Chinese Journal of Aeronautics, (2024), 37(3): 237–257

Chinese Society of Aeronautics and Astronautics
& Beihang University

Chinese Journal of Aeronautics

cja@buaa.edu.cn
www.sciencedirect.com

FULL LENGTH ARTICLE

Deep-reinforcement-learning-based UAV autonomous navigation and collision avoidance in unknown environments

Fei WANG [a], **Xiaoping ZHU** [a,*], **Zhou ZHOU** [b], **Yang TANG** [a]

[a] *School of Astronautics, Northwestern Polytechnical University, Xi'an 710072, China*
[b] *School of Aeronautics, Northwestern Polytechnical University, Xi'an 710072, China*

Received 24 February 2023; revised 28 March 2023; accepted 1 June 2023
Available online 16 October 2023

KEYWORDS

Faster R-CNN model;
Replay memory Data Deposit Mechanism (DDM);
Two-part training approach;
Image-based Autonomous Navigation and Collision Avoidance (ANCA);
Unmanned Aerial Vehicle (UAV)

Abstract In some military application scenarios, Unmanned Aerial Vehicles (UAVs) need to perform missions with the assistance of on-board cameras when radar is not available and communication is interrupted, which brings challenges for UAV autonomous navigation and collision avoidance. In this paper, an improved deep-reinforcement-learning algorithm, Deep Q-Network with a Faster R-CNN model and a Data Deposit Mechanism (FRDDM-DQN), is proposed. A Faster R-CNN model (FR) is introduced and optimized to obtain the ability to extract obstacle information from images, and a new replay memory Data Deposit Mechanism (DDM) is designed to train an agent with a better performance. During training, a two-part training approach is used to reduce the time spent on training as well as retraining when the scenario changes. In order to verify the performance of the proposed method, a series of experiments, including training experiments, test experiments, and typical episodes experiments, is conducted in a 3D simulation environment. Experimental results show that the agent trained by the proposed FRDDM-DQN has the ability to navigate autonomously and avoid collisions, and performs better compared to the FR-DQN, FR-DDQN, FR-Dueling DQN, YOLO-based YDDM-DQN, and original FR output-based FR-ODQN.

1. Introduction

Unmanned Aerial Vehicles (UAVs) are autonomous or semi-autonomous aerial vehicles with high maneuverability, good stealth and adaptability, which can replace humans to perform some dangerous military missions.[1–3] For example, UAVs can replace humans in search and rescue missions to land and sea as well as reconnaissance missions.[4–7] In some military application scenarios, the communication between a UAV and a ground station may be disturbed, and the UAV cannot perform its mission by manual control, which requires the UAV to have autonomous navigation capabilities. In addition, in order to cope with military scenarios where an on-board radar is not available or fails, a UAV should have the ability to avoid obstacles through multiple sensors. On-board electro-optical

* Corresponding author.
E-mail address: zhuxp@nwpu.edu.cn (X. ZHU).

https://doi.org/10.1016/j.cja.2023.09.033

Nomenclature

A_{U}	action space set
a_{U}	action of the UAV
d_{c}	distance between the UAV and the field of view
D_{g}	distance between the UAV and the destination in 3D space
D_{g2d}	distance between the UAV and the destination in 2D space
D_{obs}	distance between the UAV and the obstacle in 3D space
D'_{OtoU}	relative distance between the UAV and the obstacle in the image
DE	danger experience set
H_{UtoG}	altitude difference between the UAV and the destination
$L(\theta)$	loss function
m	size of mini-batch
N	exploration rate reset cycle
$\boldsymbol{P}_{\mathrm{g}}$	position of the destination
$\boldsymbol{P}_{\mathrm{u}}$	position of the UAV
$\boldsymbol{P}_{\mathrm{obs}}$	position of the obstacle
$\boldsymbol{P}'_{\mathrm{obs}}$	position of the obstacle in the field of view
P_{sa}	state transition probability
Q	prediction network
$\hat{Q}$	target network
r_{U}	total reward function
r_{arrived}	reward function regarding whether the UAV reaches the destination
$r_{\mathrm{collision}}$	reward function regarding whether a collision occurs
r_{out}	reward function regarding whether the UAV is out of bounds
rm	experience data
RM	experience data set
$\mathrm{RM}_{\mathrm{Capacity}}$	capacity of the replay memory
$[u_\gamma, u_\chi]$	control variable of the agent
$[x_{\mathrm{image}}, y_{\mathrm{image}}]$	size of the image
$[x_{\mathrm{obs}},\ y_{\mathrm{obs}},\ z_{\mathrm{obs}}]$	coordinates of the obstacle
$[x_{\mathrm{oInImage}}, y_{\mathrm{oInImage}}]$	coordinates of the obstacle in the image
$[x'_o, y'_o]$	relative coordinates of the obstacle in the image
$[x_{i,1}, y_{i,1}]$	upper-left corner coordinates of the *i-th* bounding box
$[x_{i,2}, y_{i,2}]$	lower-right corner coordinates of the *i-th* bounding box
R_g	radius of the destination
R_{obs}	radius of the no-fly zone created by the obstacle
RE	result experience set
$\mathbf{obs}_{\mathrm{posImage}}$	position of the obstacle in the image
$\mathbf{obs}'_{\mathrm{pos}}$	relative position of the obstacle in the image
$\boldsymbol{s}$	input state of the agent
$\boldsymbol{s}'$	input state of the agent after optimization
$\boldsymbol{s}_{\mathrm{U}}$	UAV state in $\boldsymbol{s}$
$\boldsymbol{s}_{\mathrm{g}}$	destination state in $\boldsymbol{s}$
$\boldsymbol{s}_{\mathrm{o}}$	image information in $\boldsymbol{s}$
$\boldsymbol{s}'_{\mathrm{o}}$	obstacle information obtained from $\boldsymbol{s}_o$
SE	safety experience set
$\boldsymbol{U}'_{\mathrm{pos}}$	relative coordinates of the UAV in the image
v	speed of the UAV
V	norm of the UAV speed
z_{u} -	difference between the UAV flight altitude and the minimum flight altitude
$\gamma(t)$	flight-path angle of the UAV
$\chi(t)$	heading angle of the UAV
χ'	relative value of the UAV heading angle in the image
τ_1	scale of the image
ε	exploration rate
$\varepsilon_{\mathrm{reset}}$	reset value of the exploration rate
$\theta_{\mathrm{g}XOY}$	destination-UAV lead angle in the XOY plane
θ'_{o}	relative value of the obstacle-UAV lead angle in the image
θ_{obs}	turning angle of the moving obstacle
$[x_{\mathrm{u}},\ y_{\mathrm{u}}\ , z_{\mathrm{u}}]$	coordinates of the UAV
$[x_{\mathrm{g}}\ , y_{\mathrm{g}}\ , z_{\mathrm{g}}]$	coordinates of the destination
$[e_{\mathrm{o}}, e_{\mathrm{c}}, e_{\mathrm{out}}, e_{\mathrm{g}}]$	mission situation parameter
$[p_{\mathrm{SE}}, p_{\mathrm{DE}}, p_{\mathrm{RE}}]$	data deposit rate
$[\mathrm{RE}_{\mathrm{arrival}}, \mathrm{RE}_{\mathrm{collision}}, \mathrm{RE}_{\mathrm{out}}]$	some categories of the result experience
$[s_{\mathrm{safe}}, s_{\mathrm{obs}}, s_{\mathrm{collision}}, s_{\mathrm{out}}, s_{\mathrm{arrival}}]$	some categories of status s

devices (e.g., on-board cameras), with their small sizes and light weights, are now widely used in UAVs, especially reconnaissance UAVs and reconnaissance-and-strike-integrated UAVs. Therefore, a UAV should have the Autonomous Navigation and Collision Avoidance (ANCA) capability through images captured by its on-board camera.

In order to solve the ANCA problem, many studies use traditional methods, which include analytical geometry-based methods, search-based methods, sample-based methods, etc. Analytical geometry-based methods, such as Dubins path,[8,9] Clothoid path,[10] and PH curve,[11] are difficult to directly find paths that satisfy both obstacle avoidance and curvature continuity when the environment contains multiple obstacles.[12–14] Search-based methods, such as A^* algorithm,[15] Dijkstra algorithm,[16] ant colony algorithm,[17] particle swarm optimization algorithm,[18] and genetic algorithm,[19] requires creation of a rasterized map before planning and further optimization of results after planning.[20–23] Sample-based methods, such as RRT algorithm,[24] require frequent re-planning when the environment changes, which will incur significant computational costs. Therefore, in unknown dynamic environments, this type of method is less reliable and more computationally intensive.

To overcome the limitations of the above methods, many researchers have used the deep reinforcement learning method for the ANCA problem in recent years. Based on the Deep Q-Network (DQN) method, in Refs. 25–27, ANCA was achieved for unmanned vehicles and robots, respectively, by distance-measuring sensors. Ref. 28 proposed an improved

DQN method for performing mobile robot ANCA tasks based on LiDAR information, which achieved better control and higher learning efficiency in a simulation environment compared to those of DQN and Dueling DQN. Although ANCA was achieved in Refs. 25–28, their scenarios were too simple and unable to perform image-based navigation. In Refs. 29, 30, the DQN method was combined with a Convolutional Neural Network (CNN) to implement image-based ANCA for a robot and a unmanned surface vehicle in a simulation environment, respectively. Ref. 31 proposed a dueling architecture-based double DQN to implement image-based ANCA for a mobile robot in a simple simulation environment by a Kinect sensor. Although Refs. 29–31 have implemented image-based ANCA, their simulation environments were all small in size and too simple, where the simulation environments were all solid-colored backgrounds and the obstacles were replaced by simple geometric shapes in solid colors. In summary, the above-mentioned studies based on the DQN have two problems: (A) unable to implement image-based navigation or only able to implement simple-image-based navigation; (B) environments are too simple, and the convergence speed and effect of the DQN will be affected when environments are more complex.

In order to obtain a stronger capability for image-based navigation, CNN-based target detection algorithms can be merged into the DQN method. In Ref. 32, quadrotor autonomous navigation was performed based on images taken by an on-board camera. In this study, the human in an image was detected by the YOLO, the detection result was translated into the coordinates of the center of the human body, and then, the quadrotor was guided by this coordinate for autonomous navigation. In Ref. 33, lawn mower autonomous navigation was implemented by the DQN method combined with the YOLO. In this case, a specified marker was recognized by the YOLO from an image taken by a camera, and then the recognition result was input directly into the DQN for autonomous navigation. In Refs. 34, 35, the DQN method was combined with the Faster R-CNN model to achieve autonomous obstacle avoidance for cars in different scenarios. However, in the above method, processing of the recognition results of the target detection algorithm is too simple to be applicable to the dynamic and unknown UAV mission scenarios in this paper.

In order to solve the problems of poor convergence and reduced convergence speed caused by complex environments, the probability that the experience in the replay memory is sampled should be optimized. In Ref. 36, a correlation coefficient calculation function was designed based on an agent's observed data, and this value was calculated for each experience. At training time, those experiences that had the smallest difference in the correlation coefficient with the experience at the current time step were selected and used to optimize the network. In Ref. 37, the error, i.e., the difference between the output and estimated values of the action was firstly calculated for each experience, and then the probability of being sampled was set for each experience based on the error value. Refs. 38, 39 both devised their own methods to calculate the experience priority coefficient and calculate this value for each experience. In sampling, the higher the priority coefficient, the higher the probability of being sampled. However, in the dynamic and unknown UAV mission scenarios studied in this paper, the above methods have two drawbacks: (A) a new parameter needs to be calculated for each experience, which will bring a lot of extra computation and reduce the training speed; (B) each experience is stored in the replay memory, which will occupy a large amount of computational memory.

Therefore, in order to solve the above problems, this paper proposes an improved deep reinforcement learning algorithm, the DQN method with the Faster R-CNN model and the Data Deposit Mechanism (FRDDM-DQN). In the FRDDM-DQN, images are firstly processed by the Faster R-CNN model, and obstacle information is extracted from them. After combining the processed obstacle information with the location information, it is fed back to the agent. The agent outputs an action based on the current state information according to the action selection policy. The UAV performs this action and receives a reward from the environment. Next, after filtering the experience according to the proposed data deposit mechanism, the experience to be saved is deposited into the replay memory, where the experience consists of a state, an action, and a reward. In order to reduce the training time consumption, the FRDDM-DQN is trained in the training environment based on the simulated value of the Faster R-CNN module output, and the agent is optimized by randomly sampling experiences from the replay memory until the agent converges to the optimal state. Finally, in order to test the performance of the FRDDM-DQN, it is verified in a 3D simulation environment through a series of experiments, including training experiments, test experiments, and typical episodes experiments.

In summary, the major contributions of this paper are as follows.

(1) We optimize the output of the Faster R-CNN model according to the characteristics of a scenario and the kinematic properties of a UAV, which reduces the training difficulty of the agent decision network.
(2) We propose a data deposit mechanism to adjust the proportion of various types of experience in the replay memory, which accelerates the convergence of the proposed method and obtains a better control effect without adding an additional computational effort.
(3) We use a two-part training approach, i.e., divide the training into training of the Faster R-CNN model and training of the FRDDM-DQN based on simulated values. This training approach reduces the time consumption for training and retraining when obstacles change.

This paper is organized as follows. In Section 2, the ANCA problem is introduced in detail and defined as a reinforcement learning problem. Additionally, the UAV kinematic model is defined. In Section 3, the FRDDM-DQN method is presented. Finally, before conclusions are made in Section 5, Section 4 describes results of the training and tests the trained agent in a test environment.

2. Problem formulation

In this section, firstly, the UAV ANCA problem studied in this paper is defined, and the UAV kinematic model, which is the control object in this problem, is given. Since the FRDDM-DQN proposed in this paper is an improved deep reinforcement learn-

ing algorithm, its essence is still a reinforcement learning algorithm. Therefore, the relevant elements in the reinforcement learning algorithm are defined in Section 2.3.

2.1. UAV ANCA problem

In actual military applications, UAVs are often used to perform low-altitude missions such as reconnaissance, search and rescue. In this paper, the mission performed by a UAV is defined as navigating to a pre-specified destination as soon as possible without a collision. During the mission, the environment may contain multiple static or moving obstacles that affect the flight of the UAV, such as military facilities on the ground or at sea, ships, missile launching vehicles, etc. This type of obstacle will create a hemispherical no-fly zone. When the UAV enters a no-fly zone created by an obstacle, it is considered that the UAV collides with the obstacle. In order to ensure navigation safety, the UAV needs to adjust its flight direction in time to avoid collisions. In addition, considering the complexity of military scenarios, this paper assumes that the radar systems of both the enemy and ours do not work. At this time, the UAV can only navigate based on images captured by its on-board camera, and the influence ranges of obstacles will be reduced (i.e., the size of a no-fly zone is decreased).

As shown in Fig. 1, the starting point and destination of the UAV and the positions of obstacles are randomly generated, and the UAV needs to find an obstacle-free path in an environment with multiple obstacles. Among them, the yellow star represents the destination, the red areas are the no-fly zones created by obstacles, and the green rectangular frame is the field of view of the UAV's on-board camera. When approaching obstacles, after considering both the obstacles and the destination location, the UAV should decide how to safely pass through a dangerous area, i.e., whether to change its heading or flight-path angle. After that, the UAV is required to continue performing its tasks. When the UAV collides with an obstacle or reaches its destination, the mission restarts.

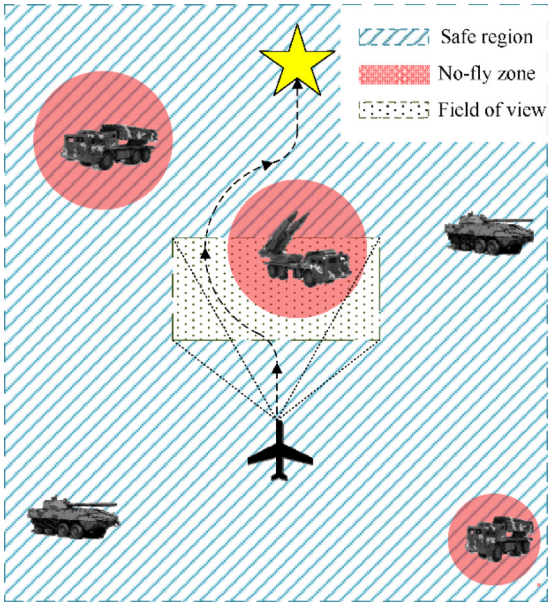


Fig. 1 Diagram of the mission scenario.

2.2. UAV kinematic model

In this paper, it is assumed that the UAV can achieve accurate attitude control as well as speed maintenance, and the influence of wind disturbances is ignored. To simplify the ANCA problem, the UAV is modeled as a mass point flying at a fixed speed in a three-dimensional space.

As illustrated in Fig. 2, $\boldsymbol{P}_{\mathrm{u}} = [x_{\mathrm{u}}(t), y_{\mathrm{u}}(t), z_{\mathrm{u}}(t)]^{\mathrm{T}}$ represents the position of the UAV; $v(t)$, $\gamma(t)$ and $\chi(t)$ represent the speed, flight-path angle, and heading angle of the UAV, respectively; $\boldsymbol{P}_{\mathrm{g}} = [x_{\mathrm{g}}(t), y_{\mathrm{g}}(t), z_{\mathrm{g}}(t)]^{\mathrm{T}}$ represents the position of the destination; the gray dashed frame represents the field of view of the UAV's on-board camera; $\boldsymbol{P}'_{\mathrm{obs}}$ represents the position of the obstacle in the field of view; $\boldsymbol{P}_{\mathrm{obs}} = [x_{\mathrm{obs}}(t), y_{\mathrm{obs}}(t), z_{\mathrm{obs}}(t)]^{\mathrm{T}}$ represents the true position of the obstacle, which is not available to the UAV and is only used for collision detection; D_{g} represents the distance between the UAV and the destination in the 3D space, and D_{g2d} represents that distance in a horizontal plane.

$$D_{\mathrm{g}}(t) = \sqrt{\left(x_{\mathrm{u}}(t) - x_{\mathrm{g}}(t)\right)^2 + \left(y_{\mathrm{u}}(t) - y_{\mathrm{g}}(t)\right)^2 + \left(z_{\mathrm{u}}(t) - z_{\mathrm{g}}(t)\right)^2} \tag{1}$$

$$D_{\mathrm{g2d}}(t) = \sqrt{\left(\left(x_{\mathrm{u}}(t) - x_{\mathrm{g}}(t)\right)^2 + \left(y_{\mathrm{u}}(t) - y_{\mathrm{g}}(t)\right)^2\right)} \tag{2}$$

D_{obs} represents the true distance between the UAV and the obstacle in the space, that is,

$$D_{\mathrm{obs}}(t) = \sqrt{(x_{\mathrm{u}}(t) - x_{\mathrm{obs}}(t))^2 + (y_{\mathrm{u}}(t) - y_{\mathrm{obs}}(t))^2 + (z_{\mathrm{u}}(t) - z_{\mathrm{obs}}(t))^2} \tag{3}$$

D_{obs} is mainly used for collision detection. Same as $\boldsymbol{P}_{\mathrm{obs}}$, D_{obs} is not available to the UAV. R_{g} represents the radius of the destination, and R_{obs} represents the radius of the no-fly zone created by the obstacle. When the distance D_{obs} between the UAV and the obstacle is less than R_{obs}, it is considered that the UAV has collided with the obstacle. Similarly, when the distance D_{g} between the UAV and the destination is less than R_{g}, it is considered that the UAV has reached the destination. In addition, $\theta_{\mathrm{g\_}XOY}$ is the destination-UAV lead angle in the XOY plane, that is the angle between the UAV's heading angle and the projection of the UAV-destination line of sight in the horizontal plane, which is calculated as follows:

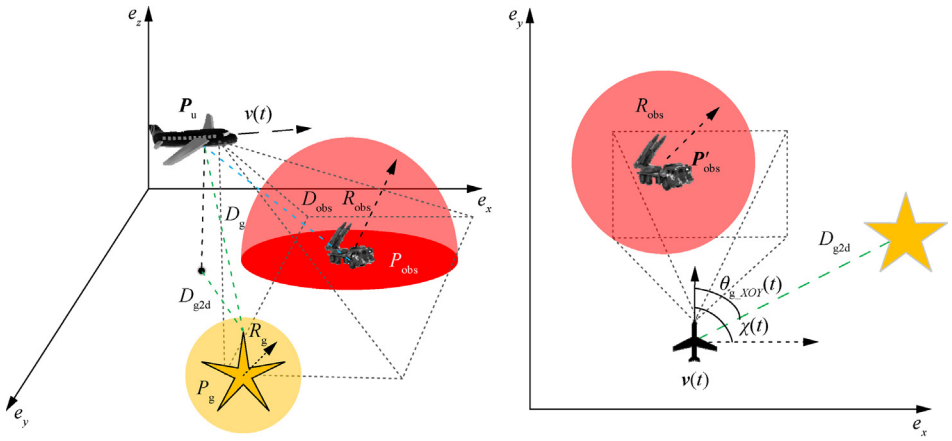
$$\theta_{\mathrm{g\_}XOY}(t) = \arctan\frac{y_{\mathrm{g}}(t) - y_{\mathrm{u}}(t)}{x_{\mathrm{g}}(t) - x_{\mathrm{u}}(t)} - \chi(t) \tag{4}$$

To get closer to the reality, in this paper, we assume that the UAV is flying in the 3D space with a constant speed $V = \| \boldsymbol{v}(t) \|$. The control model can be described with the following equations:

$$\begin{cases} \dot{x_{\mathrm{u}}}(t) = V\cos\gamma(t)\cos\chi(t) \\ \dot{y_{\mathrm{u}}}(t) = V\cos\gamma(t)\sin\chi(t) \\ \dot{z_{\mathrm{u}}}(t) = -V\sin\gamma(t) \\ \dot{\gamma}(t) = u_{\gamma} \\ \dot{\chi}(t) = u_{\chi} \end{cases} \tag{5}$$

where u_{γ} and u_{χ} are the control variables of the agent, that is,

$$\boldsymbol{a}_{\mathrm{U}} = [u_{\gamma}, u_{\chi}] \tag{6}$$



(a) 3D diagram (b) 2D diagram

Fig. 2 Motion model of the UAV, an obstacle, and the destination.

2.3. Reinforcement learning for the UAV ANCA problem

This section defines the state, action, and reward function in reinforcement learning, and then models the environment in reinforcement learning, that is, the UAV ANCA problem is modeled as a Markov decision process in Section 2.3.3.

2.3.1. State and action

In order to solve the UAV ANCA problem better, it is necessary to define the input state s(t) and the output action $\boldsymbol{a}_{\mathrm{U}}(t)$, where t represents time. During a mission, the state $\boldsymbol{s}(t)$ obtainable by the UAV consists of the UAV state $\boldsymbol{s}_{\mathrm{U}}$, the destination state $\boldsymbol{s}_{\mathrm{g}}$, and the image information $\boldsymbol{s}_{\mathrm{o}}$. The UAV state $\boldsymbol{s}_{\mathrm{U}}$ is obtained by GPS and gyroscope devices, where $\boldsymbol{s}_{\mathrm{U}} = [x_{\mathrm{u}}(t)\ ,\ y_{\mathrm{u}}(t), z_{\mathrm{u}}(t), V, \gamma(t), \chi(t)]$; the destination state $\boldsymbol{s}_{\mathrm{g}}$ is pre-specified before the mission or specified in real time in the UAV communicable state, where $\boldsymbol{s}_{\mathrm{g}} = [x_{\mathrm{g}}(t)\ ,\ y_{\mathrm{g}}(t), z_{\mathrm{g}}(t)]$; $\boldsymbol{s}_{\mathrm{o}}$ is the image captured by the on-board camera, which is mainly used to guide the UAV for collision avoidance. Among them, the UAV state $\boldsymbol{s}_{\mathrm{U}}$ and the destination state $\boldsymbol{s}_{\mathrm{g}}$ are values in the inertial coordinate system. In order to make the method more widely applicable, especially that when a scenario changes, the method is still available, the input state $\boldsymbol{s}(t)$ is defined as

$$\boldsymbol{s}(t) = \left[z_{\mathrm{u}\text{-}}(t), H_{\mathrm{UtoG}}(t), D_{\mathrm{g}}(t), \theta_{\mathrm{g}XOY}(t), \chi(t), \boldsymbol{s}_{\mathrm{o}}(t)\right] \tag{7}$$

where $z_{\mathrm{u}\text{-}}(t)$ is the difference between the current flight altitude and the set minimum flight altitude; $H_{\mathrm{UtoG}}(t)$ is the altitude difference between the UAV and the destination; $\theta_{\mathrm{g}\boldsymbol{XOY}}$ and $\left[H_{\mathrm{UtoG}}(t), D_{\mathrm{g}}(t)\right]$ are used to represent the relative positions of the UAV and the destination in the XOY and XOZ planes, respectively. Based on the above input $\boldsymbol{s}(t)$, the action $\boldsymbol{a}_{\mathrm{U}}(t) = \left[u_{\gamma}, u_{\boldsymbol{\chi}}\right]$, where u_{γ} and $u_{\boldsymbol{\chi}}$ represent the control inputs of the UAV's flight-path and heading angles, respectively.

2.3.2. Reward function

In the process of the UAV performing its mission, the agent outputs an action $\boldsymbol{a}_{\mathrm{U}}$ according to the current state $\boldsymbol{s}(t)$, and gets a new state $\boldsymbol{s}(t+1)$ and feedback from the environment after performing the action $\boldsymbol{a}_{\mathrm{U}}$. The output of the reward function r_{U} is the feedback from the environment, and it is also the standard by which the agent evaluates its current action $\boldsymbol{a}_{\mathrm{U}}$. In order to speed up the convergence of the agent, reward clipping[40] is used, i.e., the reward value is set to $+1$ or -1. In the mission scenario of this paper, it is expected that the UAV can navigate to its destination autonomously. Therefore, the reward function regarding whether the UAV reaches the destination is set to

$$r_{\mathrm{arrived}} = \begin{cases} +1, & \text{if UAV hasarrived} \\ 0, & \text{others} \end{cases} \tag{8}$$

To avoid collisions, the reward function regarding whether a collision occurs is set to

$$r_{\mathrm{collision}} = \begin{cases} -1, & \text{if UAV enter the no} - \text{flyz one} \\ 0, & \text{others} \end{cases} \tag{9}$$

In order to meet the mission requirements and improve the training efficiency, a navigation range is defined. The reward function regarding whether the UAV is out of bounds is set to

$$r_{\mathrm{out}} = \begin{cases} -1, \text{ifUAVisoutofbounds} \\ 0, \text{others} \end{cases} \tag{10}$$

Therefore, the reward function is set as follows:

$$r_{\mathrm{U}}(\boldsymbol{s}(t), \boldsymbol{a}_{\mathrm{U}}) = r_{\mathrm{arrived}} + r_{\mathrm{collision}} + r_{\mathrm{out}} \tag{11}$$

2.3.3. Markov decision process

During the UAV mission, the agent selects an action $\boldsymbol{a}_{\mathrm{U}}(t)$ to be performed based on the state $\boldsymbol{s}(t)$ at the current time step t. After executing the action $\boldsymbol{a}_{\mathrm{U}}(t)$, the state $\boldsymbol{s}(t)$ shifts to $\boldsymbol{s}(t+1)$ and receives a reward $r_{\mathrm{U}}(t+1)$ from the environment. It can be considered that the process of the agent selecting actions has a Markov property, and this process can be called a Markov decision process.[41] This Markov decision process consists of a four-tuple, that is,

$$\begin{cases} \mathbf{MDP} = [\boldsymbol{S}_{\mathrm{U}}, \boldsymbol{A}_{\mathrm{U}}, P_{\mathrm{sa}}, R_{\mathrm{U}}] \\ \boldsymbol{S}_{\mathrm{U}} = [\boldsymbol{s}(1), \boldsymbol{s}(2), ..., \boldsymbol{s}(t), ...] \\ \boldsymbol{A}_{\mathrm{U}} = [\boldsymbol{a}_{\mathrm{U}}(1), \boldsymbol{a}_{\mathrm{U}}(2), ..., \boldsymbol{a}_{\mathrm{U}}(t), ...] \\ P_{\mathrm{sa}} = p(\boldsymbol{s}(t+1), R_{\mathrm{U}}(t+1) | \boldsymbol{s}(t), \boldsymbol{a}_{\mathrm{U}}(t)) \\ R_{\mathrm{U}}(t+1) = r_{\mathrm{U}}(\boldsymbol{s}(t), \boldsymbol{a}_{\mathrm{U}}(t)) \end{cases} \tag{12}$$

in which $\boldsymbol{S}_{\mathrm{U}}$ is the state space set, and $\boldsymbol{s}(t)$ represents the state of the t-th time step; $\boldsymbol{A}_{\mathrm{U}}$ is the action space set, and $\boldsymbol{a}_{\mathrm{U}}(t)$ represents the action executed at the t-th time step. P_{sa} is the state

transition probability, which represents the probability of transferring to a new state $\boldsymbol{s}(t+1)$ and receiving a reward $r_{\mathrm{U}}(t+1)$ after performing the action $\boldsymbol{a}_{\mathrm{U}}(t)$ in the current state $\boldsymbol{s}(t)$. R_{U} is the reward function, and $r_{\mathrm{U}}(t+1)$ is the reward obtained after performing the action $\boldsymbol{a}_{\mathrm{U}}(t)$ in state $\boldsymbol{s}(t)$.

3. FRDDM-DQN method

To solve the UAV ANCA problem defined in Section 2.1, an improved DQN method is proposed in this section. In the military scenario studied by this paper, the UAV can only achieve image-based autonomous navigation through the on-board camera. In order to obtain a strong image navigation capability, the Faster R-CNN model is introduced into the DQN method, that is, the DQN method with the Faster R-CNN model (FR-DQN). Furthermore, owing to the changes in scenario characteristics, the experience replay approach in the DQN no longer works well, which will result in the agent unable to obtain better training effects. Therefore, based on the FR-DQN, we propose an improved experience replay approach, the replay memory Data Deposit Mechanism (DDM), that is, the DQN method with the Faster R-CNN model and the Data Deposit Mechanism (FRDDM-DQN).

This section firstly introduces the DQN method which is the basis of the FRDDM-DQN, and presents the overall framework of the FRDDM-DQN in Section 3.2. Then, improvement on the DQN in this paper is presented, where Sections 3.3 and 3.4 introduce the FR-DQN and the DDM, respectively. Finally, the training procedure of the FRDDM-DQN and its pseudo-code are presented in Section 3.5.

3.1. DQN method

The DQN[40] algorithm is a deep reinforcement learning algorithm proposed by the DeepMind team that combines a Q-Learning algorithm, a deep neural network, and an experience replay mechanism to generate action-value functions within an infinite state space. In the DQN, the action- and target-value functions are fitted through the prediction network $Q(\boldsymbol{s},\boldsymbol{a},\theta)$ and the target network $\hat{Q}(\boldsymbol{s},\boldsymbol{a},\theta^{-})$ with the same structure. In order to quantify the fitting effects of $Q(\boldsymbol{s},\boldsymbol{a},\theta)$ and $\hat{Q}(\boldsymbol{s},\boldsymbol{a},\theta^{-})$, and to make sure that the fitting effects can be optimized, the loss function of the network is defined in Eq. (13). Furthermore, the gradient of parameter θ can be calculated in Eq. (14), and the gradient descent algorithm is used to optimize parameter θ. After extensive training, the optimal action-value function can be obtained.

$$L(\theta)=E\left[\left(r+\gamma\max_{\boldsymbol{a}_{\mathrm{U}}(t+1)}\hat{Q}\left(\boldsymbol{s}(t+1),\boldsymbol{a}_{\mathrm{U}}(t+1),\theta^{-}\right)-Q(\boldsymbol{s}(t),\boldsymbol{a}_{\mathrm{U}}(t),\theta)\right)^{2}\right] \tag{13}$$

$$\begin{aligned}\nabla_{\theta}L(\theta)=E\Big[\Big(&r+\gamma\max_{\boldsymbol{a}'}\hat{Q}\left(\boldsymbol{s}(t+1),\boldsymbol{a}_{\mathrm{U}}(t+1),\theta^{-}\right)\\&-Q(\boldsymbol{s}(t),\boldsymbol{a}_{\mathrm{U}}(t),\theta)\Big)\nabla_{\theta}Q(\boldsymbol{s}(t),\boldsymbol{a}_{\mathrm{U}}(t),\theta)\Big]\end{aligned} \tag{14}$$

When applying the DQN method to the scenario in this paper, there are two problems: the lack of image-based ANCA capability and the no longer applicable experience replay approach. Therefore, in the FRDDM-DQN proposed in this paper, the structure of the DQN method is changed.

3.2. Framework of the FRDDM-DQN method

The structure and data flow of the FRDDM-DQN method are shown in Fig. 3. Compared with the DQN, the changes in the FRDDM-DQN are: (A) the CNN module used to extract image information is replaced by the Faster R-CNN model with a higher target detection capability; (B) a data deposit mechanism is proposed for filtering the experiences deposited in the replay memory.

Specifically, the UAV collects information $\boldsymbol{s}(t)$ through the on-board sensors and feeds the image $\boldsymbol{s}_{\mathrm{o}}$ to the Faster R-CNN model. The Faster R-CNN model performs feature extraction on the image and outputs the obstacle information $\boldsymbol{s}'_{\mathrm{o}}$. After merging $\boldsymbol{s}'_{\mathrm{o}}$ with the location information, the state $\boldsymbol{s}'(t)$ is fed into the prediction network. Based on the status $\boldsymbol{s}'(t)$, the prediction network outputs the optimal action $\boldsymbol{a}_{\mathrm{U}}(t)$. After performing the action $\boldsymbol{a}_{\mathrm{U}}(t)$, the environment will feedback a new status $\boldsymbol{s}(t+1)$ and a corresponding reward $r_{\mathrm{U}}(\boldsymbol{s}(t),\boldsymbol{a}_{\mathrm{U}}(t))$. Next, determine whether the current experience $\mathbf{rm}=[\boldsymbol{s}(t),\boldsymbol{a}_{\mathrm{U}}(t),r_{\mathrm{U}}(\boldsymbol{s}(t),\boldsymbol{a}_{\mathrm{U}}(t)),\boldsymbol{s}(t+1)]$ is stored in the replay memory according to the data deposit mechanism, and if not, **rm** is discarded. After filtering, the proportion of various types of experience in the replay memory will change, so that all types of experience have an appropriate probability of being sampled. Since then, m sets of experience are randomly sampled from the replay memory and fed to the prediction network as well as the target network, and then an error is calculated based on the loss function. The prediction network is optimized by the gradient descent algorithm according to the error. After a lot of training and optimization, finally, an agent with an ANCA capacity is obtained.

3.3. Faster R-CNN model for the FR-DQN

In the mission scenario studied by this paper, the UAV can only achieve image-based ANCA by its on-board camera. In Ref. 30, image-based autonomous navigation was implemented by the DQN with the CNN. However, since the scenario of this paper is complex, it is difficult for the CNN to extract valid information from captured images, where valid information is used to guide the agent for obstacle avoidance. Instead of the CNN, this paper obtains obstacle information from images by object detection algorithms, such as the Faster R-CNN[42] and the YOLO.[43] Compared with the Faster R-CNN, the YOLO has a lower accuracy and a higher miss detection rate for smaller objects in an image.[44,45] Considering the small size of the obstacle to be recognized in the image of this paper, in this section, the Faster R-CNN with a high accuracy of object detection is introduced into the DQN, i.e., the FR-DQN. In order to reduce the training difficulty and obtain better training results, this section is to further optimize the output of the Faster R-CNN model according to the characteristics of images captured by the on-board camera and the kinematic properties of the UAV in this paper.

During the UAV mission, there are multiple possible orientations of obstacles at the same coordinates, which means that the Faster R-CNN model also has multiple possible outputs at this time. As shown in Fig. 4, when the obstacle is in the same location but different orientations, the Faster R-CNN model outputs different results (different red boxes). Because the DQN does not have a memory function, the agent can only

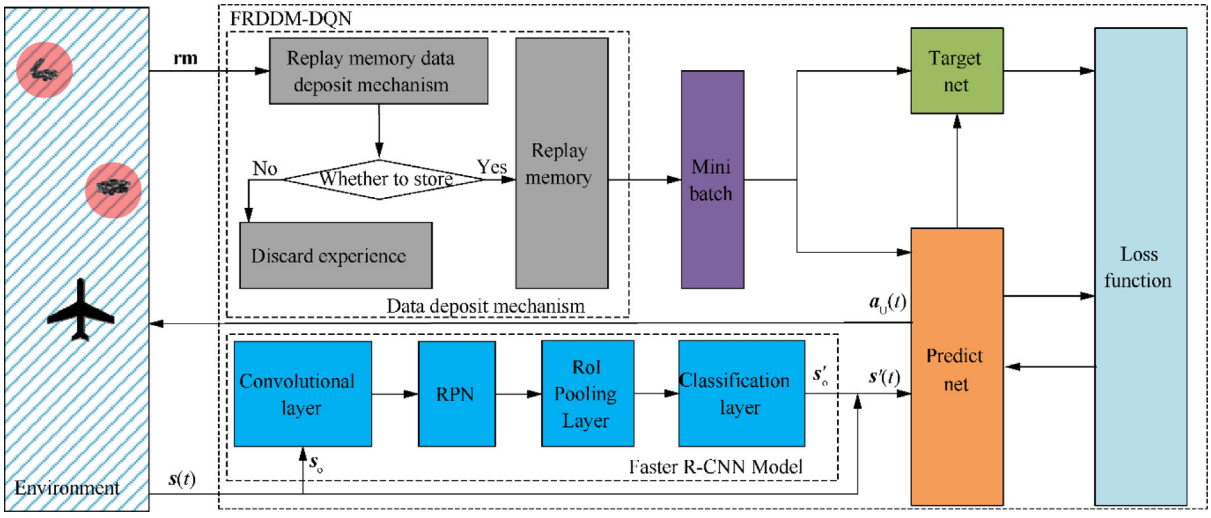


Fig. 3 Structure and data flow of the FRDDM-DQN.

make decisions based on the input of the current time step.[46] Therefore, when a moving obstacle is encountered, the agent will avoid the obstacle based on the current obstacle location, but cannot recognize that the obstacle is moving. This means that regardless of whether the encountered obstacle is moving or not, for the agent, the obstacle is always a static obstacle. In addition, the no-fly zone generated by the obstacle is hemispherical, and its range is only related to the location of the obstacle, not to the orientation. Therefore, based on the conclusions that the agent always treats the obstacle as a static obstacle and the range of the no-fly zone is not related to the orientation of the obstacle, it can be known that the agent will perform the same obstacle avoidance action as long as its coordinates are the same, regardless of its orientation. Therefore, in order to accelerate the convergence of the proposed method in this paper, the recognition result of the Faster R-CNN model is converted from bounding box coordinates to the center coordinates $\mathbf{obs}_{\text{posImage}}$ of the bounding box, that is,

$$\begin{aligned}\mathbf{obs}_{\text{posImage}} &= \left[x_{\text{oInImage}}, y_{\text{oInImage}}\right] \\ &= \left[x_{i,1} + \frac{x_{i,1} - x_{i,2}}{2}, y_{i,1} + \frac{y_{i,1} - y_{i,2}}{2}\right]\end{aligned} \tag{15}$$

where $\left[x_{\text{oInImage}}, y_{\text{oInImage}}\right]$ represents the center coordinates of the bounding box; i represents the $i-$th anchor in the image; $\left[\left(x_{i,1}, y_{i,1}\right), \left(x_{i,2}, y_{i,2}\right)\right]$ represents the bounding box coordinates of the $i-$th anchor which are output by the Faster R-CNN model; $\left(x_{i,1}, y_{i,1}\right)$ and $\left(x_{i,2}, y_{i,2}\right)$ represent the upper-left and lower-right coordinates of the bounding box, respectively.

$\mathbf{obs}_{\text{posImage}}$ is the coordinates of the obstacle in the image, which cannot be used directly for agent collision avoidance, and it is needed to convert to the relative coordinates obs'_{pos} between the obstacle and the UAV. As shown in Fig. 5, obs'_{pos} can be calculated by

$$\text{obs}'_{\text{pos}} = \left[x'_{\text{o}}, y'_{\text{o}}\right] = \left[\tau_1 x_{\text{oInImage}}, -1 \times \tau_1 \times y_{\text{oInImage}}\right] \tag{16}$$

where $\left[x'_{\text{o}}, y'_{\text{o}}\right]$ is the coordinates of obs'_{pos}; τ_1 is the scale of the captured image, that is, the ratio of the image size to the real size of the scene in the image. When the on-board camera with a zoom function is fixed to the UAV, it can be considered that τ_1 is a constant during the mission of the UAV, which is related to the installation position and orientation of the camera. In addition, in this scene, the relative position of the UAV to the field of view of the on-board camera is fixed. The relative coordinates of the UAV are

$$\boldsymbol{U}'_{\text{pos}} = \left[x'_{\text{U}}, y'_{\text{U}}\right] = \left[\tau_1 x_{\text{image}}/2, -\left(\tau_1 y_{\text{image}} + d_{\text{c}}\right)\right] \tag{17}$$

where $\left[x'_{\text{U}}, y'_{\text{U}}\right]$ is the coordinates of U'_{pos}; x_{image} and y_{image} are the sizes of the captured images; d_c is the distance between the UAV and the field of view of the camera. When the on-

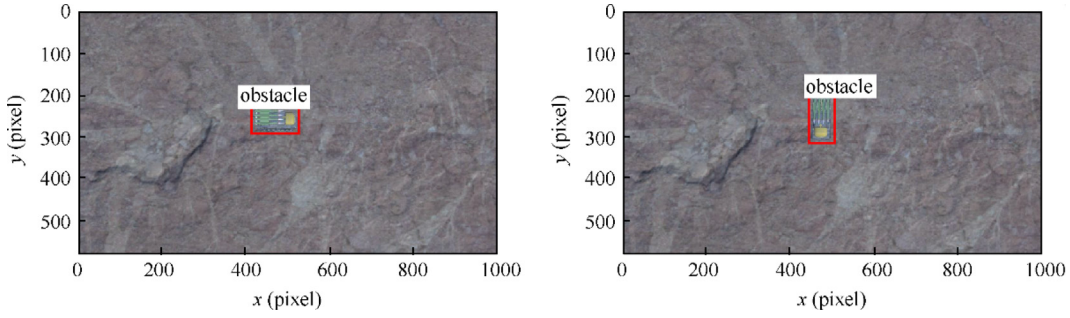


Fig. 4 Obstacle detection result of the Faster R-CNN model.

board camera is fixed to the UAV, x_{image}, y_{image}, and d_c are constants. According to Section 2.2, it is known that the agent controls the flight-path and heading angles of the UAV, and coordinates are not conducive to the convergence of the FRDDM-DQN. Therefore, through the relative coordinates between the UAV and the obstacle, the obstacle-UAV lead angle θ'_{o} and distance D'_{OtoU} can be calculated as follows:

$$D'_{\text{OtoU}} = \sqrt{\left(x'_{\text{U}} - x'_{\text{o}}\right)^2 + \left(y'_{\text{U}} - y'_{\text{o}}\right)^2} \tag{18}$$

$$\theta'_{\text{o}} = \arctan\frac{y'_{\text{U}} - y'_{\text{o}}}{x'_{\text{U}} - x'_{\text{o}}} - \chi' \tag{19}$$

where χ' represents the relative value of the UAV heading angle in the current scene. When the on-board camera is installed on the UAV, χ' is a constant. Finally, the recognition result of the Faster R-CNN model is transformed into $\boldsymbol{s}'_{\text{o}} = \left[D'_{\text{OtoU}}, \theta'_{\text{o}}\right]$. At this point, the input state of the agent changes from $\boldsymbol{s}(t)$ to

$$\boldsymbol{s}'(t) = \left[z_{\text{u-}}(t), H_{\text{UtoG}}(t), D_{\text{g}}(t), \theta_{\text{g\_XOY}}(t), \chi(t), D'_{\text{OtoU}}, \theta'_{\text{o}}\right] \tag{20}$$

In summary, replacing the CNN in the DQN with the Faster R-CNN model and converting the recognition result of the Faster R-CNN model can reduce the complexity of the experience data in the replay memory and facilitate the convergence of the FR-DQN. However, during the training process, it was found that the number of several types of experiences in the replay memory was unreasonable, which led to difficulty in convergence of agent. Therefore, this paper designs a new replay memory data deposit mechanism that can accelerate the convergence in Section 3.4.

3.4. Replay memory data deposit mechanism

In mission scenarios with a larger area similar to that of this paper, the experience replay approach in the DQN method no longer works well, which will prevent the agent from getting better training results. Therefore, based on the FR-DQN in Section 3.3, according to the characteristics of the scenario in this paper, a novel DDM is proposed to optimize the experience replay approach in the DQN method, i.e., the FRDDM-DQN method.

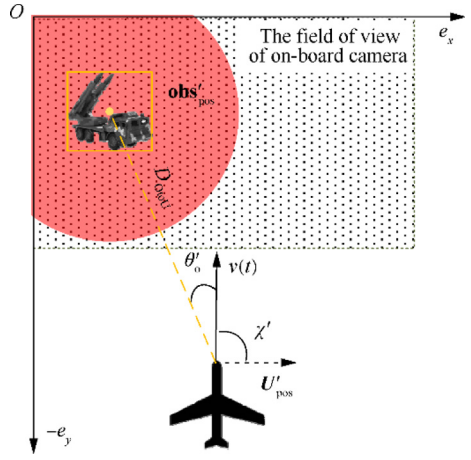


Fig. 5 Diagram of the relative position of the UAV to an obstacle.

There are three characteristics of the mission scenario studied in this paper: (A) the area of the scenario is relatively large; (B) the speed of the UAV is low compared to the area of the scenario; (C) the reward function only rewards or penalizes the last step of each episode, as shown in Section 2.3.2. These characteristics will lead to an imbalance in the share of various types of experiences in the replay memory. For example, the replay memory will store a large amount of intermediate state experience with a zero reward value and a small amount of result step experience with a non-zero reward value; however, the latter is more important for training the agent compared to the former. On the other hand, in the DQN method, the process of agent training is the process of continuously optimizing the agent's decision network with a set of randomly sampled experiences (i.e., minibatch) from the replay memory. In most cases, the size of the replay memory is much larger than that of the minibatch m (e.g., the size of the replay memory in this paper is 300000, and $m = 64$). In summary, due to the imbalance in the share of various types of experiences as well as the small amount of experience sampled at a time for training the agent, it is difficult for the agent to obtain better training results at this time by the experience replay approach in the DQN method.

Therefore, we propose a replay memory DDM with a less computation effort to vary the probability of each type of experience being sampled. This DDM classifies experiences and assigns a deposit ratio to each type of experience. With this deposit ratio, the share of each type of experience in the replay memory is rebalanced so that all types of experiences have a chance to be sampled by the minibatch.

In the UAV ANCA problem, the experience **RM** deposited in the replay memory consists of the performed action $\boldsymbol{a}_{\text{U}}$ and states $\boldsymbol{s}^i(t)$, i.e.,

$$\mathbf{RM} = \left\{\mathbf{rm}(i)\middle|\mathbf{rm}(i) = \left[\boldsymbol{s}^i(a-), \boldsymbol{a}_{\text{U}}, \boldsymbol{s}^i(a+), r_{\text{U}}\right], i < \mathbf{RM}_{\text{Capacity}}\right\} \tag{21}$$

where i is the serial number of the experience data; $a-$ represents the time step before performing the action $\boldsymbol{a}_{\text{U}}$; $a+$ represents the time step after performing the action $\boldsymbol{a}_{\text{U}}$; $\mathbf{RM}_{\text{Capacity}}$ is the capacity of the replay memory. In this case, the state $\boldsymbol{s}^i(a+)$ after performing the action in the i-th experience $\boldsymbol{rm}(i)$ is also the state $\boldsymbol{s}^{i+1}(a-)$ before performing the action in the $(i+1)$-th experience $\mathbf{rm}(i+1)$.

Based on the analysis of the state data, the mission situation represented by $\boldsymbol{s}^i(t)$ is defined as

$$\left[\!\left[\boldsymbol{s}^i(t)\right]\!\right] = \left[e_{\text{o}}, e_{\text{c}}, e_{\text{out}}, e_{\text{g}}\right] \tag{22}$$

where $[\![\cdot]\!]$ is defined as the mission situation represented by the state; $e_{\text{o}}, e_{\text{c}}, e_{\text{out}}, e_{\text{g}}$ are the parameters used to define the mission situation: e_{o} indicates whether the agent has detected obstacles, e_{c} indicates whether the agent has collided, e_{out} indicates whether the agent is out of bounds, and e_{g} indicates whether the agent has reached the destination. The values and definitions of $e_{\text{o}}, e_{\text{c}}, e_{\text{out}}, e_{\text{g}}$ are listed in Table 1.

According to the above mission situation, the state $\boldsymbol{s}^i(t)$ obtained by the agent can be divided into the following categories: state $\boldsymbol{s}_{\text{safe}}$ in which the UAV does not detect obstacles and does not collide, go out of bounds, or reach the destina-

tion; state $\boldsymbol{s}_{\text{obs}}$ in which the UAV detects obstacles and does not collide, go out of bounds, or reach the destination; state $\boldsymbol{s}_{\text{collision}}$ in which the UAV collides with obstacles; state $\boldsymbol{s}_{\text{out}}$ in which the UAV goes out of bounds; state $\boldsymbol{s}_{\text{arrival}}$ in which the UAV reaches the destination. That is,

$$\boldsymbol{s}^i(t) \in \left\{\boldsymbol{s}_{\text{safe}}, \boldsymbol{s}_{\text{obs}}, \boldsymbol{s}_{\text{collision}}, \boldsymbol{s}_{\text{out}}, \boldsymbol{s}_{\text{arrival}}\right\} \tag{23}$$

$$\boldsymbol{s}_{\text{safe}} = \left\{\boldsymbol{s}^i(t) \middle| [\![\boldsymbol{s}^i(t)]\!] = \left[e_{\text{o}} = 0, e_{\text{c}} = 0, e_{\text{out}} = 0, e_{\text{g}} = 0\right]\right\} \tag{24}$$

$$\boldsymbol{s}_{\text{obs}} = \left\{\boldsymbol{s}^i(t) \middle| [\![\boldsymbol{s}^i(t)]\!] = \left[e_{\text{o}} = 1, e_{\text{c}} = 0, e_{\text{out}} = 0, e_{\text{g}} = 0\right]\right\} \tag{25}$$

$$\boldsymbol{s}_{\text{collision}} = \left\{\boldsymbol{s}^i(t) \middle| [\![\boldsymbol{s}^i(t)]\!] = \left[e_{\text{o}} \in \{0,1\}, e_{\text{c}} = 1, e_{\text{out}} = 0, e_{\text{g}} = 0\right]\right\} \tag{26}$$

$$\boldsymbol{s}_{\text{out}} = \left\{\boldsymbol{s}^i(t) \middle| [\![\boldsymbol{s}^i(t)]\!] = \left[e_{\text{o}} \in \{0,1\}, e_{\text{c}} = 0, e_{\text{out}} = 1, e_{\text{g}} = 0\right]\right\} \tag{27}$$

$$\boldsymbol{s}_{\text{arrival}} = \left\{\boldsymbol{s}^i(t) \middle| [\![\boldsymbol{s}^i(t)]\!] = \left[e_{\text{o}} \in \{0,1\}, e_{\text{c}} = 0, e_{\text{out}} = 0, e_{\text{g}} = 1\right]\right\} \tag{28}$$

Furthermore, according to the type of state $\boldsymbol{s}^i(t)$, classify the experiences $\mathbf{rm}(i) \in \mathbf{RM}$ in the replay memory as follows, where $\mathbf{rm}(i) = [\boldsymbol{s}^i(a-), \boldsymbol{a}_{\text{U}}, \boldsymbol{s}^i(a+), r_{\text{U}}]$.

(1) Result Experience (RE). This type of experience refers to an experience where the agent is in the last step of each episode. It can be further divided into arrival experience $\mathbf{RE}_{\text{arrival}}$, collision experience $\mathbf{RE}_{\text{collision}}$, and out-of-bounds experience $\mathbf{RE}_{\text{out}}$, i.e.,

$$\mathbf{RE} = \{\mathbf{RE}_{\text{arrival}}, \mathbf{RE}_{\text{collision}}, \mathbf{RE}_{\text{out}}\}, \mathbf{RE} \in \mathbf{RM} \tag{29}$$

$$\mathbf{RE}_{\text{arrival}} = \{\mathbf{rm}(i) | \{\boldsymbol{s}^i(a-) \in \boldsymbol{s}_{\text{safe}}, \boldsymbol{s}^i(a+) \in \boldsymbol{s}_{\text{arrival}}\} \cup \{\boldsymbol{s}^i(a-) \in \boldsymbol{s}_{\text{obs}}, \boldsymbol{s}^i(a+) \in \boldsymbol{s}_{\text{arrival}}\}\} \tag{30}$$

$$\mathbf{RE}_{\text{collision}} = \{\mathbf{rm}(i) | \{\boldsymbol{s}^i(a-) \in \boldsymbol{s}_{\text{safe}}, \boldsymbol{s}^i(a+) \in \boldsymbol{s}_{\text{collision}}\} \cup \{\boldsymbol{s}^i(a-) \in \boldsymbol{s}_{\text{obs}}, \boldsymbol{s}^i(a+) \in \boldsymbol{s}_{\text{collision}}\}\} \tag{31}$$

$$\mathbf{RE}_{\text{out}} = \{\mathbf{rm}(i) | \{\boldsymbol{s}^i(a-) \in \boldsymbol{s}_{\text{safe}}, \boldsymbol{s}^i(a+) \in \boldsymbol{s}_{\text{out}}\} \cup \{\boldsymbol{s}^i(a-) \in \boldsymbol{s}_{\text{obs}}, \boldsymbol{s}^i(a+) \in \boldsymbol{s}_{\text{out}}\}\} \tag{32}$$

All these three types of experiences have non-zero rewards, by which the agent can learn an effective strategy. Specifically, with a positive reward, the agent can learn that it should navigate to its destination; with a negative rewards, the agent can learn that it should avoid collisions and out-of-bounds. In addition, since the **RE** type of experience is generated only once at the end of each episode, its number is small compared to the capacity of the replay memory. Therefore, a higher deposit rate p_{RE} should be set for the **RE** type of experience.

(2) Danger Experience (DE). This type of experience refers to an experience that the agent has detected obstacles. At this point, it means that the UAV is close to an obstacle, and the agent needs to decide whether to avoid the obstacle and determine an obstacle avoidance action (when obstacle avoidance is required) based on the obstacle location and the destination location, which is

Table 1 System description parameters.

Parameters	Meanings of parameters	Meanings of different values	
		Yes	No
e_{o}	Obstacles detected or not	1	0
e_{c}	Collision or not	1	0
e_{out}	Out of bounds or not	1	0
e_{g}	Arrival at destination or not	1	0

$$\mathbf{DE} = \{\mathbf{rm}(i) | \{\boldsymbol{s}^i(a-) \in \boldsymbol{s}_{\text{obs}}, \boldsymbol{s}^i(a+) \in \boldsymbol{s}_{\text{safe}}\} \cup \{\boldsymbol{s}^i(a-) \in \boldsymbol{s}_{\text{safe}}, \boldsymbol{s}^i(a+) \in \boldsymbol{s}_{\text{obs}}\} \cup \{\boldsymbol{s}^i(a-) \in \boldsymbol{s}_{\text{obs}}, \boldsymbol{s}^i(a+) \in \boldsymbol{s}_{\text{obs}}\}\} \tag{33}$$

where $\{\mathbf{rm}(i) | \boldsymbol{s}^i(a-) \in \boldsymbol{s}_{\text{obs}}, \boldsymbol{s}^i(a+) \in \boldsymbol{s}_{\text{safe}}\}$ is the experience of detecting obstacles before performing the action and not detecting obstacles after performing the action, $\{\mathbf{rm}(i) | \boldsymbol{s}^i(a-) \in \boldsymbol{s}_{\text{safe}}, \boldsymbol{s}^i(a+) \in \boldsymbol{s}_{\text{obs}}\}$ is the experience of not detecting obstacles before performing the action and detecting obstacles after performing the action, and is the experience of detecting obstacles both before and after performing the action.

Both the **DE** and $\text{RE}_{\text{collision}}$ types of experience are related to obstacles, but the roles of the two are different. Through the $\text{RE}_{\text{collision}}$ type of experience, the agent can learn not to collide with an obstacle when navigating near it, while through the DE type of experience, the agent can learn how to avoid collision. During the training, the DE type of experience is generated when the UAV is close to an obstacle. As the number of training steps increases, the agent gradually learns to avoid obstacles, and the cases in which the UAV approaches obstacles (i.e., navigates between obstacles) increase. At this time, the number of DE experiences in the replay memory may be higher. Therefore, a lower deposit ratio p_{DE} should be set for the DE type of experience.

(3) Safety Experience (SE). This type of experience is the intermediate state in which the UAV navigates toward its destination and away from obstacles, i.e.,

$$\mathbf{SE} = \left\{\mathbf{rm}(i) \middle| \left\{\boldsymbol{s}^i(a-) \in \boldsymbol{s}_{\text{safe}}, \boldsymbol{s}^i(a+) \in \boldsymbol{s}_{\text{safe}}\right\}\right\} \tag{34}$$

In order to get a positive reward at the end of each episode, this type of experience helps the agent learn how to navigate the UAV to its destination. Due to the large mission scenario considered in this paper, the number of SE experiences is high. Therefore, a smaller deposit ratio p_{SE} should be set for the SE type of experience.

During the training, as described above, different deposit ratios are set for each type of experience, and only a portion of each type of experience is deposited into the replay memory.

The quantity relationship between the number of each type of experience in the new replay memory after adjustment is

$$|\mathbf{RM}'| = p_{\mathrm{RE}} \times |\mathbf{RE}| + p_{\mathrm{DE}} \times |\mathbf{DE}| + p_{\mathrm{SE}} \times |\mathbf{SE}| \tag{35}$$

where $|\cdot|$ indicates the number of the specified type of experience in the replay memory, and $\mathbf{RM}'$ represents the set of all experiences in the new replay memory. The specific operation of the data deposit mechanism is shown in Algorithm 1.

Algorithm 1. Replay memory data deposit mechanism method.

ExperienceStorage (**rm**)
Initialize experience storage flags , experience deposit ratio $p_{\mathrm{RE}}, p_{\mathrm{DE}}, p_{\mathrm{SE}}$
Generate a random value p', and $p' \in [0, 1]$
Type(**rm**) ← Determine the type of experience rm
if Type(**rm**) is **RE do**
  if $p' < p_{\mathrm{RE}}$ **do**
    $F_{\mathrm{storage}} \leftarrow$ true
  end if
else if Type(**rm**) **is DE do**
  if $p\prime < p_{\mathrm{DE}}$ **do**
    $F_{\mathrm{storage}} \leftarrow$ true
  end if
else if Type(**rm**) is **SE do**
  if $p\prime < p_{\mathrm{SE}}$ **do**
    $F_{\mathrm{storage}} \leftarrow$ true
  end if
end if
Return F_{storage}

3.5. FRDDM-DQN algorithm

In this section, the pseudo-code of the FRDDM-DQN (Algorithm 2) is proposed based on the replay memory data deposit mechanism in Section 3.4. Among them, the learning rate adjustment strategy and the exploration rate adjustment strategy are shown in Algorithm 3 and Algorithm 4, respectively.

Algorithm 2. The FRDDM-DQN method

Inputs: state $\boldsymbol{s}(t)$
Output: parameters of target network and prediction network θ
Initialize:
maximum number of total effective steps T_{t}, maximum number of steps per episode T_{e},
initial exploration rate ε_0, minimum exploration rate $\varepsilon_{\min}$,
reset value of exploration rate $\varepsilon_{\mathrm{reset}}$,
the piecewise learning rate $\alpha_1, \alpha_2, \alpha_3, \alpha_4$, the boundaries of piecewise learning rate n_1, n_2, n_3, n_4,
the size of the minibatch m, discount rate γ, network update frequency **C**;
data deposit ratio for various experiences $p_{\mathrm{RE}}, p_{\mathrm{DE}}, p_{\mathrm{SE}}$
Initialize: replay memory D,
the parameter θ of prediction network Q, the parameter θ^- of target network $\hat{Q}$
Initialize: exploration rate $\varepsilon \leftarrow \varepsilon_0$, learning rate $\alpha \leftarrow \alpha_1$, the number of total effective steps performed t_t, number of steps performed in one episode t_e
for $t_{\mathrm{t}} = 1, T_{\mathrm{t}}$ **do**
  Initialize the task and get state $\boldsymbol{s}(t)$
  while not reached destination and no collision and no out of bounds and $t_{\mathrm{e}} < T_{\mathrm{e}}$ **do**
    Select actions $\boldsymbol{a}_{\mathrm{U}}$ based on $\varepsilon-$ greedy strategy
    Perform action $\boldsymbol{a}_{\mathrm{U}}$ and get reward r_{U}, new state information $\boldsymbol{s}(t+1)$
    Generate experience $\mathbf{rm} = [\boldsymbol{s}(t), \boldsymbol{a}_{\mathrm{U}}, r_{\mathrm{U}}, \boldsymbol{s}(t+1)]$
    Determine whether the experience RM is deposited into the replay memory according to the data deposit mechanism $F_{\mathrm{storage}} \leftarrow$ ExperienceStorage (**rm**)
    if F_{storage} **do**
      $t_{\mathrm{t}} \leftarrow t_{\mathrm{t}} + 1$
      Store experience **rm** in the replay memory D
    else do
      discard experience data **rm**
    end if
    $\alpha \leftarrow$ LearningRateChange$(t_{\mathrm{t}}, \mathrm{isTraining})$
    $\varepsilon \leftarrow$ ExplorationRateChang$(\varepsilon, t_{\mathrm{t}}, F_{\mathrm{storage}}, \mathrm{isTraining})$
    if isTraining and F_{storage} **do**
      Select m samples from the replay memory D to the minibatch
      if reached destination and collision and out of bounds and $t_{\mathrm{e}} == T_{\mathrm{e}}$ **do**
        $y = r_{\mathrm{U}}$
      else do
        $y = r_{\mathrm{U}} + \gamma \max_{\boldsymbol{a}_{\mathrm{U}}(t+1)} \hat{Q}(\boldsymbol{s}(t+1), \boldsymbol{a}_{\mathrm{U}}(t+1), \theta^-)$
      Update the parameters θ of the prediction network Q in the loss function $L(\theta) = E(y - Q(\boldsymbol{s}(t), \boldsymbol{a}_{\mathrm{U}}(t), \theta))$ using the gradient descent algorithm
      Set the parameters θ^- of the target network $\theta^- = \theta$ every step C
      end if
    end if
    $\boldsymbol{s}(t) \leftarrow \boldsymbol{s}(t+1), t_{\mathrm{e}} \leftarrow t_{\mathrm{e}} + 1$
  end while
end for

Algorithm 3. Learning rate adjustment strategy

LearningRateChange $(t_{\mathrm{t}}, \mathrm{isTraining})$
Initialize learning rate $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ and boundaries of learning rate n_1, n_2, n_3, n_4
if isTraining **and** $t_{\mathrm{t}} \leqslant n_1$ **do**
  $\alpha \leftarrow \alpha_1$
else if isTraining **and** $n_1 < t_{\mathrm{t}} \leqslant n_2$ **do**
  $\alpha \leftarrow \alpha_2$
else if isTraining **and** $n_2 < t_{\mathrm{t}} \leqslant n_3$ **do**
  $\alpha \leftarrow \alpha_3$
else if isTraining **and** $n_3 < t_{\mathrm{t}} \leqslant n_4$ **do**
  $\alpha \leftarrow \alpha_4$
end if
Return α

Algorithm 4. Exploration rate adjustment strategy

ExplorationRateChang $(\varepsilon, t_t, F_{storage}, \text{isTraining})$
Initialize exploration rate reset cycle N
Initialize the minimum value of the exploration rate ε_{min}, reset value of exploration rate ε_{reset},
decay value of the exploration rate for a single step $p_\Delta = \frac{1}{N}$
if isTraining **and** $F_{storage}$ **and** $t_t < N$ **do**
  if $\varepsilon \geqslant \varepsilon_{min}$ **do**
    $\varepsilon \leftarrow \varepsilon - p_\Delta$
  else do
    $\varepsilon \leftarrow \varepsilon_{min}$
  end if
else if isTraining **and** $F_{storage}$ **and** $t_t \geqslant N$ **do**
  if $t_t \% N == 0$ **do**
    $\varepsilon \leftarrow \varepsilon_{reset}$
  else do
    if $\varepsilon \geqslant \varepsilon_{min}$ **do**
      $\varepsilon \leftarrow \varepsilon - p_\Delta$
    else do
      $\varepsilon \leftarrow \varepsilon_{min}$
    end if
  end if
end if
Return ε

4. Simulation results

To solve the UAV ANCA problem, the FRDDM-DQN method proposed in Section 3.5 is trained and verified in this section. We propose a two-part training approach, that is, divide the training into two parts: training of the Faster R-CNN model and training of the DDM-DQN based on the Simulated Value of the Faster R-CNN model Output (SVFO). Then, the agent trained by the FRDDM-DQN is tested in a designed Unity3D-based simulation environment.

4.1. Simulation environment

The simulation environment in this paper is divided into two parts: the training environment in python and the test environment in Unity3D. The training of the FRDDM-DQN based on the SVFO is performed in the training environment; the training of the Faster R-CNN model is performed using the obstacle images captured by the UAV in the test environment. Finally, the agent trained by the FRDDM-DQN is tested in the test environment.

In the mission scenario of this paper, the UAV's speed and maximum heading angular velocity are set to 42 m/s and 0.084 rad/s, respectively; the UAV's maximum flight-path angular velocity and maximum flight-path angle are set to 0.0023 rad/s and 0.14 rad, respectively. When the scenario contains moving obstacles, the moving obstacles are traveling at a constant speed of 8.3 m/s. The obstacles make a turn every 30 s, and the turning angle θ_{obs} is randomly generated, which is $\theta_{obs} \in [-\pi/2, \pi/2]$. For the small UAV used in this paper, with both radio and radar being turned off, the radius of the no-fly zone is set to 1.5 km, and the radius of the destination area is set to 2.0 km. In order to train the FRDDM-DQN based on the SVFO, a 15 km × 18 km training environment is designed using the matplotlib plotting library in python, as shown in Fig. 6, in which the green point, the blue rectangle, and the red circles represent the current position of the UAV, the field of view of the on-board camera, and the no-fly zones created by the obstacles at a 900-meter altitude, respectively.

Since Unity3D has the ability to detect collisions in real time as well as the ability to simulate on-board camera functions, in order to test the FRDDM-DQN proposed in this paper, we design a test environment in Unity3D. The test environment is set on land, which includes hills and plains. Fig. 7 shows snapshots of the environment. The plains are randomly distributed with various objects, such as missile launching vehicles (as shown in Fig. 7(e) and (f)) and tanks (as shown in Fig. 7(d)). In the test environment, the missile launching vehicles are considered as obstacles for the UAV to avoid because of their ground-to-air attack capability; tanks are not considered as obstacles because they do not have ground-to-air attack capability. Their positions and orientations are random, and they may be in a moving or static state.

4.2. Training experiments

Before testing and applying the FRDDM-DQN, it is necessary to train the FRDDM-DQN first through the two-part training approach. The training of the FRDDM-DQN is divided into two parts: training of the Faster R-CNN model and training of the DDM-DQN based on the SVFO. Training the DDM-DQN based on the SVFO can effectively reduce the training time, because the calculation of the SVFO is faster than that of the Faster R-CNN model output. In addition, when the UAV mission scenario is changed, this two-part training approach takes less time to retrain. For example, when an obstacle to be avoided by the UAV changes, only the Faster R-CNN model needs to be retrained, and the DDM-DQN based on the SVFO does not need to be retrained, while the training of the DDM-DQN based on the SVFO is the most time-consuming part of the training of the FRDDM-DQN.

4.2.1. Training the Faster R-CNN model to obtain obstacle information

In order to get the ability to recognize obstacles, the Faster R-CNN needs to be trained by images containing obstacles,

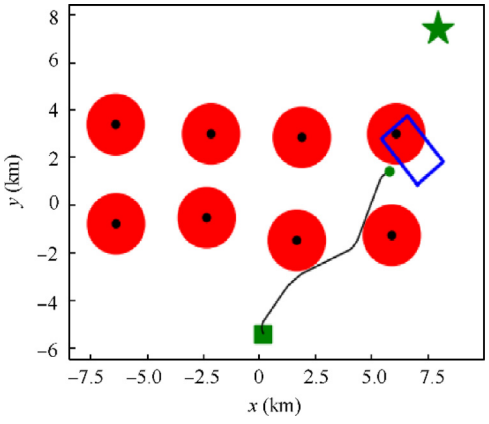


Fig. 6 Training environment at a 900 m height.

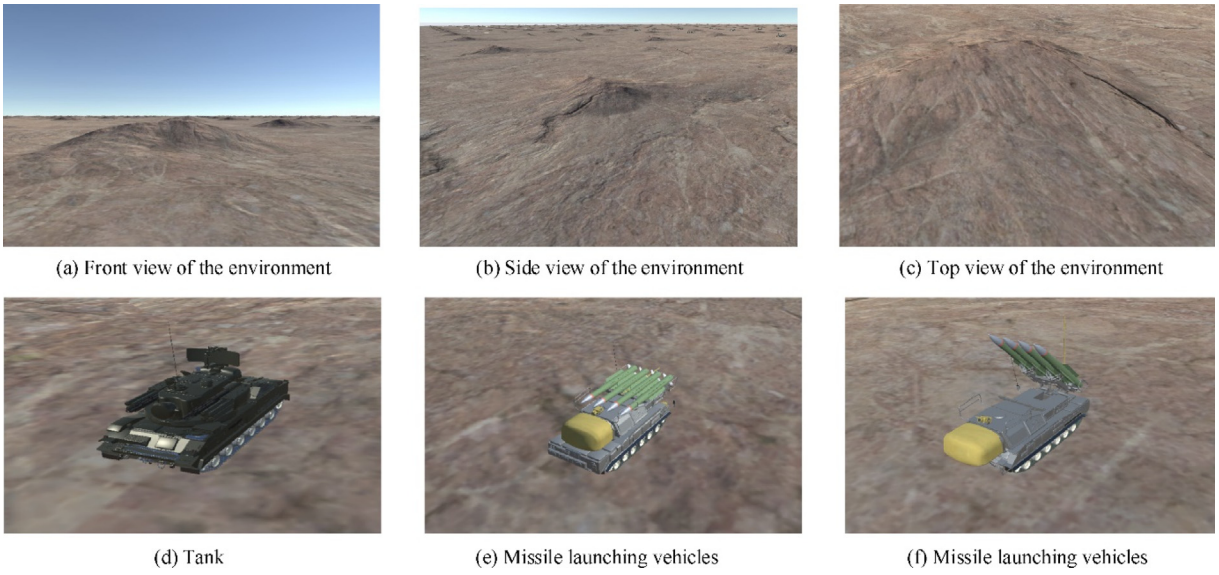


Fig. 7 Snapshots of the test environment in Unity3D.

where these images can be obtained through previous reconnaissance. In the military scenario, owing to the little amount of enemy information (obstacle images) available to the reconnaissance aircraft in advance, this paper assumes that only 5 images containing obstacles are acquired, and the Faster R-CNN model is trained by these 5 images. During the training, the initial learning rate is set to 0.001, the decay coefficient of the learning rate is set to 0.1, and the weight decay is set to 0.0005. In this paper, the Faster R-CNN is trained 20 epochs on the image dataset with a batch size of 1. After the training, the Faster R-CNN has been able to achieve the recognition of specified obstacles, and some of the results are shown in Fig. 8. After getting the coordinates of an obstacle in an image, the coordinates are converted to the obstacle state information s'_o according to Eqs. (16)–(20), where s'_o represents the relative position of the obstacle to the UAV.

As a comparison, another classical object detection algorithm, YOLO, is tested in this paper. Similarly, the YOLO is trained by images of the Faster R-CNN, in which the initial learning rate is set to 0.005, and the weight decay is set to 0.00005; the YOLO is trained with 750 epochs, and the batch size is 1. After the training, to completely simulate the military application scenario, a test is performed through the above-mentioned images used for training (as there is no more image available in the real scenario for testing). In order to show the training effects more clearly, the Faster R-CNN is tested along with the YOLO, and results are shown in Fig. 9 and Table 2.

During the mission, due to the small sizes of the obstacles in the images captured by the on-board camera, in order to display the recognition results more clearly, only the obstacles and their surrounding images are shown in Fig. 9. The test statistical results are shown in Table 2. In the test, the Faster R-CNN has recognized all the obstacles in the images with a 100% recognition rate, and the confidence coefficients are all greater than 0.9; the YOLO has successfully recognized 3 images with a 60% recognition rate, and the confidence coefficients of the successfully recognized images are all less than 0.5.

Therefore, it is concluded that the recognition capability of the Faster R-CNN is stronger than that of the YOLO in the current scenario where the objects to be recognized are small. In addition, further recognition capability tests are performed in test experiments.

4.2.2. Training of the FRDDM-DQN based on the SVFO

(1) Training settings

In order to reduced time consumption for training and retraining when obstacles change, the DDM-DQN is trained based on the SVFO in the training environment shown in Fig. 6. Training curves with multiple data deposit rate setting schemes and multiple learning rate setting schemes are shown respectively in Fig. 10, where the arrival rate corresponding to each episode is the arrival rate of the latest 100 episodes including the current episode. Note that in those figures, each point in the shaded areas indicates the statistical value of the arrival rate, and the solid lines in the shaded areas represent arrival rate curves after smoothing. In the DDM-DQN, the exploration rate ε is reset to ε_{reset} every N effective steps, as shown in Algorithm 4, so the arrival rate demonstrates periodic fluctuations. From Fig. 10, it can be seen that the piecewise constant decay learning rate and the data deposit rate $\boldsymbol{p} = [0.16, 0.05, 1]$ are optimal. Therefore, the hyperparameters of the network are set as in Table 3. To verify the effectiveness of the data deposit mechanism (proposed in Section 3.4) in the DDM-DQN, as a comparison, the DQN, DDQN, and Dueling DQN are trained based on the SVFO and with the same hyperparameters (except for the data deposit rate $\boldsymbol{p}$ and the maximum total effective steps T_t), where the network structure of the Dueling DQN is $(29 \times 512 \times 128 \times 6, 29 \times 512 \times 128 \times 1)$. To verify the effectiveness of the optimization of the Faster R-CNN output proposed in this paper, as a comparison, the DQN is trained based on the Simulated Values of the Original Faster R-CNN Output (SVOO) and with the same hyperparameters, and this method is denoted as O-DQN.

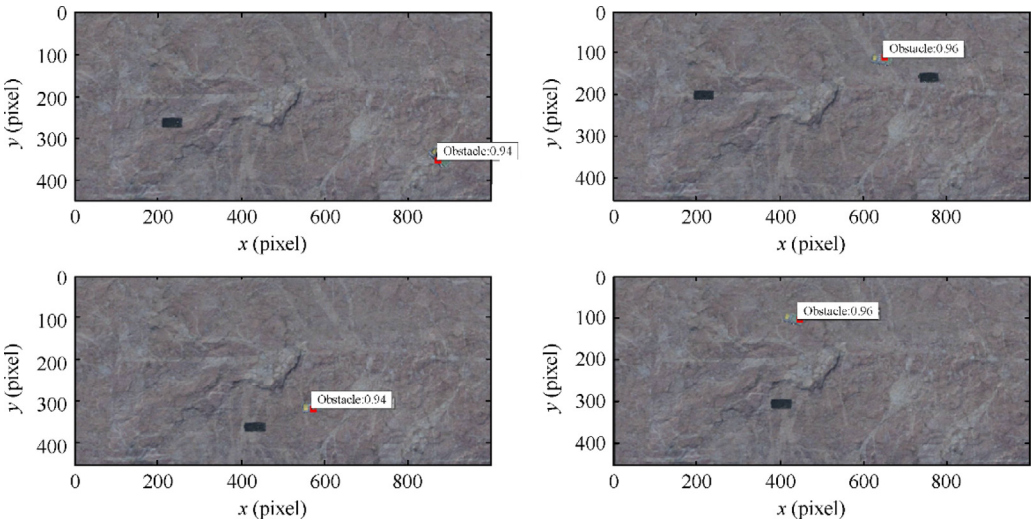


Fig. 8 Detection results of the Faster R-CNN model.

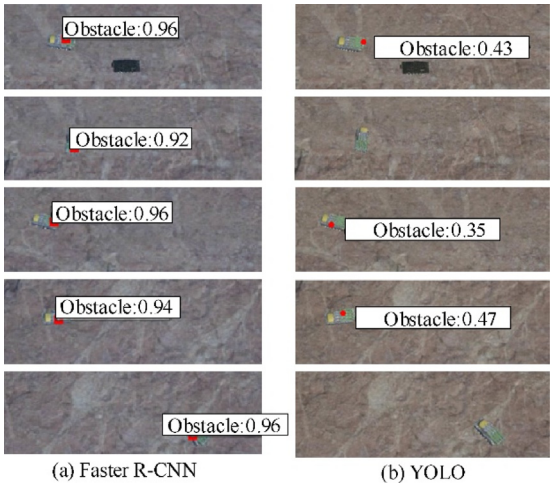


Fig. 9 Detection results of the Faster R-CNN and the YOLO.

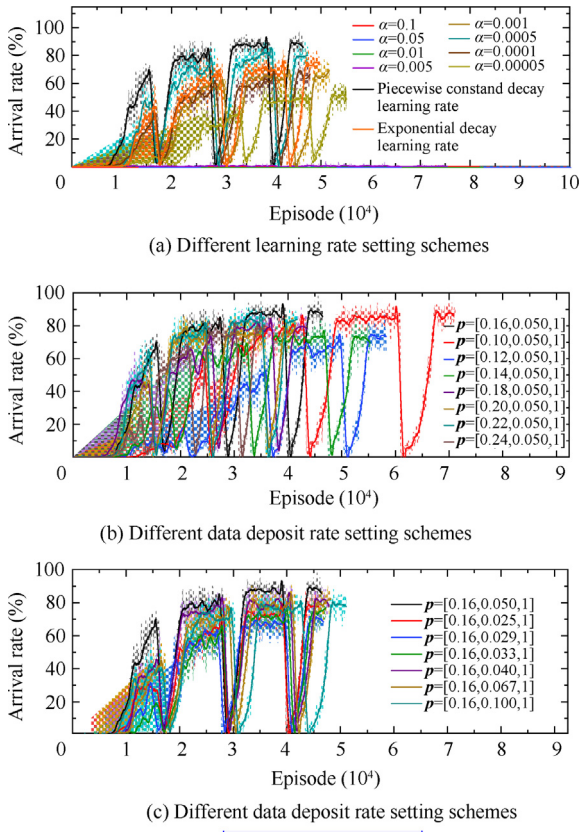
(2) Training results

In this section, the DDM-DQN, DQN, DDQN, and Dueling DQN are trained based on the SVFO, and the O-DQN is trained based on the SVFOO. These methods are all trained in the environment shown in Fig. 6, and training results are shown in Fig. 11, which record the changing trends of the arrival rate in the above methods during the training, respectively. From Algorithm 3, it can be seen that the exploration rate ε is adjusted according to the number of effective steps, and the number of DDM-DQN effective steps grows more slowly than those of the DQN, DDQN, Dueling DQN, and O-DQN, which makes the DQN, DDQN, Dueling DQN, and O-DQN show a shorter fluctuation period. In addition, since the obstacles, starting points, and destinations of each episode are randomly generated, sometimes unreasonable distributions are generated, such as the distance between obstacles being too short or starting points being too close to obstacles, which also leads to fluctuations in the arrival rate. In order to display the trend of data changes during the training process more clearly, actual data is smoothed by the Savitzky-Golay method, where the shaded part is the actual values and the solid line is the smoothed values. From Fig. 11, we can see that the performances of the DQN, DDQN, Dueling DQN, and O-DQN do not improve significantly from 8000 episodes, while that of the DDM-DQN has gradually increased. The final arrival rate of the DDM-DQN converges to 83%, which is higher than those of the DQN, DDQN, Dueling DQN, and O-DQN.

In addition, to further show the changing trend of the training results, the trained models are saved once every 5000 training episodes. After the training, each model is tested for 500 episodes, and the arrival rate is recorded, of which the results are shown in Fig. 12. In the initial phase of training, i.e., before 15000 episodes, the DQN, DDQN, Dueling DQN, and O-DQN perform better than the DDM-DQN. This is due to the fact that the optimization of the network is performed only at effective steps, and every step performed in the DQN, DDQN, Dueling DQN, and O-DQN is an effective step, while

Table 2 Detection results of the Faster R-CNN and the YOLO.

Image number	Faster R-CNN		YOLO	
	Success in recognition or not	Confidence coefficient	Success in recognition or not	Confidence coefficient
1	Success	0.96	Success	0.43
2	Success	0.92	Failure	
3	Success	0.96	Success	0.35
4	Success	0.94	Success	0.47
5	Success	0.96	Failure	



(a) Different learning rate setting schemes

(b) Different data deposit rate setting schemes

(c) Different data deposit rate setting schemes

Fig. 10 Training curves.

only some of the steps performed in the DDM-DQN are effective steps. The lower frequency of optimization in the DDM-DQN at the early stage of training results in a worse performance than those of the DQN, DDQN, Dueling DQN, and O-DQN at this stage. After 15000 episodes, the arrival rate of the DDM-DQN continues to increase, while the performances of the DQN, DDQN, Dueling DQN, and O-DQN

Table 3 Hyperparameter values of the FRDDM-DQN.

Hyperparameter	Value
Network structure	$29 \times 512 \times 128 \times 6$
Minibatch size m	64
Replay memory size $RM_{Capacity}$	300000
Discount factor γ	0.95
Piecewise learning rate $[\alpha_1, \alpha_2, \alpha_3, \alpha_4]$	[0.001,0.0005,0.0001,0.00005]
Boundaries of the piecewise learning rate $[n_1, n_2, n_3, n_4]$	[0,100000, 200000, 300000,350000]
Initial exploration rate ε_0	1.0
Minimum exploration rate ε_{min}	0.001
Exploration rate reset cycle N	100,000
Reset value of the exploration rate ε_{reset}	0.5
Maximum steps per episode T_e	600
Maximum total effective steps T_t	350000
Target network update interval C	3000
Data deposit rate $\boldsymbol{p} = [p_{SE}, p_{DE}, p_{RE}]$	[0.16,0.05,1]

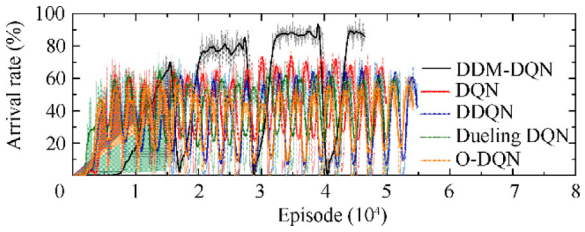


Fig. 11 Training curves of the DDM-DQN, DQN, DDQN, Dueling DQN, and O-DQN.

are limited by the unreasonable shares of various types of experience in the replay memory. As the training proceeds, the arrival rate of the DDM-DQN gradually increases and finally reaches 93%, while the arrival rates of the DQN, DDQN, Dueling DQN, and O-DQN keep fluctuating around 65%. From this training experiment, it can be seen that the convergence effect of the DDM-DQN is better than those of the DQN, DDQN, Dueling DQN, and O-DQN, and the convergence of the DQN is better than those of the DDQN, Dueling DQN, and O-DQN. Next, after combining the DDM-DQN with the Faster R-CNN, the agent trained by the FRDDM-DQN is tested in the test environment built in Unity3D.

4.3. Testing in simulation environments

In this section, the agent trained by the FRDDM-DQN proposed in this paper is marked as agent-FRDDMDQN. The agent-FRDDMDQN controls the UAV to perform ANCA missions, where this UAV is marked as UAV-FRDDMDQN. The test environment is shown in Fig. 7. In this case, the input states of the agent-FRDDMDQN consist of the images captured by the on-board camera and the UAV position information acquired by the GPS and gyroscope device.

In order to verify the effectiveness of the DDM in the proposed method, as a comparison, the DQN, DDQN, and Dueling DQN should be tested in the same scenario. However, due to the fact that the DQN, DDQN, and Dueling DQN do not have object detection capability, after adding the improved Faster R-CNN proposed in Section 3.2, the DQN with the Faster R-CNN (FRDQN), the DDQN with the Faster R-CNN (FRDDQN), and the Dueling DQN with the Faster

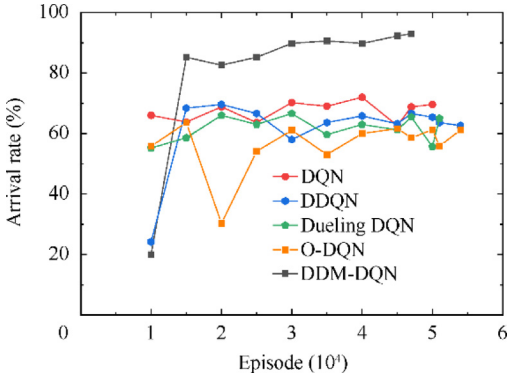


Fig. 12 Arrival rates at different training stages.

R-CNN (FRDueling DQN) are tested, as shown in Table 4. Furthermore, to verify the performance of the Faster R-CNN, after combining the DDM proposed in Section 3.4, the DQN with the YOLO and the DDM (YDDMDQN) is also tested; to verify the effectiveness of the optimization for the Faster R-CNN output proposed in Section 3.3, as a comparison, the DQN with the original Faster R-CNN output (FRODQN) is tested. The abbreviation of the above methods, trained agents and controlled UAVs are listed in Table 5.

4.3.1. Testing with different numbers of static obstacles

In this experiment, agent-FRDDMDQN, agent-FRDQN, agent-FRDDQN, agent-FRDueling DQN, agent-YDDMDQN, and agent-FRODQN are tested in an environment containing multiple static obstacles. In each test scenario, the number of obstacles is sequentially increased from 1 to 12. In each scenario with a different specified number of obstacles, the above-mentioned agents are tested separately for 500 episodes. In each episode, the locations of the obstacles and the starting point and destination of the UAV are generated randomly. Test results are shown in Fig. 13.

From Fig. 13, it can be seen that: (A) the arrival rate of agent-FRDDMDQN is significantly higher than those of other agents in the whole scenario containing multiple static obstacles; (B) agent-FRDDMDQN has the least decrease in the arrival rate. With an increase in the number of static obstacles, the arrival rate of agent-FRDDM gradually decreases from 99.6% to 95.2%, with a 4.4% decrease in the arrival rate. These data for other agents are shown in Table 6. This indicates that a change in the number of static obstacles has a greater impact on agent-FRDQN, agent-FRDDQN, and agent-FRDueling DQN, compared to that on agent-FRDDMDQN. Especially, when a scenario contains more than two static obstacles, the arrival rates of all agents show a rapid decrease, except for that of agent-FRDDMDQN which changes slowly. Due to the addition of the DDM proposed in this paper, the arrival rate of agent-YDDMDQN is higher than those of agent-FRDQN, agent-FRDDQN, and agent-FRDueling DQN. On the other hand, because of the high miss detection rate of the YOLO for small targets, its arrival rate is lower than that of agent-FRDDMDQN.

Compare the FRDQN using the optimized Faster R-CNN output with the FRODQN using the original Faster R-CNN output, the arrival rate of the agent trained by the FRDQN is higher than that of the FRODQN; comparing the methods containing the DDM (i.e., FRDDMDQN and YDDMDQN) with those without the DDM (i.e., FRDQN, FRDDQN, and FRDueling DQN), the arrival rates of the agents trained by the methods containing the DDM are significantly higher; on the basis that both methods include the DDM, comparing the method using the YOLO (YDDMDQN) with that using the Faster R-CNN (FRDDMDQN), the arrival rate of the agent using the Faster R-CNN is higher. Therefore, the agent trained by the FRDDMDQN proposed in this paper performs better and is more stable in scenarios containing static obstacles.

Table 4 Statistics table of features for the methods to be tested.

Method	DDM contained or not	Contained object detection
FRDQN	No	Faster R-CNN
FRDDQN	No	Faster R-CNN
FRDueling DQN	No	Faster R-CNN
FRDDMDQN	Yes	Faster R-CNN
YDDMDQN	Yes	YOLO
FRODQN	No	Faster R-CNN

Table 5 Abbreviations of methods, agents, and UAVs.

Method	Agent trained by method	Agent-controlled UAV
FRDQN	agent-FRDQN	UAV-FRDQN
FRDDQN	agent-FRDDQN	UAV-FRDDQN
FRDueling DQN	agent-FRDueling DQN	UAV-FRDueling DQN
FRDDMDQN	agent-FRDDMDQN	UAV-FRDDMDQN
YDDMDQN	agent-YDDMDQN	UAV-YDDMDQN
FRODQN	agent-FRODQN	UAV-FRODQN

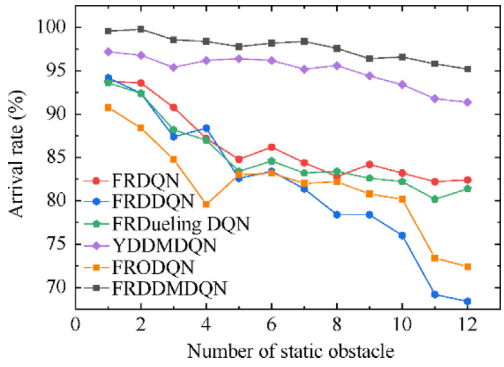


Fig. 13 Arrival rates with different numbers of static obstacles.

4.3.2. Testing with different numbers of moving obstacles

In this experiment, agent-FRDDMDQN, agent-FRDQN, agent-FRDDQN, agent-FRDueling DQN, agent-YDDMDQN, and agent-FRODQN are tested in an environment containing multiple moving obstacles. Similar to the test scenario containing multiple static obstacles, in each test scenario, the number of moving obstacles increases sequentially from 1 to 12. In each scenario with a specified number of obstacles, the above-mentioned agents are tested for 500 episodes, respectively. In each episode, the starting positions of the obstacles and the starting position and destination of the UAV are randomly generated. Test results are shown in Fig. 14.

From Fig. 14, it can be seen that: (A) the arrival rate of agent-FRDDMDQN is higher than those of other agents, especially in tests with a high number of moving obstacles; (B) the decrease in the arrival rate of agent-FRDDMDQN is significantly less than those of other agents. The arrival rate of agent-FRDDMDQN gradually decreases from 98.4% to

Table 6 Arrival rate statistics in a static obstacle scenario.

Agent	Arrival rate with the least obstacles (%)	Arrival rate with the most obstacles (%)	Decrease amount (%)
agent-FRDDMDQN	99.6	95.2	4.4
agent-FRDQN	93.8	82.4	11.4
agent-FRDDQN	94.2	68.4	25.8
agent-FRDueling DQN	93.6	81.4	12.2
agent-YDDMDQN	97.2	91.4	5.8
agent-FRODQN	90.8	72.4	18.4

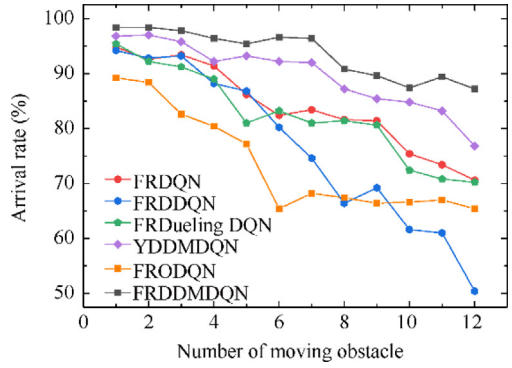


Fig. 14 Arrival rates with different numbers of moving obstacles.

87.2%, a decrease of 11.2%. These data for other agents is shown in Table 7. As shown in Table 7, the decreases in the arrival rates of agent-FRDQN, agent-FRDDQN, and agent-FRDueling DQN are more than twice that of agent-FRDDMDQN. The performance of agent-YDDMDQN is in between those of agent-FRDDMDQN and other agents. Moreover, since moving obstacles are more difficult to avoid, the decreases in the arrival rates of agents in this scenario are slightly higher as the number of obstacles increases compared to those in Section 4.3.1.

Similar to Section 4.3.1, the agent trained by the FRDQN using the optimized Faster R-CNN output outperforms the agent trained by the FRODQN using the original Faster R-CNN output; the agents trained by the methods containing the DDM perform significantly better than those without the DDM; on the basis that both methods include the DDM, the agent using the Faster R-CNN has a better performance than that using the YOLO. Therefore, the agent trained by the FRDDMDQN proposed in this paper performs better in scenarios containing moving obstacles.

4.3.3. Testing with different numbers of moving obstacles in mixed obstacles

In this experiment, agent-FRDDMDQN, agent-FRDQN, agent-FRDDQN, agent-FRDueling DQN, agent-YDDMDQN, and agent-FRODQN are tested in an environment containing multiple static and moving obstacles. In each test scenario, the total number of obstacles is 12, where the number of moving obstacles increases sequentially from 1 to 12. In each scenario with specified numbers of static and moving obstacles, the above-mentioned agents are tested for 500 episodes, respectively. In each episode, the starting positions of the obstacles and the starting point and destination of the UAV are generated randomly. Test results are shown in Fig. 15.

In all tests of the mixed-obstacle scenario, as the total number of obstacles is always 12, the improvement in the arrival rate for all agents as the number of moving obstacles decreases is not as significant as in Sections 4.3.1 and 4.3.2. In particular, for agent-YDDMDQN, the arrival rate fluctuates significantly when the number of moving obstacles is high (more than 5) due to the influence of the YOLO recognition performance. Therefore, from Fig. 15, it can be seen that: (A) similar to Sections 4.3.1 and 4.3.2, the method proposed in this paper that contains the DDM and the Faster R-CNN has the best performance, compared to those methods in Table 5 that do not contain the DDM or use the YOLO or use the original Faster R-CNN output; (B) as the number of moving obstacles changes, agent-FRDDMDQN performs best, and statistics of the arrival rate are shown in Table 8.

Considering the test results of static obstacles (Section 4.3.1), moving obstacles (Section 4.3.2), and mixed obstacles (Section 4.3.3) together, it can be concluded that the agent

Table 7 Arrival rate statistics in a moving obstacle scenario.

Agent	Arrival rate with the least obstacles (%)	Arrival rate with the most obstacles (%)	Decrease amount (%)
agent-FRDDMDQN	98.4	87.2	11.2
agent-FRDQN	94.8	70.6	24.2
agent-FRDDQN	94.2	50.4	43.8
agent-FRDueling DQN	95.4	70.2	25.2
agent-YDDMDQN	96.8	76.8	20.0
agent-FRODQN	89.2	65.4	23.8

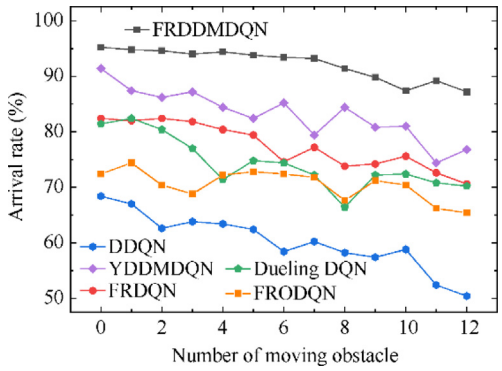


Fig. 15 Arrival rates with different numbers of moving obstacles in mixed obstacles.

trained by the FRDDM-DQN proposed in this paper can achieve ANCA with images captured by the on-board camera, and compared with the other agents (as shown in Table 5), agent-FRDDMDQN has a stronger ANCA capability.

4.4. Analysis of typical episodes

On the basis of the tests in Section 4.3, in order to demonstrate the effectiveness of agent-FRDDMDQN more clearly, this section analyzes typical episodes during testing, including an episode of the UAV performing its mission in a scenario containing multiple static obstacles and an episode of the UAV performing its mission in a scenario containing multiple moving obstacles. As a comparison, the agents trained by other methods in Table 4 are tested in the same scenario. In Fig. 16, the path colors of the agent-controlled UAVs are listed in Table 9.

Figs. 16–18 illustrate the test results of agents in the scenario containing multiple static obstacles. To better display, red spheres are used to represent the no-fly zones generated by obstacles; green and yellow spheres represent the starting point and destination of the UAV. Fig. 16 shows the scenes at different moments. Among them, Fig. 16(e) and (f) represent the moments when UAV-FRDDMDQN and UAV-FR reach the destination, respectively. Fig. 17 shows the image captured by the on-board camera of UAV-FRDDMDQN at the critical moment (i.e., the moment when an obstacle enters the field of view), where the image is part of the input to agent-FRDDMDQN. In this figure, the position of the obstacle is marked with a red rectangle, and the yellow arrows represent the actions that have been performed by UAV-FRDDMDQN in the current time step and the output actions of agent-FRDDMDQN in the next time step, respectively; these flags are not included in the images input to agent-FRDDMDQN. Fig. 18 shows the snapshots of the scenes at the corresponding moments in Fig. 17.

In the initial stage of this test, the flight trajectories of all the UAVs are approximately the same, that is, they are traveling in the direction of destination, as shown in Fig. 16(b). When obstacles enter the on-board camera, agent-FRDQN exhibits different obstacle avoidance strategies from those of other agents. As shown in Fig. 16(c) and Fig. 17(b), when an obstacle appears in the upper right corner of the on-board camera, both left- and right-turn strategies can achieve obstacle avoidance. The right-turn strategy will make the UAV approach the obstacle before moving away from it, while the left-turn strategy moves directly away from the obstacle. That is, for obstacle avoidance, the left-turn strategy is optimal. However, for ANCA, the right-turn strategy is optimal, which can achieve both obstacle avoidance and fast navigation to the destination at the same time. In the end, agent-FRDDMDQN chooses the right-turn strategy, as shown in Fig. 17(b). Compared with agent-FRDDMDQN, the performances of other agents are as follows: (A) UAV-FRDQN shows a significant detour, and its path length is much greater than that of UAV-FRDDMDQN; (B) UAV-FRDDQN fails to avoid obstacles, as shown in Fig. 16(f); (C) UAV-FRDueling DQN, UAV-YDDMDQN, and UAV-FRODQN all succeed in reaching the destination, but take a longer time, as shown in Fig. 16(e) and (f). Therefore, agent-FRDDMDQN trained by the method proposed in this paper performs better than the agents trained by other methods in Table 4 in the test scenarios containing multiple static obstacles.

In addition, we test the agents in Table 4 in scenarios that contain multiple moving obstacles. Figs. 19–21 show the test results. Similar to Figs. 16–18, Fig. 19 shows the scenes at different moments, where Fig. 19(f) represents the moment when UAV-FRDDMDQN reaches the destination; Fig. 20 shows the images captured by the on-board camera when UAV-FRDDMDQN is close to an obstacle; Fig. 21 shows the snapshots of the scenes at the corresponding moments in Fig. 20. In Fig. 19, the black spot in the middle of the red no-fly zone represents the current location of the obstacle, the black trajectory represents the movement path of the obstacle, and the path colors of the agent-controlled UAVs are listed in Table 9.

As shown in Fig. 19(b), from the beginning of the mission, UAV-FRDDMDQN and UAV-FRDDQN navigate toward the destination, which is the fastest navigation direction that can reach the destination in the scenario when the obstacle

Table 8 Arrival rate statistics in a mixed obstacle scenario.

Agent	Arrival rate with the least obstacles (%)	Arrival rate with the most obstacles (%)	Decrease amount (%)
agent-FRDDMDQN	95.2	87.2	8.0
agent-FRDQN	82.4	70.6	11.8
agent-FRDDQN	68.4	50.4	18.0
agent-FRDueling DQN	81.4	70.2	11.2
agent-YDDMDQN	91.4	76.8	14.6
agent-FRODQN	72.4	65.4	7.0

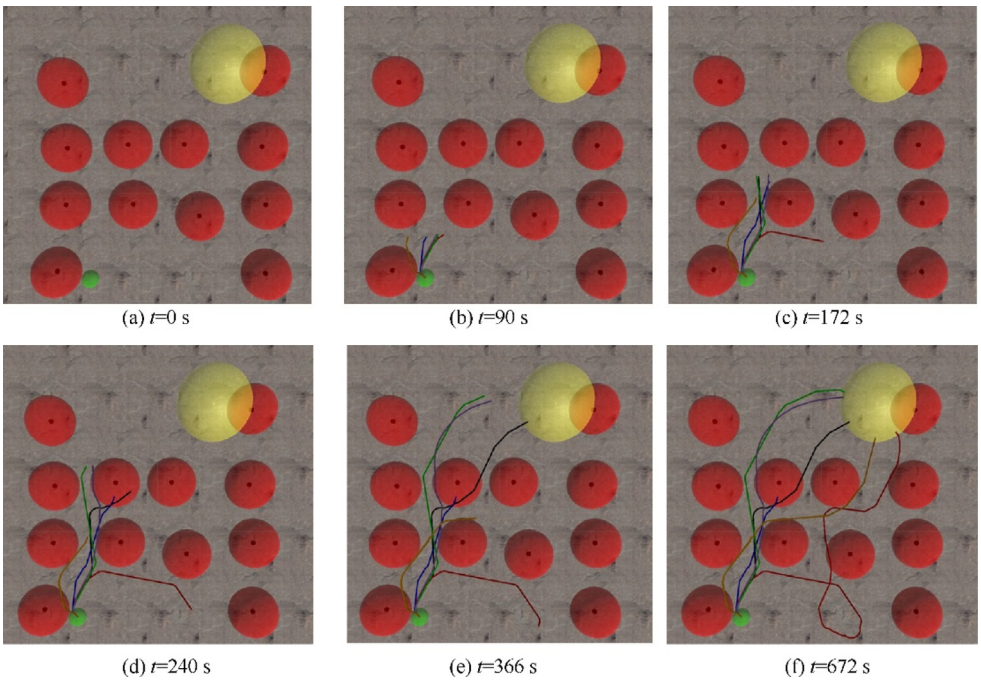


Fig. 16 Trajectories of UAVs in the scenario with static obstacles.

Table 9 Path colors of the agent-controlled UAVs.

Agent	Path color
agent-FRDDMDQN	Black
agent-FRDQN	Red
agent-FRDDQN	Blue
agent-FRDueling DQN	Green
agent-YDDMDQN	Purple
agent-FRODQN	Yellow

information is unknown, while UAV-FRDQN and UAV-YDDMDQN choose the navigation direction that is close to the destination but not the fastest to reach the destination. Among them, due to the misrecognition, it leads to agent-YDDMDQN executing a different obstacle avoidance strategy than that of agent-FRDDMDQN. Moreover, according to Figs. 20–21, it can be seen that when an obstacle appears in the field of view of the on-board camera, agent-FRDDMDQN can adjust its heading angle to avoid the obstacle in time. Particularly, when two obstacles are close together, as shown in Fig. 19(c), agent-FRDDMDQN can also accurately control UAV-FRDDMDQN to cross the no-fly zones created by the obstacles without collision. The performances of other agents are as follows: (A) UAV-FRDQN collides with an obstacle due to too late execution of the obstacle avoidance action, as shown in Fig. 19(d); (B) UAV-FRDDQN, UAV-FRDueling DQN, and UAV-FRODQN fail in their missions due to unreasonable navigation and collision avoidance strategy, as shown in Fig. 19(f); (C) UAV-YDDMDQN fails to avoid obstacles due to misrecognition and delayed recognition of agent-YDDMDQN, as shown in Fig. 19(f). Therefore, agent-FRDDMDQN trained by the method proposed in this paper performs better than the agents trained by other methods in Table 4 in the test scenarios containing multiple moving obstacles.

Synthesizing the results of the above training experiments (Section 4.2), test experiments (Section 4.3), and analysis of typical episodes (Section 4.4), it can be concluded that (A) UAV-FRDDMDQN can implement image-based ANCA; (B) agent-FRDDMDQN proposed in this paper has a significant improvement in performance and has a stronger ANCA capability, compared with the other agents in Table 5.

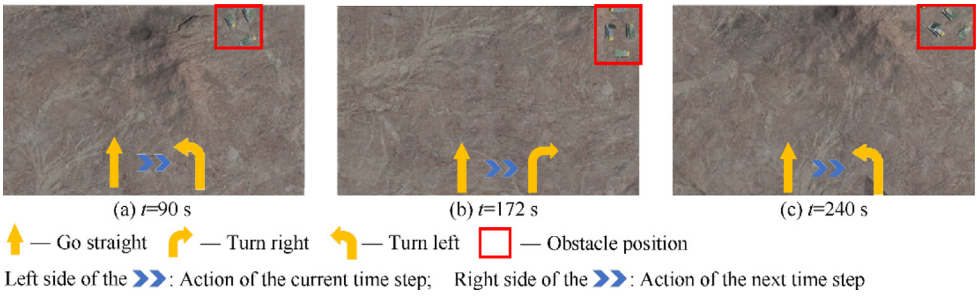


Fig. 17 Images captured by on-board cameras and the output actions of agent-FRDDMDQN in the scenario with static obstacles.

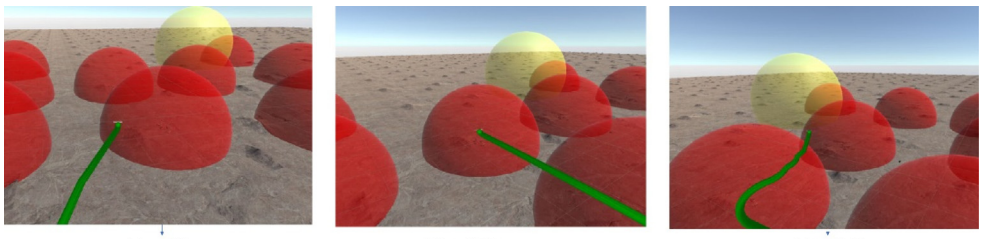


Fig. 18 Snapshots of UAV-FRDDMDQN during its mission in the scenario with static obstacles.

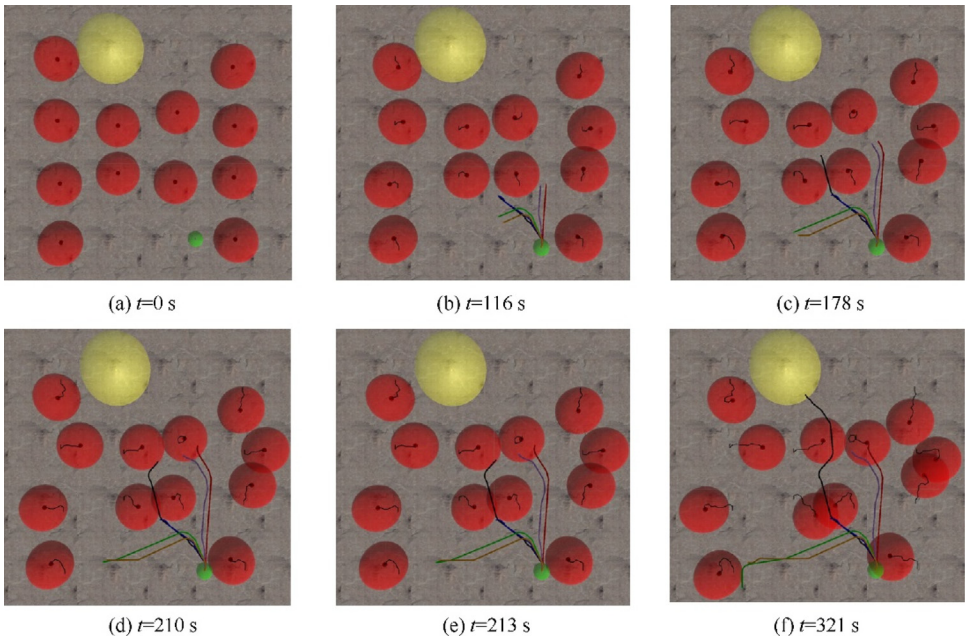


Fig. 19 Trajectories of UAVs in the scenario with moving obstacles.

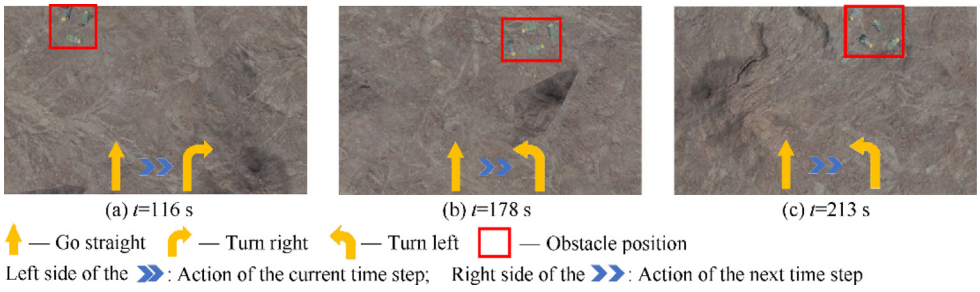


Fig. 20 Images captured by on-board cameras and the output actions of agent-FRDDMDQN in the scenario with moving obstacles.

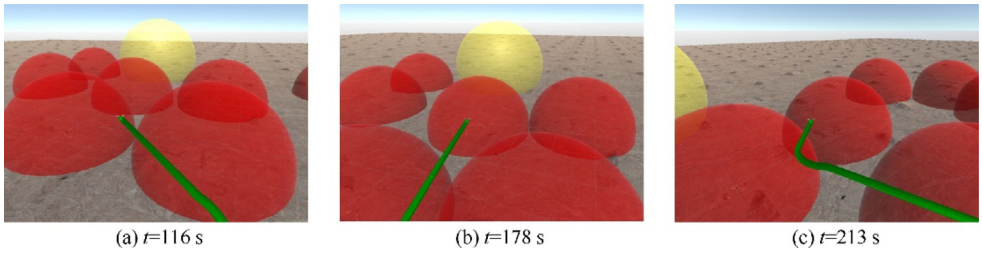


Fig. 21 Snapshots of UAV-FRDDMDQN during its mission in the scenario with moving obstacles.

5. Conclusions

To enhance the ability of UAV ANCA capability in unknown environments, this paper proposes an FRDDMDQN method, in which a Faster R-CNN model is introduced to strengthen the image recognition capability, and a new replay memory data deposit mechanism is designed to improve the training effect. This method enables a UAV to continue its mission in the case of radar and communication failure, by using images captured by an on-board camera and position information obtained by a GPS and gyroscope device. Simulation results illustrate that the agent trained by the FRDDMDQN proposed in this paper can achieve UAV ANCA in an environment containing multiple obstacles, and it has a better performance than those of the agents trained by the FRDQN, FRDDQN, FRDueling DQN, YDDMDQN, and FRODQN.

Although the agent trained by the FRDDMDQN works well in the test experiments, there are still some problems. Firstly, the agent of the FRDDMDQN is trained and tested in a simulation environment without considering the effect of wind speed, while in a real environment, wind-induced interferences are inevitable, which will be addressed by extending the present work. Secondly, obstacle avoidance on land has been achieved in this paper, and further extensions will include obstacle avoidance in air (e.g., enemy aircraft). As the use of UAVs in military applications deepens, the replay memory data deposit mechanism and the two-part training approach proposed in this paper can be extended to multi-UAV collaborative navigation and collision avoidance methods. In addition, in combination with electronic rangefinders, this method can be extended to civilian scenarios, although it is originally proposed to solve problems in military scenarios.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Yang JA, Yin D, Niu YF, et al. Distributed cooperative onboard planning for the conflict resolution of unmanned aerial vehicles. *J Guid Contr Dyn* 2019;**42**(2):272–83.
2. Radmanesh M, Sharma B, Kumar M, et al. PDE solution to UAV/UGV trajectory planning problem by spatio-temporal estimation during wildfires. *Chin J Aeronaut* 2021;**34**(5):601–16.
3. Gu JC, Ding GR, Xu YT, et al. Proactive optimization of transmission power and 3D trajectory in UAV-assisted relay systems with mobile ground users. *Chin J Aeronaut* 2021;**34**(3):129–44.
4. Sangeetha Francelin VF, Daniel J, Velliangiri S. Intelligent agent and optimization-based deep residual network to secure communication in UAV network. *Int J Intelligent Sys* 2022;**37**(9):5508–29.
5. Wang Z, Liu L, Long T. Minimum-time trajectory planning for multi-unmanned-aerial-vehicle cooperation using sequential convex programming. *J Guid Contr Dyn* 2017;**40**(11):2976–82.
6. Guo T, Jiang N, Li BY, et al. UAV navigation in high dynamic environments: A deep reinforcement learning approach. *Chin J Aeronaut* 2021;**34**(2):479–89.
7. Li XH, Xie ZQ, Ye J, et al. An aggregate flow based scheduler in multi-task cooperated UAVs network. *Chin J Aeronaut* 2020;**33**(11):2989–98.
8. Wolek A, Woolsey C. Feasible dubins paths in presence of unknown, unsteady velocity disturbances. *J Guid Contr Dyn* 2015;**38**(4):782–7.
9. Gao C, Zhen ZY, Gong HJ. A self-organized search and attack algorithm for multiple unmanned aerial vehicles. *Aerosp Sci Technol* 2016;**54**:229–40.
10. Girbés V, Vanegas G, Armesto L. Clothoid-based three-dimensional curve for attitude planning. *J Guid Contr Dyn* 2019;**42**(8):1886–98.
11. Otto G, van Schoor G, Uren KR. Geometric-dynamic trajectory: a quaternion Pythagorean hodograph curves approach. *J Guid Contr Dyn* 2021;**44**(2):283–94.
12. Shanmugavel M, Tsourdos A, White BA, et al. Differential geometric path planning of multiple UAVs. *J Dyn Syst Meas Contr* 2007;**129**(5):620–32.
13. Shanmugavel M, Tsourdos A, White B, et al. Co-operative path planning of multiple UAVs using Dubins paths with clothoid arcs. *Contr Eng Pract* 2010;**18**(9):1084–92.
14. Shanmugavel M, Tsourdos A, White BA, et al. Path planning of UAVs in urban region using Pythagorean hodograph curves. *Appl Mech Mater* 2011;**110–116**:4096–100.
15. Cai YZ, Xi QB, Xing XJ, et al. Path planning for UAV tracking target based on improved A-star algorithm. *2019 1st international conference on industrial artificial intelligence (IAI)*. Piscataway: IEEE Press; 2019.p.1–6.
16. Cowlagi RV, Zhang ZT. Route guidance for satisfying temporal logic specifications on aircraft motion. *J Guid Contr Dyn* 2017;**40**(2):390–401.
17. Blum C. Ant colony optimization: Introduction and recent trends. *Phys Life Rev* 2005;**2**(4):353–73.
18. Wu Y, Gou JZ, Hu XT, et al. A new consensus theory-based method for formation control and obstacle avoidance of UAVs. *Aerosp Sci Technol* 2020;**107**:106332.
19. Yokoyama N, Suzuki S. Modified genetic algorithm for constrained trajectory optimization. *J Guid Contr Dyn* 2005;**28**(1):139–44.
20. Eun Y, Bang H. Cooperative task assignment/path planning of multiple unmanned aerial vehicles using genetic algorithm. *J Aircr* 2009;**46**(1):338–43.
21. Jan GE, Sun CC, Tsai WC, et al. An O(nlogn) shortest path algorithm based on delaunay triangulation. *IEEE/ASME Trans Mechatron* 2014;**19**(2):660–6.
22. Gao B, Xu DM, Zhang FB, et al. Constructing visibility graph and planning optimal path for inspection of 2D workspace. *2009 IEEE international conference on intelligent computing and intelligent systems*. Piscataway: IEEE Press; 2009.p.693–8.
23. Krozel J, Andrisani II D. Navigation path planning for autonomous aircraft: Voronoi diagram approach. *J Guid Contr Dyn* 1990;**13**(6):1152–4.
24. Sun CC, Liu YC, Dai R, et al. Two approaches for path planning of unmanned aerial vehicles with avoidance zones. *J Guid Contr Dyn* 2017;**40**(8):2076–83.
25. Terapaptommakol W, Phaoharuhansa D, Koowattanasuchat P, et al. Design of obstacle avoidance for autonomous vehicle using deep Q-network and CARLA simulator. *World Electr Veh J* 2022;**13**(12):239.
26. Huang RN, Qin CX, Li JL, et al. Path planning of mobile robot in unknown dynamic continuous environment using reward-modified deep Q-network. *Optim Control Appl Methods* 2023;**44**(3):1570–87.
27. Anas H, Ong WH, Malik OA. Comparison of deep Q-learning, Q-learning and SARSA reinforced learning for robot local navigation. *International conference on robot intelligence technology and applications*. Cham: Springer; 2022. p. 443-54.

28. Gu YW, Zhu ZT, Lv JD, et al. DM-DQN: Dueling Munchausen deep Q network for robot path planning. *Complex Intell Syst* 2023;**9**(4):4287–300.
29. Yi C, Qi M. Research on virtual path planning based on improved DQN. *2020 IEEE international conference on real-time computing and robotics (RCAR)*. Piscataway: IEEE Press; 2020. p. 387–92.
30. Wu X, Chen HL, Chen CG, et al. The autonomous navigation and obstacle avoidance for USVs with ANOA deep reinforcement learning method. *Knowl Based Syst* 2020;**196**:105201.
31. Ruan XG, Ren DQ, Zhu XQ, et al. Mobile robot navigation based on deep reinforcement learning. *2019 chinese control and decision conference (CCDC)*. Piscataway: IEEE Press; 2019. p. 6174–8.
32. Boudjit K, Ramzan N. Human detection based on deep learning YOLO-v2 for real-time UAV applications. *J Exp Theor Artif Intell* 2022;**34**(3):527–44.
33. Taghibakhshi A, Ogden N, West M. Local navigation and docking of an autonomous robot mower using reinforcement learning and computer vision. *2021 13th international conference on computer and automation engineering (ICCAE)*. Piscataway: IEEE Press; 2021. p. 10–4.
34. Bin Issa R, Das M, Rahman MS, et al. Double deep Q-learning and faster R-CNN-based autonomous vehicle navigation and obstacle avoidance in dynamic environment. *Sensors* 2021;**21**(4):1468.
35. Bin Issa R, Saferi Rahman M, Das M, et al. Reinforcement learning based autonomous vehicle for exploration and exploitation of undiscovered track. *2020 international conference on information networking (ICOIN)*. Piscataway: IEEE Press; 2020. p. 276–81.
36. Hu ZJ, Gao XG, Wan KF, et al. Relevant experience learning: a deep reinforcement learning method for UAV autonomous motion planning in complex unknown environments. *Chin J Aeronaut* 2021;**34**(12):187–204.
37. Li XJ, Liu H, Li JQ, et al. Deep deterministic policy gradient algorithm for crowd-evacuation path planning. *Comput Ind Eng* 2021;**161**:107621.
38. He M, Zhang B, Liu Q, et al. Multi-agent deep deterministic policy gradient algorithm via prioritized experience selected method. *Control and Decision* 2021;**36**(1):68–74 [Chinese].
39. Ke FK, Zhao DX, Sun GD, et al. A priority experience replay sampling method based on upper confidence bound. *Proceedings of the 2019 3rd international conference on deep learning technologies*. New York: ACM; 2019. p. 38–41.
40. Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. *Nature* 2015;**518**(7540):529–33.
41. Singla A, Padakandla S, Bhatnagar S. Memory-based deep reinforcement learning for obstacle avoidance in UAV with limited environment knowledge. *IEEE Trans Intell Transp Syst* 2021;**22**(1):107–18.
42. Ren SQ, He KM, Girshick R, et al. Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Trans Pattern Anal Mach Intell* 2017;**39**(6):1137–49.
43. Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection. *2016 IEEE conference on computer vision and pattern recognition (CVPR)*. Piscataway: IEEE Press; 2016. p. 779–88.
44. Jiang Q, Jia MT, Bi L, et al. Development of a core feature identification application based on the Faster R-CNN algorithm. *Eng Appl Artif Intel* 2022;**115**:105200.
45. Fang Q, Li H, Luo XC, et al. Detecting non-hardhat-use by a deep learning method from far-field surveillance videos. *Autom Constr* 2018;**85**:1–9.
46. Hausknecht M, Stone P. Deep recurrent Q-learning for partially observable MDPs. *AAAI Fall Symp Tech Rep* 2015;**FS–15–06**:29–37.