d path planning and stereo-based obstacle avoidance for rotorcraft uavs,” in 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 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. Hinze, R. Siegwart, and M. Chli, “Gomaf: Graph-optimization based multi-sensor fusion for robust uav pose estimation,” in 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 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,” The Journal of the Astronautical Sciences, vol. 66, no. 4, pp. 584–581, 2019.
[6] M. Radanesh, M. Kumar, P. H. Guenter, and M. Sarim, “Overview of path-planning and obstacle avoidance algorithms for uavs: a comparative study,” Unmanned systems, vol. 6, no. 02, pp. 95–118, 2018.
[7] “Cuda toolkit,” May 2021. 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 Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 779–788.
[9] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey et al., “Google’s neural machine translation system: Bridging the gap between human and machine translation,” arXiv preprint arXiv:1609.08144, 2016.
[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 2016 IEEE international conference on robotics and automation (ICRA). IEEE, 2016, pp. 528–535.
[12] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang et al., “End to end learning for self-driving cars,” arXiv preprint arXiv:1609.09780, 2016.
[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski 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,” arXiv preprint arXiv:1509.02971, 2015.
[15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” arXiv preprint arXiv:1707.06347, 2017.
[16] S. H. Vempala and S. Saripalli, “Collaborative localization for micro aerial vehicles,” IEEE Access, vol. 9, pp. 63043–63058, 2021.
[17] J. Kim, S. Sukkarieh et al., “6dof slam aided gnsns navigation in gns denied and unknown environments,” Positioning, vol. 1, no. 09, 2005.
[18] S. Weiss, D. Scaramuzza, and R. Siegwart, “Monocular-slam-based navigation for autonomous micro helicopters in gps-denied environments,” Journal of Field Robotics, vol. 28, no. 6, pp. 854–874, 2011.
[19] I. Dryanovski, R. G. Valentini, and J. Xiao, “An open-source navigation system for micro aerial vehicles,” Autonomous Robots, vol. 34, no. 3, pp. 177–188, 2013.
[20] L. Ma, J. Chen et al., “Using rgb image as visual input for mapless robot navigation,” arXiv preprint arXiv:1903.09927, 2019.
[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, Szczecin, China. CRC Press, 2019.
[22] G. Ni DeSouza and A. C. Kak, “Vision for mobile robot navigation: A survey,” IEEE transactions on pattern analysis and machine intelligence, vol. 24, no. 2, pp. 237–267, 2002.
[23] Y. Lu, Z. Xue, G.-S. Xia, and L. Zhang, “A survey on vision-based uav navigation,” Geo-spatial information science, vol. 21, no. 1, pp. 21–32, 2018.
[24] G. Li, M. Müller, V. Casser, N. Smith, D. L. Michels, and B. Ghanem, “Oil: Observational imitation learning,” arXiv preprint arXiv:1803.01129, 2018.
[25] M. Muller, V. Casser, N. Smith, D. L. Michels, and B. Ghanem, “Teaching uavs to race: End-to-end regression of agile controls in simulation,” in Proceedings of the European Conference on Computer Vision (ECCV) Workshops, 2018, pp. 0–0.
[26] D. Gandhi, L. Pinto, and A. Gupta, “Learning to fly by crashing,” in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017, pp. 3948–3955.
[27] N. Smolyanskiy, A. Kamenev, J. Smith, and S. Birchfield, “Toward low-latency object detection in real-time,” arXiv preprint arXiv:1609.02971, 2015.
[28] A. Giusti, J. Guzzi, D. C. Cire¸ san, F.-L. He, J. P. Rodríguez, F. Fontana, G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski et al., “Google’s neural machine translation system: Bridging the gap between human and machine translation,” arXiv preprint arXiv:1609.08144, 2016.
[29] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” nature, vol. 521, no. 7553, pp. 436–444, 2015.
[29] A. J. Barry and R. Tedrake, “Pushbroom stereo for high-speed navigation in cluttered environments,” in 2015 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2015, pp. 3046–3052.

[30] A. Loquercio, A. I. Maqueda, C. R. Del-Blanco, and D. Scaramuzza, “Dronet: Learning to fly by driving,” IEEE Robotics and Automation Letters, vol. 3, no. 2, pp. 1088–1095, 2018.

[31] A. Carrio, C. Sampedro, A. Rodriguez-Ramos, and P. Campoy, “A review of deep learning methods and applications for unmanned aerial vehicles,” Journal of Sensors, vol. 2017, 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,” Procedia computer science, vol. 133, pp. 643–650, 2018.

[33] D. K. Kim and T. Chen. “Deep neural network for real-time autonomous indoor navigation,” arXiv preprint arXiv:1511.04668, 2015.

[34] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” in 2013 30th International Conference on Machine Learning (ICML). ACM, 2013.

[35] L. P. Padhy, S. Verma, S. Ahmad, S. K. Choudhury, and P. K. Sa, “Deep reinforcement learning based intelligent agent capable of autonomous exploration in unknown environments,” Sensors, vol. 18, no. 10, p. 3575, 2018.

[36] E. Cetin, C. Barrado, G. Munoz, M. Macias, and E. Pastor, “Drone navigation and avoidance of obstacles through deep reinforcement learning,” in 2019 IEEE/ION 38th Annual Position, Location and Navigation Symposium (PLANS). IEEE, 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 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT). IEEE, 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 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017, pp. 31–36.

[39] Z.-M. 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 et al., “Virtual-to-real: Learning to control in visual semantic segmentation,” arXiv preprint arXiv:1802.00285, 2018.

[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 International conference on machine learning. PMLR, 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 et al., “Learning to navigate in complex environments,” arXiv preprint arXiv:1611.03673, 2016.

[42] T. Fan, X. Cheng, J. Pan, D. Manocha, and R. Yang, “Crowdmove: Autonomous mapless navigation in crowded scenarios,” arXiv preprint arXiv:1807.07670, 2018.

[43] A. Rodriguez-Ramos, C. Sampedro, H. Bavel, P. De La Puente, and P. Campoy, “A deep reinforcement learning strategy for uav autonomous landing on a moving platform,” Journal of Intelligent & Robotic Systems, vol. 93, no. 1-2, pp. 351–366, 2019.

[44] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, “Control of a quadrotor with reinforcement learning,” IEEE Robotics and Automation Letters, vol. 2, no. 4, pp. 2096–2103, 2017.

[45] A. Molchanov, T. Chen, W. Höng, J. A. Preiss, N. Ayaniyan, and G. S. Sukhatme, “Sun-to-(multi)-real: Transfer of low-level robust control policies to multiple quadrotors,” in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2019, pp. 59–66.

[46] A. Singla, S. Padakanda, and S. Bhatnagar, “Memory-based deep reinforcement learning for obstacle avoidance in uav with limited environment knowledge,” IEEE Transactions on Intelligent Transportation Systems, 2019.

[47] M. A. Anwar and A. Raychowdhury, “Navren-r: Learning to fly in real environment via end-to-end deep reinforcement learning using monocular images,” in 2018 25th International Conference on Mechatronics and Machine Vision in Practice (M2VIP). IEEE, 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 Transactions on Vehicular Technology, vol. 68, no. 3, pp. 2124–2136, 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 2019 international conference on robotics and automation (ICRA). IEEE, 2019, pp. 6008–6014.

[50] F. Sadeghi and S. Levine, “Caddrl: Real single-image flight without a single real image,” arXiv preprint arXiv:1611.04201, 2016.

[51] O. Doukhi, A. R. Fayjie, and D. J. Lee, “Intelligent controller design for quad-rotor stabilization in presence of parameter variations,” Journal of Advanced Transportation, vol. 2017, 2017.

[52] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng et al., “Ros: An open-source robot operating system,” in ICRA workshop on open source software, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[53] L. Meier, D. Honegger, and M. Pollefeys, “P4x: A node-based multithreaded open source robotics framework for deeply embedded platforms,” in 2015 IEEE international conference on robotics and automation (ICRA). IEEE, 2015, pp. 6235–6240.

[54] B. M. Sathyaraj, L. C. Jain, A. Finn, and S. Drake, “Multiple uavs path planning algorithms: a comparative study,” Fuzzy Optimization and Decision Making, vol. 7, no. 3, p. 257, 2008.

---

DEOK-JIN LEE received his Ph.D. degree in Mechanical Engineering from Kunsan National University in February 2020. Currently, he is a Postdoctoral Researcher at the center of Artificial Intelligence and Autonomous Systems (CAIAS) at Jeonbuk National University, Korea. His research interests include state estimation and control, computer vision and machine learning, and deep reinforcement learning.

QUALID DOUKHI received his Ph.D. degree in Mechanical Engineering from Kunsan National University in May 2005. He worked for Agency for Defense Development (ADD) from 2006 to 2007 and Korean Air R&D 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, and integrated navigation and localization, multi-agent control.