Journal of Systems Engineering and Electronics
Vol. 32, No. 6, December 2021, pp.1490 – 1508

A learning-based flexible autonomous motion control method for UAV in dynamic unknown environments

WAN Kaifang*, LI Bo, GAO Xiaoguang, HU Zijian, and YANG Zhipeng

School of Electronic and Information, Northwestern Polytechnical University, Xi'an 710072, China

Abstract: This paper presents a deep reinforcement learning (DRL)-based motion control method to provide unmanned aerial vehicles (UAVs) with additional flexibility while flying across dynamic unknown environments autonomously. This method is applicable in both military and civilian fields such as penetration and rescue. The autonomous motion control problem is addressed through motion planning, action interpretation, trajectory tracking, and vehicle movement within the DRL framework. Novel DRL algorithms are presented by combining two difference-amplifying approaches with traditional DRL methods and are used for solving the motion planning problem. An improved Lyapunov guidance vector field (LGVF) method is used to handle the trajectory-tracking problem and provide guidance control commands for the UAV. In contrast to conventional motion-control approaches, the proposed methods directly map the sensor-based detections and measurements into control signals for the inner loop of the UAV, i.e., an end-to-end control. The training experiment results show that the novel DRL algorithms provide more than a 20% performance improvement over the state-of-the-art DRL algorithms. The testing experiment results demonstrate that the controller based on the novel DRL and LGVF, which is only trained once in a static environment, enables the UAV to fly autonomously in various dynamic unknown environments. Thus, the proposed technique provides strong flexibility for the controller.

Keywords: autonomous motion control (AMC), deep reinforcement learning (DRL), difference amplify, reward shaping.

DOI: 10.23919/JSEE.2021.000126

1. Introduction

Over recent decades, there has been an increasing trend in the application of unmanned aerial vehicles (UAVs) in both military and civilian missions, such as intelligence, surveillance, reconnaissance [1], suppression of enemy air defences (SEAD) [2], search and rescue [3], and goods delivery [4]. A common and long-term theme among such tasks is the establishment of an intelligent system for enabling UAVs to perform tasks autonomously without any human intervention [5−7]. In particular, the development of techniques that enable an UAV to fly autonomously from arbitrary starting points to destinations with obstacles avoidance in dynamic unknown surroundings is imminent.

As an open research topic, the UAV autonomous motion control (AMC) problem has attracted sustained attention, a series of methods have been proposed for addressing it. These methods can be generally classified into three groups: rule-based, planning-based, and learning-based. In rule-based approaches, a state trigger mechanism is designed and deterministic control strategies are provided in a hard-coded manner [8−10]. The most common rule-based approach is sensing-avoiding, which directly turns the vehicle with a specified angle to avoid collisions. Chee et al. [8], for example, turned the UAV in the opposite direction whenever an obstacle is sensed in front of it, and then a new path is searched by a rapidly random tree (RRT) algorithm. Panagou [9] designed regular velocity-control laws and used them to realize distributed motion control of vehicles in different situations. Israelsen et al. [10] combined a reactive quad-directional approach with a pre-defined waypoint-tracking algorithm to implement navigation. These rule-based approaches can always produce a feasible solution but not the optimal one. They are only applicable in some simplified deterministic scenarios because it is hard to consider thoroughly and make comprehensive strategies for all possible situations, especially when faced with dynamic environments. Another kind of applicable approach is planning-based, which can address uncertain circumstances typically relying on a model of the environment [11−17]. If the environment model is available in advance, path (re-)planning [11−15] combined with trajectory following [16,17] can be used to realize an optimal controller that drives the UAV for flying across the envi-

Manuscript received December 02, 2020.
*Corresponding author.
This work was supported by the National Natural Science Foundation of China (62003267), the Natural Science Foundation of Shaanxi Province (2020JQ-220), and the Open Project of Science and Technology on Electronic Information Control Laboratory (JS20201100339).

ronment autonomously. A series of specific approaches, such as A* algorithm [11], bio-inspired computation [12], dynamic planning [13,14], and artificial potential fields [15] have been used to generate optimal paths while facing a known environment. If we know nothing about the environment in advance, a simultaneous localization and mapping (SLAM) method [18−21] can be used to construct a map of the environment first, and navigation can then be achieved via online path planning. A common feature of planning-based approaches is their over-reliance on the model of the environment, and an inaccurate model can easily result in poor performance. Besides, this type of method includes the use of an open-loop mechanism that generate strategies without any projection about what's next, which affects its adaptation to dynamic environments. To deal with these shortcomings, reinforcement learning (RL) techniques have been explored by researchers to develop novel learning-based controllers for UAVs.

RL is a suitable method for dealing with sequential decision-making problems in machine learning. Its basic principle is to learn by trial-and-error through an interactive process between an agent and an environment and to select current action based on the predicted long-term cumulative rewards [22]. An RL-based controller can produce strategies adapted to stochastic dynamic environments in the absence of an available model, as long as the training process is performed. To perceive the high-dimensional continuous state space better, a deep neural network is introduced into the conventional RL to develop the deep reinforcement learning (DRL) method. By leveraging the perceptual abilities of deep learning (DL) and the decision-making capabilities of RL, DRLs have performed well in the field of UAV control [5]. In [23], Kersandt tried to construct a high-level controller to navigate an UAV to fly across an area with dense obstacles. The original deep Q network (DQN) [24], double DQN [25], and dueling DQN [26] were directly migrated to an UAV application and trained in a simple virtual environment. The experiments demonstrated the feasibility of all these three algorithms, and also showed their sample-inefficiency in navigating problems. After tens of millions of training episodes, the UAV only gained some simple skills of moving at low speed in a static environment. To improve learning efficiency, Fan et al. [27] introduced a model-based stochastic search strategy to produce high-quality samples for learning, and the learned controller is successfully used for controlling UAV swarms in a hostile environment. Wang et al. [28] changed the parameter-updating strategy from the end-of-an-episode to step-by-step and produced a fast recurrent deterministic policy gradient (Fast-RDPG) algorithm. This algorithm contributed sufficient utilization of the experience and exploration of the environment and has shown great performance in addressing navigation problems in a large-scale complex environment. However, there was an oversimplification of the UAV's control system in [28], where only a discrete flight direction was used for navigating the UAV and the guidance process was discarded. It is good for accelerating the training of the network but not for application. Experiments have shown that a controller learned from this simplified model usually produces unstable and wobbling trajectories. Alejandro et al. [29] adopted the same simplified strategy while addressing an autonomous landing problem, where a DQN-based controller is learned to navigate a UAV landing on a moving platform. To achieve a directive stable control, William et al. [30] introduced the deep deterministic policy gradient (DDPG) [31], the trust region policy optimization (TRPO) [32], and the proximal policy optimization (PPO) [33] to design an intelligent flight control system that provides continuous end-to-end control for the UAVs. The learned controller directly mapped the UAVs ' raw sensory measurements into control signals. It sounds exciting that a complete intelligent controller has been constructed. However, considering various disturbances in practical applications and many classical control theories that have already been proved to be reliable and stable, it is unlikely that we give the entire control authority to artificial intelligence. A better compromise strategy is to construct a learning-based high-level controller for motion planning with a control-theory-based low-level controller for trajectory tracking.

Under motion planning, trajectory tracking is the ultimate objective of autonomous flight control that guarantees the vehicle to fly over a series of particular geolocations. It is a well-studied topic with extensive solutions based on different mathematical theories, such as potential functions [34], vortex fields [35], navigation functions [36], harmonic functions [37], and vector fields [38,39]. Potential functions and vortex fields can provide closed-form solutions by optimizing scalar equations, but it is easy to fall into local minima. The navigation function can avoid local minima with Morse property, but it is hard to determine the Morse parameter in advance. The harmonic function also guarantees a global optimum, but it suffers from a huge computational cost for constructing a harmonic function. To overcome these drawbacks, a vector field is defined to create a control law for vehicle guidance. As a nonlinear cascade control approach, it specifies a desired velocity vector field that provides global attraction to the desired path. Given its simplicity and ease of implementation with limited computational resources, this notion of vector field control has received

considerable recent attention for UAV's guidance in trajectory tracking [40−42]. In this paper, a Lyapunov vector field concept is introduced into DRL to construct the low-level controller.

The main contribution of this paper is that we develop a hierarchical control approach that could realize closed-loop navigation and guidance, and provide flexible and reliable control for UAVs. More specifically:

(i) To maintain flexibility and reliability, we develop a DRL-based motion-control framework for realizing autonomous flight control of UAVs in dynamic unknown surroundings, where novel DRL algorithms are presented for motion planning and an improved Lyapunov guidance vector field (LGVF) method is introduced for trajectory tracking. This hierarchical framework combines the flexibility of the learning-based policy and the reliability of the control-theory-based method.

(ii) To address the sample-inefficiency challenge, we propose two novel learning approaches based on the difference amplifying (DA) technique. As we know, DRLs learn experiences from feedback rewards. An apropriate learning rate and a motivative reward signal will intuitively learn a better agent. Researches in DL show that a dynamic learning rate usually performs better convergence than a stable one. By amplifying the temporal difference (TD), we design a variable learning rate that changes dynamically with the temporal difference error (TD-error). It is an algorithm-independent strategy that can be combined with general DRL algorithms. Another novel learning approach comes from the status quo phenomenon in psychology [43], where a person is inclined to make a decision which has achieved good returns in some similar scenarios. If a person finds a new method of a problem which is better than the previous methods, after confirming that the new method is effective and reliable, he will use this method to solve a similar problem confidently. Correspondingly, if one fails unexpectedly near success, such as an accidental failure with a high score in a video game, one will become conservative if the same situation happens again, or even stay in place without taking risky moves. To model this phenomenon, a parameterized reward function obtained by amplifying the reward difference is formulized. We add the two tricks into some state-of-the-art DRL algorithms and verify their validity and applicability through experiments and tests.

(iii) To maintain the practicability of the learned controller, we construct a general UAV mission environment for the training and testing process. The environment can be used to simulate a series of different dynamic missions and to evaluate the effectiveness of the proposed methods and their adaptability to dynamic environments.

The remainder of this paper is organized as follows: Section 2 presents the formulation of the problem, where an AMC problem is first analyzed, and our DRL-based solving framework is then introduced. Sections 3 presents a discussion of the core solving approaches for the AMC problem, including DA-DRL-based algorithms for motion planning and an LGVF-based algorithm for trajectory tracking. Section 4 introduces the training and testing environment constructed for the experiments, which are presented in Section 5. The performances of the proposed techniques are demonstrated both from the algorithm training perspective and the application testing perspective in Section 5. Section 6 presents the conclusions of this study as well as prospective future works.

2. Problem formulation

2.1 AMC problem analysis

In our autonomous motion-control scenario, a fixed-wing UAV is required to fly across an unknown area until a specified target is finally reached. Such a scenario reflects many real-world applications, e.g., a military attack during a SEAD [2] or a precise target tracking in massively hostile surroundings. In contrast to static motion-control scenarios (such as goods delivery [4]), the UAV herein is expected to navigate more complicated circumstances, that is, dynamic environments with intensively deployed mobile threats and moving targets. To illustrate this idea, Fig. 1 presents the top view of the spatial relationship of the UAV and the environment.

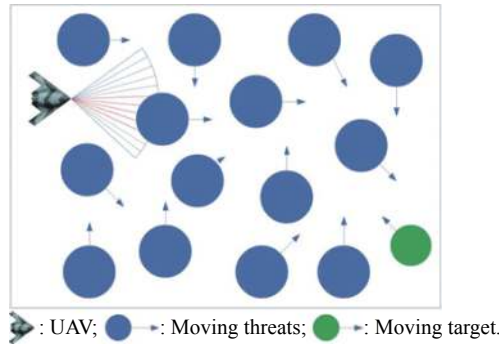


Fig. 1 Top view of the AMC scenario

It is difficult for a UAV to fly autonomously in the environment presented in Fig. 1. There exist at least two challenges in this scenario.

(i) Partial observability of the environment: the information of threats is unknown to the UAV when the mission begins and only partial states of the environment can be obtained by the onboard forward-looking sensor, which makes the hard-coded path-planning methods unsuitable.

(ii) Unpredictability in external conditions: the irregular mobility of the unknown obstacle threats provides the

UAV with an unstable surrounding that influences the performance of the navigation techniques on the basis of SLAM and also leaves the UAV a very short response time. This means that the sensing-planning-based navigation methods could be inefficient for online control.

To cope with these issues, more intelligent motion-control techniques are required to be developed such that the UAV can fly autonomously in various dynamic unknown environments.

2.2 AMC problem-solving framework

In this paper, we focus on learning-based approaches. Specifically, a DRL technique is used to design the controller for the UAV. DRL allows the agent to learn an optimal model of state and action from historical trajectories accumulated through trial-and-error interactions with the environment. The learning-based scheme provides the UAV with the abilities to address the dynamic uncertain states and directly generate a flexible and (near-)optimal control policy sequentially. Fig. 2 illustrates the DRL-based motion-control structure of the UAV.

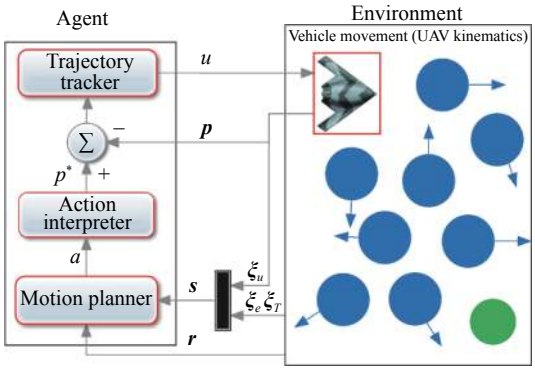


Fig. 2 DRL-based motion-control structure of the UAV

In this control scheme, four core modules collaborate to maintain the UAV's autonomous flight. At each step, the motion planner learns a task-driven favorable action a based on the collected information $(\boldsymbol{s},\boldsymbol{r})$, where the system state $\boldsymbol{s}$ is composed of the UAV state $\boldsymbol{\xi}_u$, the target state $\boldsymbol{\xi}_T$, and environment state $\boldsymbol{\xi}_e$. The action interpreter then translates the high-level action a into a specific reference position $\boldsymbol{p}^*$ such that the trajectory tracker can generate a concrete control command u for the vehicle movement through the UAV kinematics. Four modules are described in the following subsections.

2.2.1 UAV kinematics

A six-degrees-of-freedom (6-DoFs) aircraft model is the most accurate one for UAV flight control. However, by assuming that our UAV flies at a fixed altitude with constant velocity and inertially coordinated [44] turns, we can simplify the UAV kinematics down to four DoFs. It is rational in many real situations, and we can concentrate on developing movement decision approaches [45]. The continuous-time model is defined as follows:

$$\dot{\boldsymbol{\xi}}_u = \frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} x_u \\ y_u \\ \psi_u \\ \varphi_u \end{pmatrix} = \begin{pmatrix} v_u\cos\psi_u + \eta_x \\ v_u\sin\psi_u + \eta_y \\ -(g/v_u)\tan\varphi_u + \eta_\theta \\ u \end{pmatrix} \tag{1}$$

where $(x_u, y_u) \in \mathbf{R}^2$ is the planar position, $\psi_u \in S^1$ is the heading angle, and $\varphi_u \in S^1$ is the roll angle of the UAV. g represents the acceleration of gravity, and v_u denotes the UAV's linear speed. u represents the control command defined by the turn rate of the roll angle ω_u, that is, $u = \omega_u$, and the UAV state is represented by the vector $\boldsymbol{\xi}_u := [x_u\ y_u\ \psi_u\ \varphi_u]^{\mathrm{T}}$. Besides, the model takes the disturbance terms η_x, η_y, and η_θ into consideration, which is drawn from the normal distributions $N(0,\sigma_x^2)$, $N\left(0,\sigma_y^2\right)$, and $N(0,\sigma_\theta^2)$, respectively. Fig. 3 illustrates the mathematical definitions within an attitude map of the UAV.

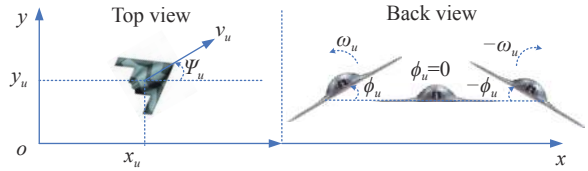


Fig. 3 Mathematical definitions of the UAV

2.2.2 Motion planner

The motion planner is the most important part of the AMC system, which uses the RL scheme to provide adaptable strategies. The RL scheme comprises the use of the Markov decision process (MDP) for modeling the perceiving-acting-learning process of the agent. At time t, the agent makes an action $\boldsymbol{a}_t$ based on the perception of state $\boldsymbol{s}_t$. The generated decision is then executed, and the system state is updated to $\boldsymbol{s}_{t+1}$ with a reward r_t that the agent acquires. In contrast to traditional techniques in which decisions are made on the basis of the instant reward $\boldsymbol{r}_t$, the RL scheme makes the optimal decision by maximizing an expected long-term reward Q^*, which assistis the agent in handling dynamic stochastic status information. The expected long-term reward $Q(\boldsymbol{s}_t,\boldsymbol{a}_t)$ with regard to the given state-action value $(\boldsymbol{s}_t,\boldsymbol{a}_t)$ could be expressed as

$$Q(\boldsymbol{s}_t,\boldsymbol{a}_t) = \mathrm{E}\left[\sum_{l=0}^{H}\gamma^l r(\boldsymbol{s}_{t+l},\boldsymbol{a}_{t+l})|\boldsymbol{s}_t,\boldsymbol{a}_t\right] = \mathrm{E}_{s'\sim\boldsymbol{P}(s'|s,a)}\left(r(\boldsymbol{s}') + \gamma\max_{a'}Q(\boldsymbol{s}',a')\right) \tag{2}$$

where $\boldsymbol{P}(\boldsymbol{s}'|\boldsymbol{s},a)$ is defined as the transition probability from state $\boldsymbol{s}'$ to $\boldsymbol{s}'$ owing to action a. H denotes the scope

of the prediction. $\gamma \in (0,1]$ and represents the discount coefficient. Subsequently, the optimal action could be definded as follows:

$$\pi^*(\boldsymbol{s}_t) = \arg\max_a Q(\boldsymbol{s}_t, a_t). \tag{3}$$

The non-myopic scheme in (3) provides a robust control policy by considering its impact on the future cumulative rewards (illustrated in Fig. 4). In this paper, the action is defined as the desired flight direction, where a = –1, 0, or 1 indicates that the suggestion is to turn left, go straight, or turn right, respectively. Despite the limitation of the action space, it is troubled by the curse of dimensionality while trying to calculate the Q-value exactly. As the agent is facing continuous state space in this scenario, and more seriously, the transition dynamics $\boldsymbol{P}(\boldsymbol{s}'|\boldsymbol{s}, a)$ is unknown to the agent. Any heuristic or evolutionary algorithms become intractable for solving (3). To address this, a deep neural network is designed to approximate the $Q(\boldsymbol{s}_t, a_t)$, and the DQN will be used to provide a sub-optimal motion policy for the UAV. The details will be introduced in Section 3.1.

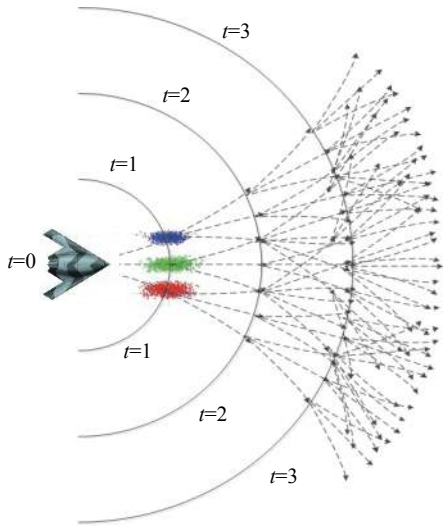


Fig. 4 Diagram of UAV sequential decision process

2.2.3 Action interpreter

The action interpreter is an auxiliary module that connects the motion planner and trajectory tracker. In the control cycle, once an action is selected by the motion planner, the action interpreter has to turn it into a specific reference position immediately. Let us suppose that the decision cycle of the planner is ΔT, and let $\boldsymbol{p}_{u,t} = (x_{u,t}, y_{u,t})$ denote the position of the UAV at time t. For a selected action a_t^*, the UAV has to move to a reference position $\boldsymbol{p}^*_{u,t+\Delta T} = (x_{u,t+\Delta T}, y_{u,t+\Delta T})$ when the time is $t+\Delta T$. $x_{u,t+\Delta T}$ and $y_{u,t+\Delta T}$ are updated as follows:

$$\begin{cases} x_{u,t+\Delta T} = x_{u,t} + v_u \cos(a_t^* \Delta\psi_u)\Delta T \\ y_{u,t+\Delta T} = y_{u,t} + v_u \sin(a_t^* \Delta\psi_u)\Delta T \end{cases} \tag{4}$$

where $\Delta\psi_u$ is the heading angle that can be rotated by performing a single action.

2.2.4 Trajectory tracker

The trajectory tracker is used to make the current position $\boldsymbol{p}_{u,t} = (x_{u,t}, y_{u,t})$ of the UAV track a reference position $\boldsymbol{p}^*_{u,t+\Delta T} = (x^*_{u,t+\Delta T}, y^*_{u,t+\Delta T})$. The objective of the tracker is to generate an optimal tracking controller u_t^* by minimizing a predefined performance function $f(\boldsymbol{p}_{u,t}, \boldsymbol{p}^*_{u,t+\Delta T}, u)$, i.e.,

$$u_t^* = \arg\min_u f(\boldsymbol{p}_{u,t}, \boldsymbol{p}^*_{u,t+\Delta T}, u). \tag{5}$$

To design a proper optimizer for (5), we adopt the Lyapunov function [46] as an indicator of the tracking performance, and the detailed derivation can be found in Section 3.2.

3. AMC problem-solving approaches

This section presents DA-DRL algorithms for providing the UAV end-to-end motion strategies, and an LGVF controller is introduced here for driving it to track a specified trajectory.

3.1 DA-DRL for motion planning

DRL algorithms can be classified into two major types according to the form of the action space. For discrete action, DQN, dueling DQN, double DQN, and some other DQN-based methods perform better, while DDPG, TRPO, and PPO are more suitable for continuous action. In our problem, the action is defined as the requested flight direction in a discrete form. Thus, a basic DQN-based planning architecture is designed for solving the AMC problem, and two DA policies are introduced into some traditional DQN-based algorithms for improving their performances.

3.1.1 DRL-based planning architecture

Since $Q(\boldsymbol{s}, a)$ in (3) is intractable, the approximation is an alternative strategy. As shown in Fig. 4, DRL uses a neural network $Q(\boldsymbol{s}, a; \boldsymbol{\theta})$ with parameter $\boldsymbol{\theta}$ in a critic module to fit $Q(\boldsymbol{s}, a)$. In our scenario, the system state $\boldsymbol{s}$ consists of a UAV state $\boldsymbol{\xi}_u = [x_u\ y_u\ \psi_u\ \varphi_u]^{\mathrm{T}}$, target state $\boldsymbol{\xi}_T = [x_T\ y_T]^{\mathrm{T}}$, and environment state $\boldsymbol{\xi}_e$, where $\boldsymbol{\xi}_e$ is an N_r-dimension vector, and $\boldsymbol{\xi}_e = [d_1, d_2, \cdots, d_{N_r}]^{\mathrm{T}}$ represents the threats' relative distances detected using an N_r-rays LiDAR sensor. N_r respresents the number of the laser beams. Therefore, a system state $\boldsymbol{s}$ with a total of $N_r + 6$ elements is fed into the critic network and three different Q-values are output for three desired flight directions in a. An actor module then selects the optimal ac-

tion a^* through arc $\max_a Q(\boldsymbol{s},a;\boldsymbol{\theta})$ or ϵ-greedy.

In DQN [23] the teamporal difference method is taken, and a decreasing gradient algorithm is introduced to optimize the critic network. An experience replay approach [47] is used in DQN to store all the collected transitions $(\boldsymbol{s},a,\boldsymbol{r},\boldsymbol{s}')$ in a pool such that, on each iteration, a batch of experience data can be sampled for the critic network training. The correlation between the experiences is destroyed by random sampling, so learning efficiency can be guaranteed. Also, a fixed target network strategy [48] is the designated-address compound-error problem, in which a copy of the critic network is stored periodically and used to construct a target network $Q'(\boldsymbol{s},a;\boldsymbol{\theta}')$ for targets generation. Let y denote the target for a given current sample $(\boldsymbol{s},a,\boldsymbol{r},\boldsymbol{s}')$:

$$y(\boldsymbol{s},a)=r(\boldsymbol{s}')+\gamma\max_{a'}Q'(\boldsymbol{s}',a';\boldsymbol{\theta}') \tag{6}$$

where $Q'(\boldsymbol{s}',a';\boldsymbol{\theta}')$ denotes the target network, and the parameters $\boldsymbol{\theta}'$ are copied from $Q(\boldsymbol{s},a;\boldsymbol{\theta})$ at the moment of target network updating. Therefore, the mean square error (MSE) loss function can be expressed as $\text{loss}=(Q(\boldsymbol{s},a;\boldsymbol{\theta})-y(\boldsymbol{s},a))^2$, and the critic network are updated by gradient descent, which is described as

$$\boldsymbol{\theta}_{t+1}=\boldsymbol{\theta}_t+\alpha_t\delta_{q,t}\nabla_\theta Q(\boldsymbol{s},a;\boldsymbol{\theta}) \tag{7}$$

where

$$\delta_{q,t}=y(\boldsymbol{s},a)-Q(\boldsymbol{s},a;\boldsymbol{\theta}_t)=r(\boldsymbol{s}')+\gamma\max_{a'}Q'(\boldsymbol{s}',a';\boldsymbol{\theta}')-Q(\boldsymbol{s},a;\boldsymbol{\theta}_t) \tag{8}$$

is the TD-error.

3.1.2 DA-based learning approaches

The use of experience replay and fixed target networks make it workable to generate solutions for UAV's motion-planning based on DRL. In order to improve performance and stability, DA ideas are introduced into the learning process. To be specific, we design a variable learning rate (see $\alpha(\delta_q)$ in (8)) by amplifying the TD and shape a parameterized reward function (see $r(\delta_r)$ in Fig. 5) by amplifying the reward difference.

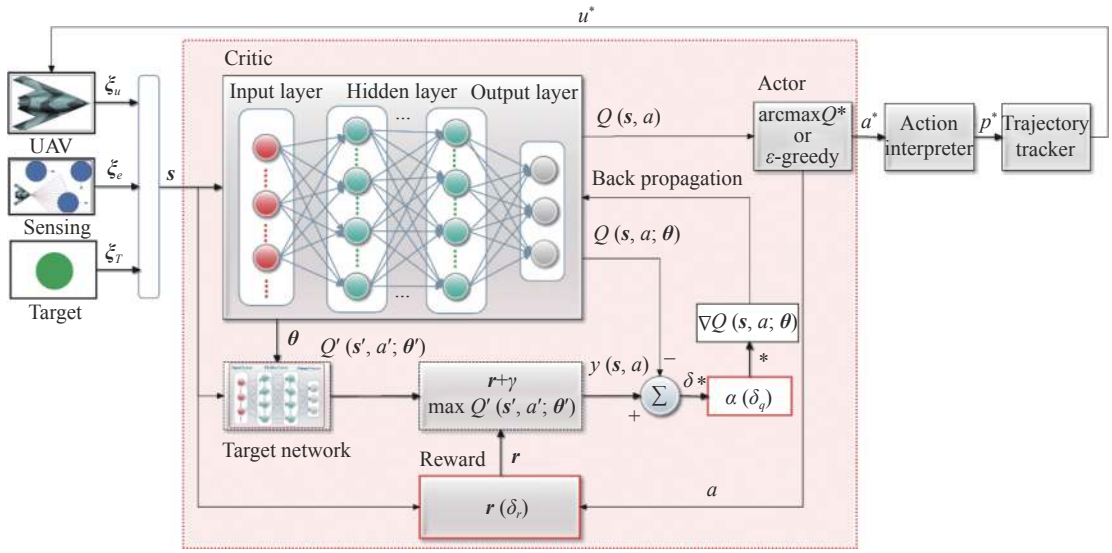


Fig. 5 DRL-based motion planning architecture

(i) Variable learning rate

In RL, designing an appropriate learning rate is crucial, in that a higher rate usually helps to achieve faster convergence, but it also increases the possibility of divergence. In essence, if the learner receives a larger difference between the feedbacks (TD-error), a larger learning rate is suitable for producing a larger update for Q. Similarly, if the difference is small, a fine adjustment of Q is more conducive to convergence, and the learning rate should be relatively small. Consequently, the learning rate should continue to vary in response to the TD-error while interacting with the dynamic environment. This scheme can be described by

$$\alpha(\delta_q)=\max\left(1-\mathrm{e}^{-\frac{|\delta_q|}{\varrho}},\varsigma\right) \tag{9}$$

where ϱ is a tuning parameter that ensures that α is within a reasonable interval, and ς sets a lower bound for α. δ_q is the TD-error defined by (8). This variable scheme can adjust the learning process dynamically and make a balance between the learning speed and the convergence.

Theoretically, to ensure convergence with a probability of one, the learning rate must satisfy the following conditions:

$$\begin{cases} \sum_{k=1}^{\infty} \alpha_k = \infty \\ \sum_{k=1}^{\infty} \alpha_k^2 < \infty \end{cases} \tag{10}$$

where the first inequality in (10) (Condition 1) guarantees a final convergence no matter how far away the initial state is from the optimal one, and the second inequality in (10) (Condition 2) gives a constraint to make sure the stability of the convergence. As for the first condition, the maximizing mechanism and lower bound ς in (9) guarantee the learning rate α will never be equal to 0. That is, Condition 1 is satisfied. Condition 2 requires a gradual decrease of the learning rate until it decreases to to 0. For our learning rate function (9), α is positively correlated with the absolute value of TD-error $|\delta_q|$ in the scope of [0, 1]. That is, as TD-error declines in the learning process, the learning rate will eventually decrease to ς. Although ς holds a small value, it never equals 0, which destroys the satisfaction of Condition 2. However, despite all this, we cannot remove ς from (10) for a practical purpose that the lower bound ς ensures that α does not become too small, which gurantees that the learning process does not become too slow to be accepted. Considering that (10) gives a very strong constraint, in spite of the dissatisfaction of Condition 2, we can still own a convergent learning process by choosing appropriate parameters in (10). The experimental results described in Section 5 verify this possibility.

(ii) Shaped reward function

The reward represents the feedback signals which could be used in the training of the agent. Usually, an effective reward function is built based on rich and available human experience information, which will motivate the agent to generate better behaviors. Three reward schemes are formulated here.

i) Sparse reward. This is the simplest scheme that provides constant rewards under certain fixed conditions. In our task scenario, the UAV agent would receive a positive reward r_a if the UAV arrived at the target position; the UAV would be punished in the event of a collision with a threat obstacle; otherwise, the reward is ruled as zero.

$$r_1(\boldsymbol{s},a) = \begin{cases} r_a, & \text{arrived at the target} \\ r_b, & \text{collided with a threat} \\ 0, & \text{every step} \end{cases} \tag{11}$$

ii) Intermediate reward. The sparse scheme is inefficient because the UAV is required to fly several steps to successfully avoid threats and reach the target, which results in the agent receiving numerous invalid rewards. To address this problem, we design an intermediate reward scheme while considering the characteristics of the motion-planning problem. The main idea is to add progressive reward signals according to the relative situation at each iteration step.

$$r_2(\boldsymbol{s},a) = \begin{cases} r_a, & \text{arrived at the target} \\ r_b, & \text{collided with a threat} \\ \mu_1(D_{\text{ut}}^{\text{pre}} - D_{\text{ut}}^{\text{cur}}) + \mu_2\left(-\dfrac{\Delta\psi}{4}\right) + \mu_3\left(\dfrac{D_f}{D_s} - 1\right), & \text{every step} \end{cases} \tag{12}$$

where $D_{\text{ut}}^{\text{pre}}$ and $D_{\text{ut}}^{\text{cur}}$ represent the previous distances and current between the UAV and the target, respectively; $\Delta\psi$ denotes the angle of the UAV flight direction deviating from the target; D_s is the detection distance of the laser; D_f is the distance of the detected threat in front of the UAV, and if there is no threat ahead of it, D_f is set as D_s. All the variables can be observed in Fig. 6. Obviously, the three sub-items in (12) denotes three parts of the rewards about distance, angle, and threat. And, three relative gain factors μ_1, μ_2, μ_3 are introduced to optimize the sum of reward r_2.

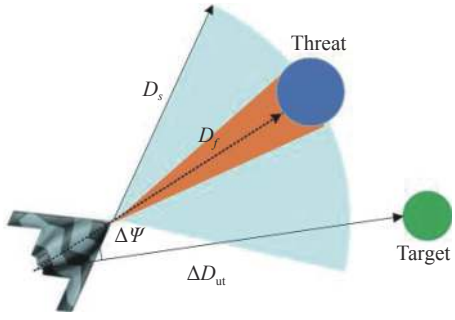


Fig. 6 Relative situations among UAV, threat, and target

iii) Difference amplified reward. It is derived from the status quo phenomenon in psychology [43], in which a person is proved more likely to take any action that has brought him a better reward in some similar situations as described in Section 1. To model these experiences, the reward difference $\delta_r = (r_t - r_{t-1})/\min(r_t, r_{t-1})$ is defined as a signal to optimize the reward. The transform law is as follows:

$$r_3(\delta_r) = \begin{cases} r_t + |r_t|\left(\arctan\left(\dfrac{\delta_r + \lambda}{\eta} \cdot \dfrac{\pi}{2}\right) - \lambda\right), & |\delta_r + \lambda| > \eta \\ r_t, & \text{otherwise} \end{cases} \tag{13}$$

where r_t and r_{t-1} represent the rewards at time t and time $t-1$; $\lambda = \text{sgn}(r_t - r_{t-1})$ is used to balance the formula such that a negative difference results in $\lambda = -1$ and a positive difference results in $\lambda = 1$. η is a tuning factor that provides agent with adaptability for different tasks. In fact, based on the existing reward function, some of the reward values are dynamically adjusted, so that the adjustment can make the reward function more in line with

human psychological expectations [49]. Besides, the difference amplification is a scheme used for adjusting the existing reward functions so that they are applicable for many DRL-based training processes. This strategy makes the agent develop its strong points in making decisions and makes it easier to generate the optimal solution to the problem. On combining the variable learning rate and the shaped reward function and adding it to the classic DQN, we obtain the DA-DRL-based motion-planning algorithm in Algorithm 1.

Algorithm 1 DA-DRL-based motion-planning algorithm (with DQN)

1: initialize hyper-parameters: experience replay buffer D, mini-batch size B, target network update frequency K, Q network with arbitrary weights $\boldsymbol{\theta}$, target network Q' with weights $\boldsymbol{\theta}' = \boldsymbol{\theta}$, greedy exploration rate , maximum motion steps of the agent in each episode T target networks update interval I

2: repeat (for each episode)

3: initialize $\boldsymbol{s}_0 \leftarrow (\boldsymbol{\xi}_{u,0}, \boldsymbol{\xi}_{T,0}, \boldsymbol{\xi}_{e,0})$ randomly;

4: while (not arrived at target and not collided and t < T) do

5: choose a_t randomly with ϵ-greedy or $\boldsymbol{a}_t = \arg\max_a Q(\boldsymbol{s}_t, a; \boldsymbol{\theta})$

6: $u_t \leftarrow$TrajectoryTracker (ActionInterpreter ($\boldsymbol{a}_t$));

7: execute u_t, new system state $\boldsymbol{s}_{t+1} \leftarrow (\boldsymbol{\xi}_{u,t+1}, \boldsymbol{\xi}_{T,t+1}, \boldsymbol{\xi}_{e,t+1})$;

8: intermediate reward $\boldsymbol{r}_t \leftarrow$Equation (11);

9: difference amplified reward $\boldsymbol{r}'_t \leftarrow$Equation (13);

10: store $(\boldsymbol{s}_t, a_t, \boldsymbol{r}'_t, \boldsymbol{s}_{t+1})$ in D;

11: sample transitions $[\boldsymbol{s}_j, \boldsymbol{a}_j, \boldsymbol{r}'_j, \boldsymbol{s}_{j+1}]_{j=1,\cdots,b}$ from D;

12: $\delta_{q,t,j} = \boldsymbol{r}'_j + \gamma \max_{a'} Q'(\boldsymbol{s}_{j+1}, a'; \boldsymbol{\theta}') - \boldsymbol{Q}(\boldsymbol{s}_j, a_j; \boldsymbol{\theta}_t)$;

13: learning rate $\alpha_t \leftarrow$Equation (9);

14: $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \alpha_t \boldsymbol{\delta}_{q,t} \nabla_\theta Q(\boldsymbol{s}_t, a_t; \boldsymbol{\theta}_t)$;

15: $\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta}$ if t mod K = 0;

16: end while

3.1.3 Complexity analysis

Suppose the Q network is an all-connected neural network, which contains M hidden layers and each hidden layer owns N nodes. In this paper, the input layer has $N_r + 6$ nodes (N_r owns a determined value) and output layer has three nodes. The complexity of a feedforward calculation with one sample is $O((N_r+6)N + N^2 + \cdots + 3N) = O(N^2)$. As the mini-batch size is B, the complexity of a feedforward calculation with B samples is $O(BN^2)$. Let's take the eposide number E and eposide length T into account, we can count the complexity line by line:

$$
\begin{aligned}
&O(7) + O(1) + O(E(N_r+6)) + O(E) + O(ETN^2) + \\
&O(ET10) + O(ET(N_r+6)) + O(ET) + \\
&O(ET) + O(ET(2N_r+14)) + \\
&O(ETB(2N_r+14)) + O\left(ETB\left(2N^2\right)\right) + \\
&O(ET) + O(ETB(N^2)) + O(1) + O(1) + O(1) = \\
&O(ETB(N^2)).
\end{aligned}
$$

Then the complexity of the DA-DRL-based motion-planning algorithm is $O(ETB(N^2))$.

3.2 LGVF for trajectory tracking

After action a_t is converted into a reference position $\boldsymbol{p}^*_{u,t+\Delta T} = (x^*_{u,t+\Delta t}, y^*_{u,t+\Delta T})$, a controller u_t is required to drive the UAV to fly to the reference position in ΔT seconds. In this work, we design a controller of the roll-angle turning rate based on the LGVF [38,39] while the original one can only provide a heading control for 3-DoFs kinematic models.

Let $\boldsymbol{r} = \boldsymbol{p}_u - \boldsymbol{p}^*_u = [x_r, y_r]^{\mathrm{T}}$ denote the relative position. The guidance vector field (GVF) method drives the UAV to fly around the reference position with a radius r_d instead of directly flying to the reference position. The anticlockwise GVF is defined as follows:

$$
f(\boldsymbol{r}) = \begin{bmatrix} \dot{x}_d \\ \dot{y}_d \end{bmatrix} = \frac{-v_d}{r(r^2 + r_d^2)} \begin{bmatrix} (r^2 - r_d^2) & 2rr_d \\ -2rr_d & (r^2 - r_d^2) \end{bmatrix} \begin{bmatrix} x_r \\ y_r \end{bmatrix} \tag{14}
$$

where $\boldsymbol{v}_d = [\dot{x}_d, \dot{y}_d]^{\mathrm{T}}$ is the desired relative velocity, $r = \|\boldsymbol{r}\|$, and $v_d = \|\boldsymbol{v}_d\|$. The desired relative heading ψ_d and the desired relative heading rate $\dot{\psi}_d$ can then be calculated as follows:

$$
\psi_d = \arctan\frac{\dot{y}_d}{\dot{x}_d} = \arctan\frac{y_r \cdot (r^2 - r_d^2) - x_r \cdot 2rr_d}{x_r \cdot \left(r^2 - r_d^2\right) + y_r \cdot 2rr_d}, \tag{15}
$$

$$
\dot{\psi}_d = 4v_d \frac{r_d r^2}{\left(r^2 + r_d^2\right)^2}. \tag{16}
$$

According to Wu et al. [46], the absolute heading rate $\dot{\psi}_u$ is generated by

$$
\dot{\psi}_u = -k(\psi_r - \psi_d) + \frac{\dot{\psi}_d}{k_\psi} \tag{17}
$$

where ψ_r is the relative heading angle and k is a gain. Since $\boldsymbol{p}^*_u$ does not move within time ΔT, i.e., $\dot{\boldsymbol{p}}^*_u = (\dot{x}^*_u, \dot{y}^*_u) = 0$. Subsequently, ψ_r=ψ_u and $k_\psi = 1$ can be derived from the relative kinematics model [46] and we obtain a simplified (17) as

$$
\dot{\psi}_u = -k(\psi_u - \psi_d) + \dot{\psi}_d. \tag{18}
$$

We then derive the roll rate $\dot{\varphi}_u$ according to (15)–(18) and $\dot{\psi}_u = -(g/v_u)\tan\varphi_u$.

$$\dot{\varphi}_u = \frac{-v_u/g}{1+(v_u/g)^2\dot{\psi}_u^2}\ddot{\psi}_u = \frac{-(v_u/g)\left[k^2(\psi_u-\psi_d)-\ddot{\psi}_d\right]}{1+(v_u/g)^2\left[\dot{\psi}_d-k(\psi_u-\psi_d)\right]^2} \tag{19}$$

where ψ_d and $\dot{\psi}_d$ can be calculated by using (15) and (16), and $\ddot{\psi}_d$ can be calculated by using (20):

$$\ddot{\psi}_d = 8v_d^2\frac{rr_d^2}{\left(r^2+r_d^2\right)^3}. \tag{20}$$

Finally, we obtain our control command u^* with the consideration of UAV kinematic constraints

$$u^* = \min(\omega_{u,\max}, \dot{\varphi}_u). \tag{21}$$

4. Training and testing environment

As is known, the DRL requires a mission environment for its interactive learning. There is no exception for our DRL-based solutions to the motion-control problem. As an important contribution, a general simulation scenario is built for the training and testing requirements, as shown in Fig. 7. The environment simulates a world of 400 m × 300 m, and a series of obstacles (or threats) are randomly scattered in the world (see the white cylinders in Fig. 7). The green circle denotes the target, and the blue agent is a fixed-wing UAV equipped with a LiDAR sensor for detecting and ranging threats and targets. The blue sector in front of the UAV represents the detection capability area of the sensor. Whenever an object is detected, the corresponding blue beams are set as red so that the user can intuitively observe the interaction between the UAV and the environment. A real-time collision detection system is used in this environment to address the interactions of collisions and sensor detections.

Fig. 7 Training and test environment

The simulated environment is developed in Python with the support of a visualization framework called Director. To ensure the continuity of our research work, we adopt an advanced modularized design, which makes our environment adaptable and easy to expand. The threats and targets can be stationary or movable and can be set to an arbitrary shape, size, and position. We provide various sensors and dynamics for the UAV and provide multiple scenario options for different mission requirements. Both single-agent and multi-agent applications can use our environment as the training and testing platforms. Moreover, we continue to improve this environment and prepare to submit it to Github to share it with more researchers.

5. Numerical simulation results and analysis

5.1 Environmental settings

In our scenario, a UAV is required to travel through an unknown environment to find a target. The UAV is equipped with a sensor that is capable of detecting up to a distance of 40 m ahead and ±45° from left to right. The sensor uses 30 beams for distance detection in the area. The velocity of the UAV is set as 15 m/s, and the disturbance parameters are set as $\sigma_x = \sigma_y = 0.1$, and $\sigma_\theta = 0.01$. The maximum heading angle rotating in one action $\Delta\psi_u$ is set as 10°, and the turning radius is set as $r_d = 4$ m.

Before its application, the DRL-based controller is first required to be trained. As described in Section 3.1, a neural network constitutes the core of the motion planner, and the network is constructed by 36×100×100×3 fully connected neural networks based on UAV state, target state, and sensed information. To demonstrate the performance of our DA-based learning approaches, we conducted four sets of experiments, wherein the different learning approaches are bound to the DQN and are trained in a static environment. Table 1 shows the differences between the introductions of the four experiments, where different DA schemes are bounded to DQN. In the DQN, the model is trained with a stable learning rate of $\alpha = 0.001$ and an intermediate reward (see (11)), where the parameters are set as $r_a = 3$, $r_b = -3$, $\mu_1 = 0.3$, $\mu_2 = 0.4$, and $\mu_3 = 0.5$. In the DQN with DA1, a variable learning rate (see (9)) is used to replace the stable one, where $\varrho = 100$ and $\varsigma = 0.001$ are set as (9). The third one, DQN with DA2, uses the same stable learning rate $\alpha = 0.001$ and a difference amplified reward (see (13)) with $\eta = 2$, and the last one, the DQN with DA3 combines the DQN along with a variable learning rate (see (9)) and a difference amplified reward (see (13)). The other common hyperparameters are set as follows: the discount coefficient $\gamma = 0.9$; mini-batch size $B = 32$; experience pool capacity $D = 5\,000$, and target network update frequency $K = 200$. A variable ϵ-greedy $\epsilon_t = \max(1 - N_l\Delta\epsilon, \epsilon_0)$ is used, where N_l is the number of learnings that has been performed, Δ is the reduction factor with a value of 0.000 02, and the least value of ϵ is set as $\epsilon_0 = 0.000\,1$. At the beginning of each

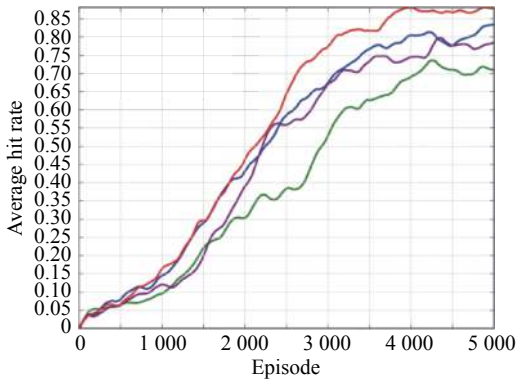
trainging round, the initial information of the UAV, target, and threats are randomly generated. All the models are employed using Tensorflow, and the models are trained by using a GeForce RTX 2080 graphics processing unit in 5 000 episodes.

Table 1 Experiments and algorithms

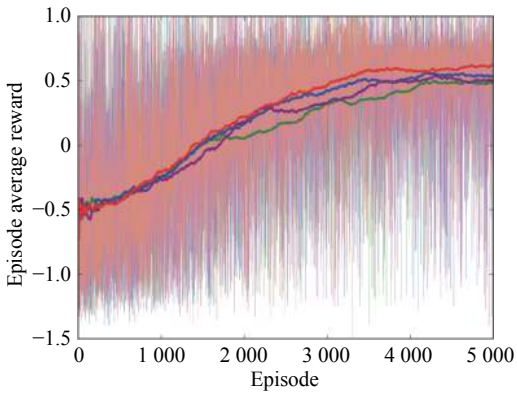
Name	Definition of algorithms
DQN	DQN with an intermediate reward and a stable learning rate
DQN with DA1	DQN with an intermediate reward and a variable learning rate
DQN with DA2	DQN with a difference amplified reward and a stable learning rate
DQN with DA3	DQN with a difference amplified reward and a variable learning rate

5.2 Training in static environments

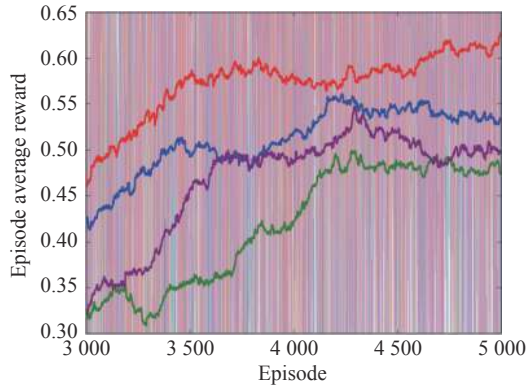
To evaluate the performance, we use the average rewards and average hit rates as quantified indicators of success, where the hit rate represents the success rate of the UAV hitting a target validly in the last 100 episodes. The experiment results are illustrated in Fig. 8. As observed in Fig. 8, all the three improved algorithms achieved better convergences than the original DQN both in terms of the average hit rate (left) and the episode average reward (right).



(a) Average hit rate curve



(b) Episode average rewards curve



(c) Partially enlarged detial of (b)

—: DQN; —: DQN with DA1;
—: DQN with DA2; —: DQN with DA3.

Fig. 8 Convergence curves of the four different algorithms

The DQN with DA3 (the red line) achieves the fastest convergence, the highest hit rate, and the largest average reward among the four algorithms because it integrates both the DA strategies, which can give the agent a greater incentive signal and a more-precise parameter-tuning mechanism. Specifically, DA3 brings a 24.0% promotion of the average hit rate for DQN and a 34.7% promotion of the average rewards for DQN, respectively. Based on further verifications of the adaptability, we bind the two DA-based learning approaches to the other two DRL algorithms—double DQN and dueling DQN—and perform comparative experiments similar to DQN. This time, we select the hit rate at the final convergence and the episode number when the hit rate first reaches 80% as indicators for directly quantifying the promotions of the novel algorithms relative to the original ones.

As shown in Table 2, the conventional algorithms in the rows plus the learning strategies in the columns provide new algorithms. For example, a double DQN in the third row plus a DA2 in the fourth column constructs an algorithm of double DQN with DA2, where the DA1, DA2, and DA3 have the same definitions as those described in Table 1. From Table 2, we can observe that all the learning strategies DA1, DA2, and DA3 provide the three conventional DRL algorithms with higher hit rates. To be specific, from the perspective of the hit rate at the final convergence, DA1, DA2, and DA3 bring learning effectiveness promotions of 17.1%, 10.3%, and 24.0%; 5.1%, 5.6%, and 12.8%; and 1.1%, 6.6%, and 11.9% for the DQN, double DQN, and dueling DQN, respectively. Meanwhile, from the perspective of episode number when the hit rate first reaches 80%, DA1, DA2, and DA3 bring learning speed promotions of 20.1%, 12.1%, and 38.3%; 28.4%, 22.7%, and 41.8%; 16.2%, 14.2%, and 33.2% for the DQN, double DQN, and dueling DQN, respectively. The vacancy in the third row and the sixth

column means that the hit rate of the DQN never reaches 80% in the learning process. With no loss of generality, the vacancy is set as the maximum episode (5 000) while calculating the promotions.

Table 2 Results of algorithms

Controller	Hit rate at final convergence				Episode number when the hit rate first reaches 80%			
	+Null	+DA1	+DA2	+DA3	+Null	+DA1	+DA2	+DA3
DQN	0.721	0.844	0.795	0.894	–	3 995	4 397	3 084
Double DQN	0.812	0.853	0.867	0.925	4 983	3 567	3 854	2 902
Dueling DQN	0.854	0.863	0.910	0.956	4 125	3 458	3 541	2 756

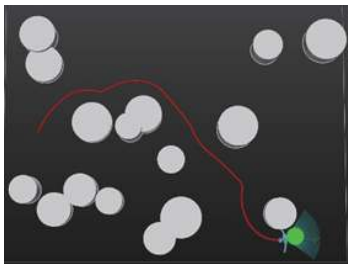
5.3 Exploiting in dynamic environments

After a series of full training performed in different settings, the autonomous motion controller is finally constructed. The controller is required to be tested in applications. In this section, we prove that our DRL-based controller, which is trained in static environments, can adapt well to other unknown dynamic environments without retraining. To ensure comprehensive proof, we design four different kinds of test experiments. Experiment I contains some new unknown environments that are completely different from the training one and four controllers are tested in them. Experiment II tests some weak dynamic circumstances with pop-up threats. Experiment III tests a strong dynamic environment with movable threats all around. Experiment IV tests a stronger dynamic circumstance of tracking a moving target in dynamic environments. In the exploiting process, four pre-trained controllers in Table 1 are used for driving the UAV to fly across the environment until the target arrives. To conduct a deeper comparison, a traditional path planning method of the artificial potential function (APF) is used to construct the fifth controller. There are a total of $4\times3\times5=60$ comparative experiments conducted.

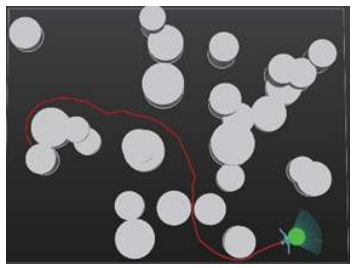
(i) Experiment I: testing in complicated environments

In this experiment, environments with different threat densities are created for testing the effectiveness of the constructed controllers and their adaptabilities to complicated environments. Specifically, a threat density of 15%, 25%, and 35% are respectively selected for randomly creating threats in the mission area (as shown in the three columns in Fig. 9). In each environment, five controllers are used to drive the UAV fly from the same initial position to the same target while avoiding the threats. The flight trajectories are shown in Fig. 9 and the trajectory parameters are listed in Table 3.

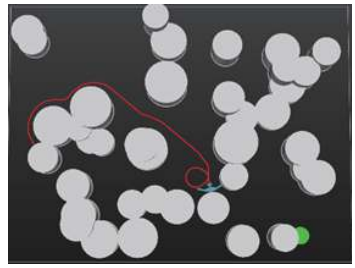
In Fig. 9 the rows represent controllers and from top to bottom are APF, DQN, DQN with DA1, DQN with DA2, DQN with DA3, respectively. The columns represent different environments with a threat density of 15%, 25%, and 35%, respectively. From the screenshots in Fig. 9, we can observe that all the five controllers successfully drive the UAV to the target while facing a low threat density environment (Fig.9(a1)−Fig.9(e1)).



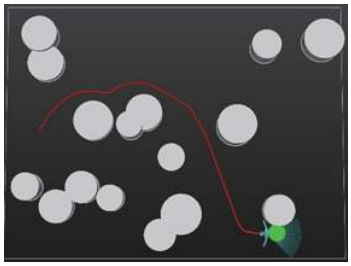
(a1) APF algorithm in environments with a threat density of 15%



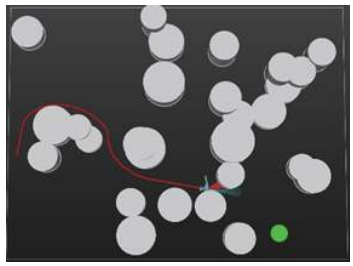
(a2) APF algorithm in environments with a threat density of 25%



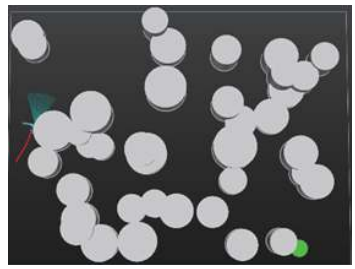
(a3) APF algorithm in environments with a threat density of 35%



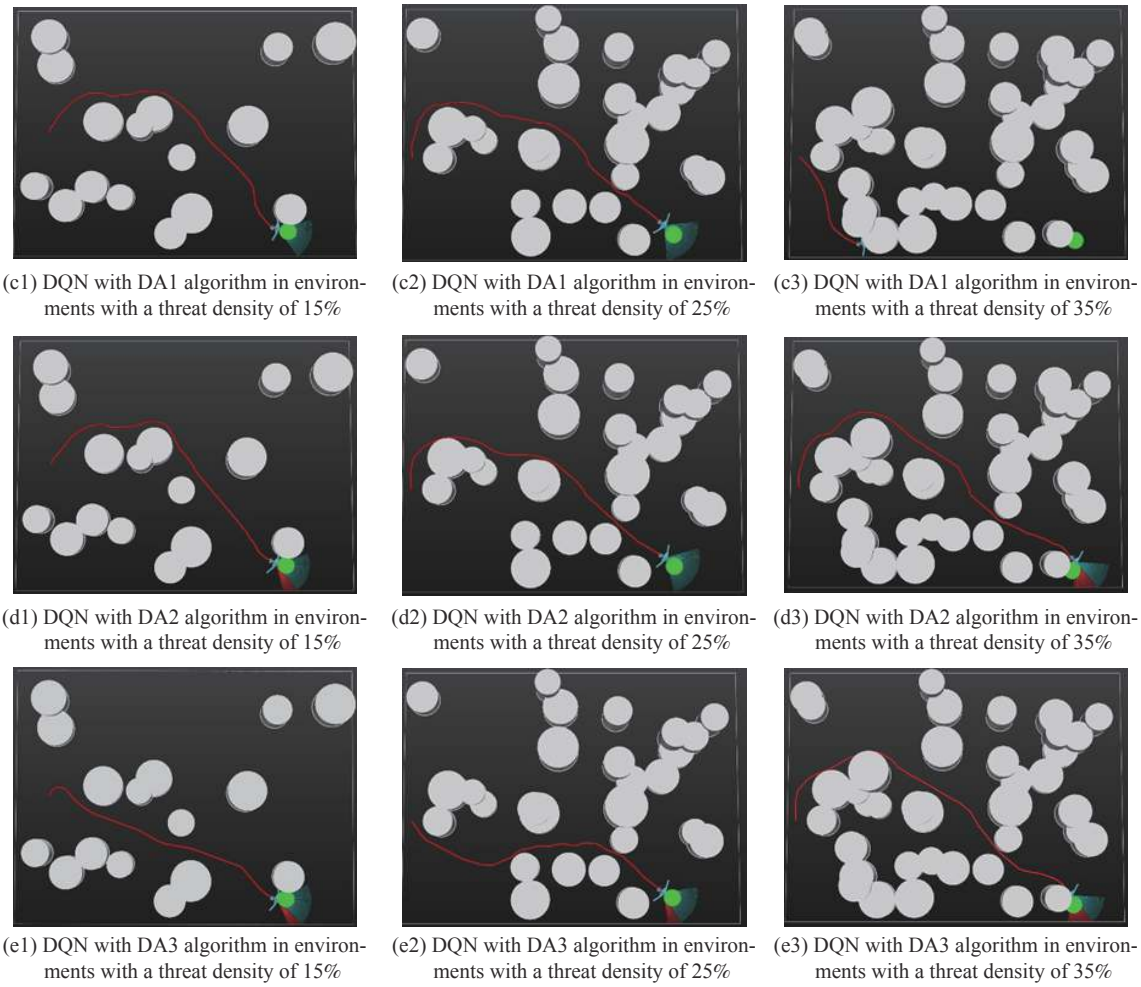
(b1) DQN algorithm in environments with a threat density of 15%



(b2) DQN algorithm in environments with a threat density of 25%



(b3) DQN algorithm in environments with a threat density of 35%



(c1) DQN with DA1 algorithm in environments with a threat density of 15%
(c2) DQN with DA1 algorithm in environments with a threat density of 25%
(c3) DQN with DA1 algorithm in environments with a threat density of 35%
(d1) DQN with DA2 algorithm in environments with a threat density of 15%
(d2) DQN with DA2 algorithm in environments with a threat density of 25%
(d3) DQN with DA2 algorithm in environments with a threat density of 35%
(e1) DQN with DA3 algorithm in environments with a threat density of 15%
(e2) DQN with DA3 algorithm in environments with a threat density of 25%
(e3) DQN with DA3 algorithm in environments with a threat density of 35%

Fig. 9 Trajectories of the UAV with different controllers while flying across environments with different threat densities

In spite of different tracks, the controller of DQN with DA3 flies a smoother and shorter path than the others. For the medium threat density (Fig.9(a2)−Fig.9(e2)), the DQN-based controller failed at the time 15.7 s and the three of the rest finally complete the mission although they are facing environments that they never meet before. With further increase of the treat density (Fig.9(a3)−Fig.9(e3)), the number of crashed UAVs increases, and only the controllers of DQN with DA2 and DQN with DA3 successfully reach the target. These results show that our proposed DA policy could greatly improve the adaptability to a complicated unknown environment of the traditional DQN algorithm.

Furthermore, we list the detailed trajectory parameters in Table 3, where the flight times and path lengths are recorded for evaluating the effectiveness. It is clear that the controller of DQN with DA3 performs the best mission effectiveness in all three circumstances with different threat densities, because for the same mission, DQN with DA3 usually spends shorter flight time and flies a more fuel-efficient path than the rest. Comparing the traditional APF method and the DRL methods, we can see that APF completes the tasks in simple environments, but it takes longer time and flies longer paths than DRL methods. When confronting with a complicated environment, APF fails to work while our method works well.

Table 3 Trajectory parameters of the four controllers

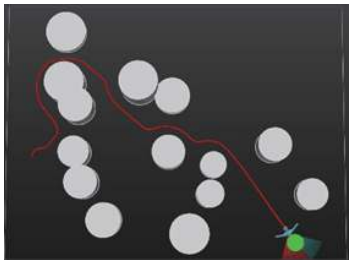
Controller	Environment with a threat density of 15%		Environment with a threat density of 25%		Environment with a threat density of 35%	
	Flight time/s	Path length/m	Flight time/s	Path length/m	Flight time/s	Path length/m
APF	21.0	420	24.1	482	Crash	Crash
DQN	20.0	400	Crash	Crash	Crash	Crash
DQN with DA1	19.0	380	20.1	402	Crash	Crash
DQN with DA2	18.8	376	19.7	394	23.1	462
DQN with DA3	15.8	316	17.3	346	22.1	442

(ii) Experiment II: testing in a dynamic environment with pop-up threats

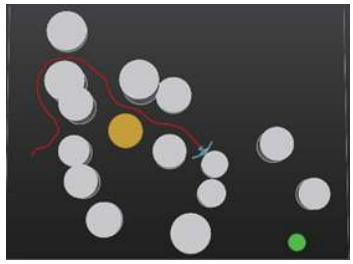
In this experiment, unknown threats are set to suddenly appear on the path of the UAV to test the control-

lers' capability of adapting pop-up circumstances. For the sake of comparison, three settings comprising no pop-up threat (the first column in Fig.10), one pop-up threat (the second column in Fig.10), and two pop-up threats (third column in Fig.10) are used. All the four controllers are tested in the three settings, respectively. The results thus obtained are shown in Fig.10 and Table 4, where the yellow cylinders represent the pop-up threats. In Fig.10 the rows represent controllers and from top to bottom are APF, DQN, DQN with DA1, DQN with DA2, DQN with DA3, respectively.

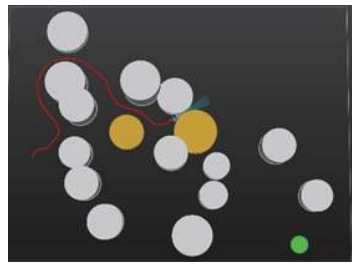
From the results, we can observe that all the five controllers enable the UAV to fly autonomously towards the target without any collision when there is no pop-up threat (as seen in Fig.10(a1)−Fig.10(e1)). As the first pop-up threat appears exactly on their straight ahead (as seen in Fig.10(a2)−Fig.10(e2)), all the UAVs make adaptive turns immediately. But unfortunately, only the controllers of APF, DQN with DA2, and DQN with DA3 successfully avoid the pop-up threat and eventually reach the target while the other two controllers crash during their path-changing process. When the second pop-up threat suddenly appears and completely blocks the original best way ((as seen in Fig.10(a3)−Fig.10(e3)), the UAVs further conduct a fine-tuning of the new path (after the first adjusting) immediately and autonomously to avoid colliding. But this time only the controller of DQN with DA3 makes it, the controller of APF and DQN with DA2 fails to find its new way. These results indicate our DA policy's outstanding performance of adapting to a dynamic environment with unknown sudden threats. Not only that, we can perceive directly from Table 4 that no matter whether there is a pop-up threat or not and how many pop-up threats are there, the controller of DQN with DA3 could provide the UAV more efficient trajectories than others and the trajectories usually cost shorter flight time and shorter flight distance. Meanwhile, we can conclude that both the traditional APF method and original DRLs have limited adaptability to poo-up threats, while our proposed method still works well without any re-training of the model.



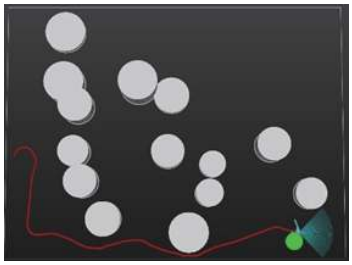
(a1) APF algorithm in environments with no pop-up threat



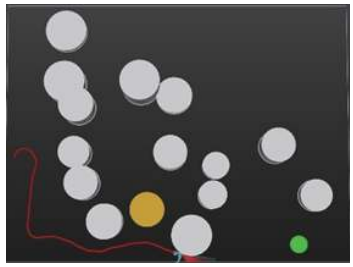
(a2) APF algorithm in environments with one pop-up threat



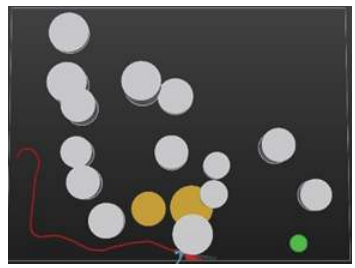
(a3) APF algorithm in environments with two pop-up threat



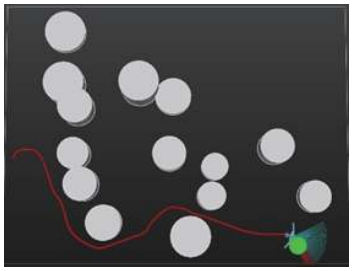
(b1) DQN algorithm in environments with no pop-up threat



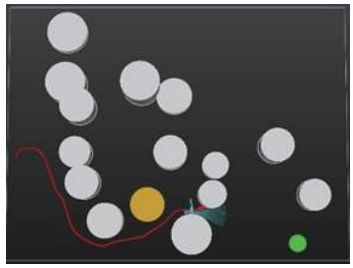
(b2) DQN algorithm in environments with one pop-up threat



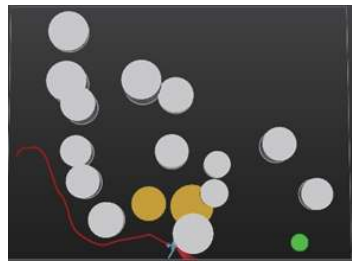
(b3) DQN algorithm in environments with two pop-up threat



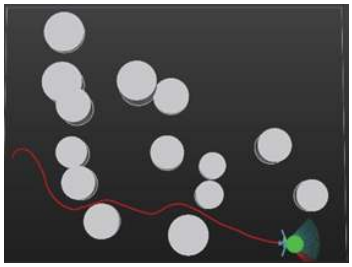
(c1) DQN with DA1 algorithm in environments with no pop-up threat



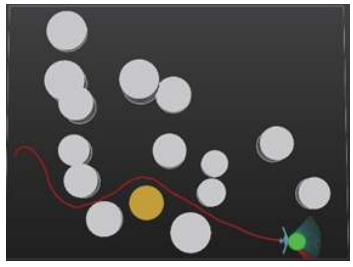
(c2) DQN with DA1 algorithm in environments with one pop-up threat



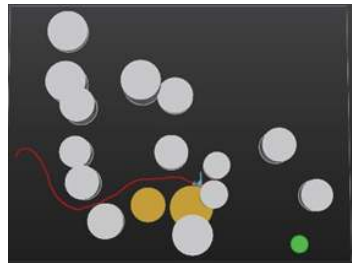
(c3) DQN with DA1 algorithm in environments with two pop-up threat



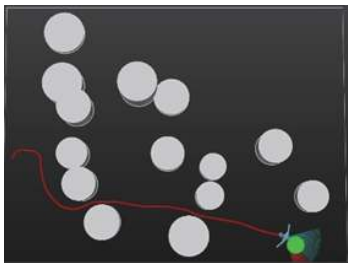
(d1) DQN with DA2 algorithm in environments with no pop-up threat



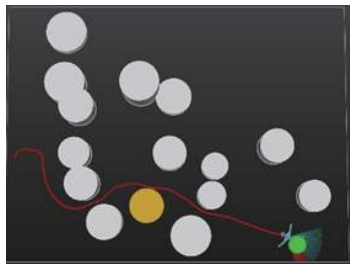
(d2) DQN with DA2 algorithm in environments with one pop-up threat



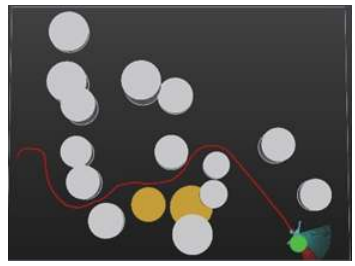
(d3) DQN with DA2 algorithm in environments with two pop-up threat



(e1) DQN with DA3 algorithm in environments with no pop-up threat



(e2) DQN with DA3 algorithm in environments with one pop-up threat



(e3) DQN with DA3 algorithm in environments with two pop-up threat

Fig. 10 Trajectories of the UAV with different controllers while facing pop-up threats

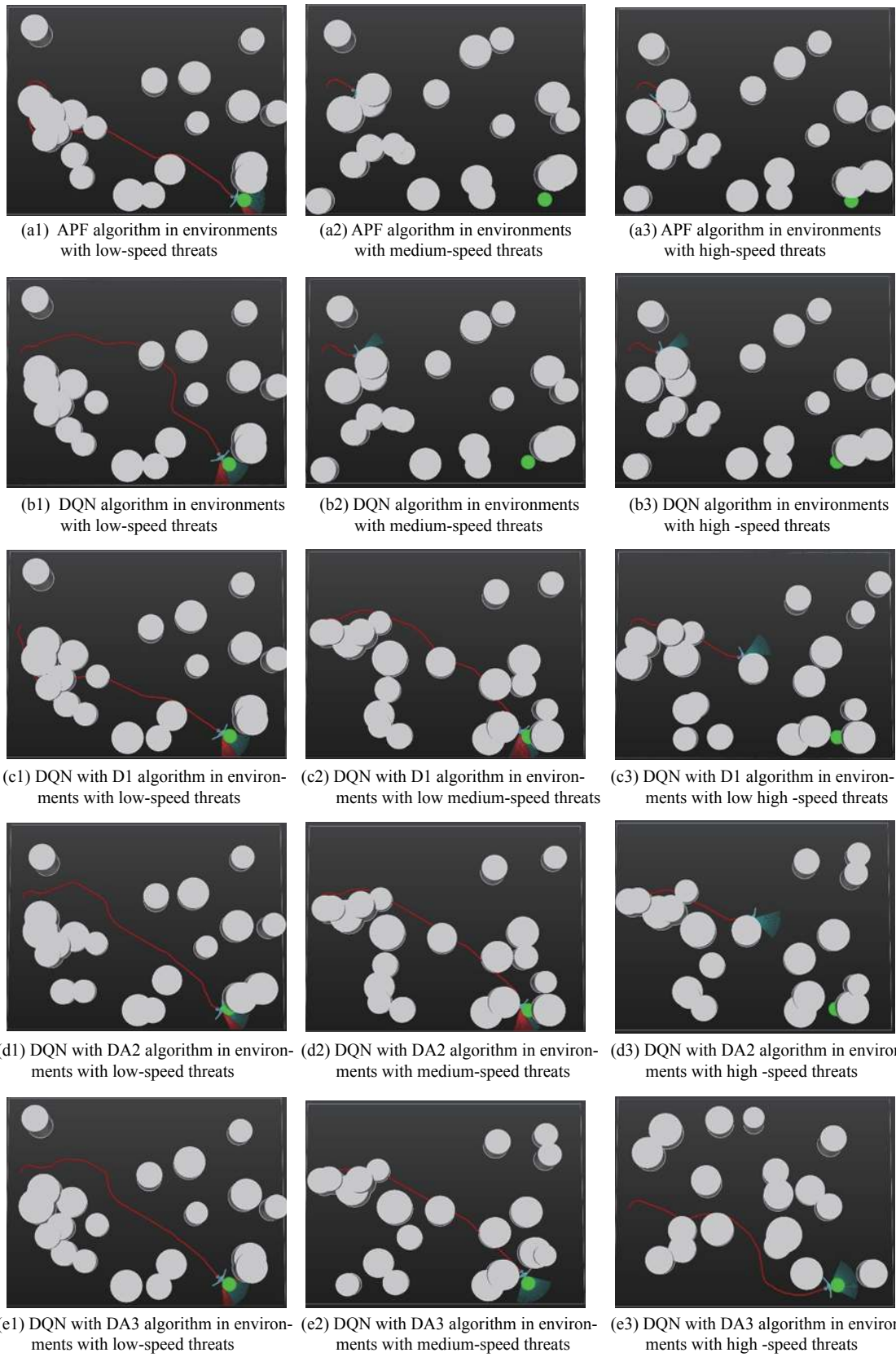
Table 4 Trajectory parameters of the four controllers

Controller	No pop-up threat		One pop-up threat		Two pop-up threats	
	Flight time/s	Path length/m	Flight time/s	Path length/m	Flight time/s	Path length/m
APF	26.2	524	Crash	Crash	Crash	Crash
DQN	23.0	460	Crash	Crash	Crash	Crash
DQN with DA1	21.0	420	Crash	Crash	Crash	Crash
DQN with DA2	19.1	382	20.1	402	Crash	Crash
DQN with DA3	18.9	378	19.6	392	22.1	442

(iii) Experiment III: testing in a dynamic environment with moving threats

In this experiment, all the threats are set to move at a certain speed and the controller has to be as flexible as enough to avoid the UAV colliding with these ubiquitous threats. For comparison, we initialize the threats with three kinds of speed, i.e., low-speed, medium-speed and high-speed. In the low-speed case, each threat is distributed with a randomly generated speed that obeys a uniform distribution $U(0, 5$ m/s), while the medium-speed and high-speed obey uniform distribution $U(5$ m/s, 10 m/s) and $U(10$ m/s,15 m/s), respectively. Here, the high-speed, medium-speed, and low-speed are defined relative to UAV's speed. The five controllers are used to guide the UAV from the same departure to the same target while crossing the dynamic environments. The results are shown in Fig. 11 and Table 5.

In Fig. 11, the rows represent controllers and from top to bottom are DQN, DQN with DA1, DQN with DA2, DQN with DA3, respectively. The columns represent different circumstances with no low-speed threats, with medium-speed threats, and with high-speed threats. As illustrated in Fig.11, in the low-speed case, all the five controllers successfully guide the UAV to the target without any collision (Fig.11(a1)−Fig.11(e1)). However, the DQN with DA3 provides the UAV a more efficient trajectory. When the speeds of the threats increase to the medium-speed range, the DQN-based controller and APF-based controller fail and the other three manage to avoid the moving threats and finish the mission (Fig.11(a2)−Fig.11(e2)). As the threat's speed goes to a high-speed range, only DQN with DA3 makes it and all the APF, DQN, DQN with DA1, and DQN with DA2 fail to adapt to the dynamic environment (Fig.11(a3)−Fig.11(e3)). As we can observe, our DA policy provides the UAV better adaptability to dynamic environments. However, we also have to admit that our policy could not adapt in all cases. In fact, in our tests, when the speeds of the treats are set too large, the UAV will collide with a threat because of its limited mobility. Also, Table 5 directly reflects the effect of the DA policy proposed in this paper from the perspective of flight time and path length.



(a1) APF algorithm in environments with low-speed threats
(a2) APF algorithm in environments with medium-speed threats
(a3) APF algorithm in environments with high-speed threats
(b1) DQN algorithm in environments with low-speed threats
(b2) DQN algorithm in environments with medium-speed threats
(b3) DQN algorithm in environments with high -speed threats
(c1) DQN with D1 algorithm in environments with low-speed threats
(c2) DQN with D1 algorithm in environments with low medium-speed threats
(c3) DQN with D1 algorithm in environments with low high -speed threats
(d1) DQN with DA2 algorithm in environments with low-speed threats
(d2) DQN with DA2 algorithm in environments with medium-speed threats
(d3) DQN with DA2 algorithm in environments with high -speed threats
(e1) DQN with DA3 algorithm in environments with low-speed threats
(e2) DQN with DA3 algorithm in environments with medium-speed threats
(e3) DQN with DA3 algorithm in environments with high -speed threats

Fig. 11 Trajectories of the UAV with different controllers while facing moving threats

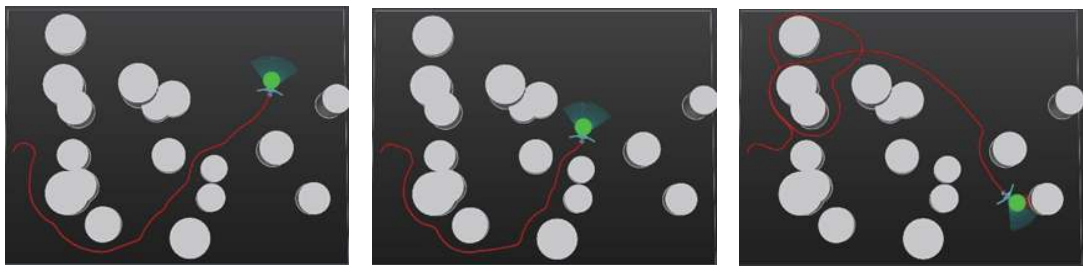
Table 5 Trajectory parameters of the four controllers

Controller	Low-speed threat		Medium-speed threat		High-speed threat	
	Flight time/s	Path length/m	Flight time/s	Path length/m	Flight time/s	Path length/m
APF	22.2	444	Crash	Crash	Crash	Crash
DQN	20.3	406	Crash	Crash	Crash	Crash
DQN with DA1	19.2	384	18.7	374	Crash	Crash
DQN with DA2	19.1	382	18.2	364	Crash	Crash
DQN with DA3	18.7	374	17.9	358	18.5	370

(iv) Experiment IV: testing in a dynamic environment with moving target

In this test, the target is set as movable, and the UAV is required to track a moving target. As comparison, the target is set to perform a uniform linear motion with a constant velocity of 10 m/s (lowspeed), 15 m/s (medium speed) and 20 m/s (highspeed), respectively. Once it hits the boundary, the target obtains a new reflective direction. In Fig. 12, we present some screenshots of the results of the experiments. The rows represent controllers and from top to bottom are APF, DQN, DQN with DA1, DQN with DA2, DQN with DA3, respectively. The columns represent different circumstances with a low-speed target, a medium-speed target, and a high-speed target.

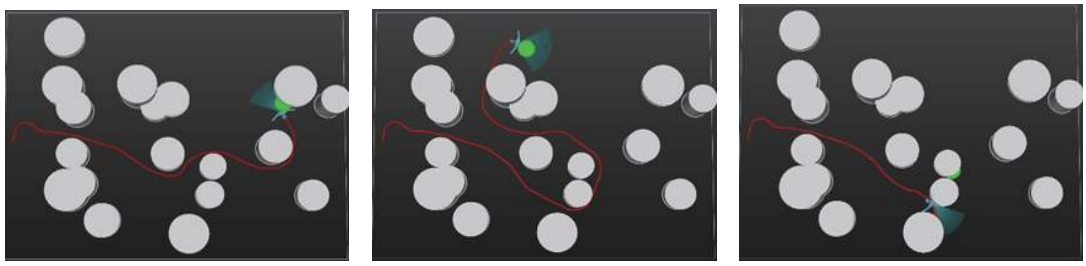
From the screenshots in Fig.12, we can see that all five controllers work well when the target moves at a relatively low-speed. In the process of tracking, the UAV can normally avoid treats until finally reach the moving target (Fig.12(a1)−Fig.12(e1)). When the target moves at a medium speed of 15 m/s, DQN with DA2 and DQN with DA3 show greater advantages because they take less time to finally reach the moving target, but APF, DQN with DA1, and DQN also complete the mission despite taking more time (Fig.12(a2)−Fig.12(e2)). It is a real challenge for the UAV when the target moves at a high-speed (as fast as the UAV), because both the DQN and DQN with DA1 fail and crashs (Fig.12(b3)−Fig.12(c3)).



(a1) APF algorithm in environments with a low-speed target

(a2) APF algorithm in environments with a medium -speed target

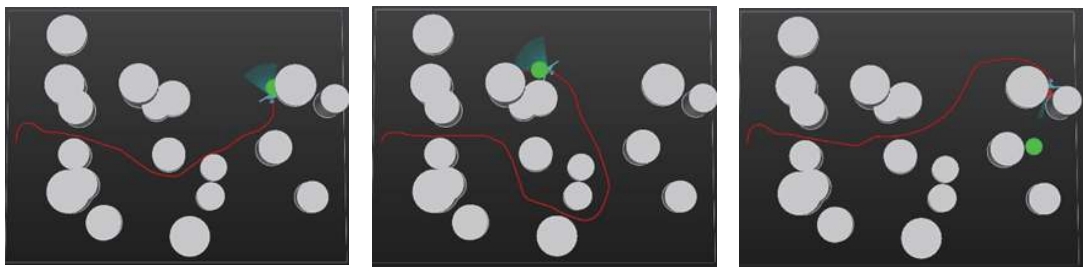
(a3) APF algorithm in environments with a high-speed target



(b1) DQN algorithm in environments with a low-speed target

(b2) DQN algorithm in environments with a medium -speed target

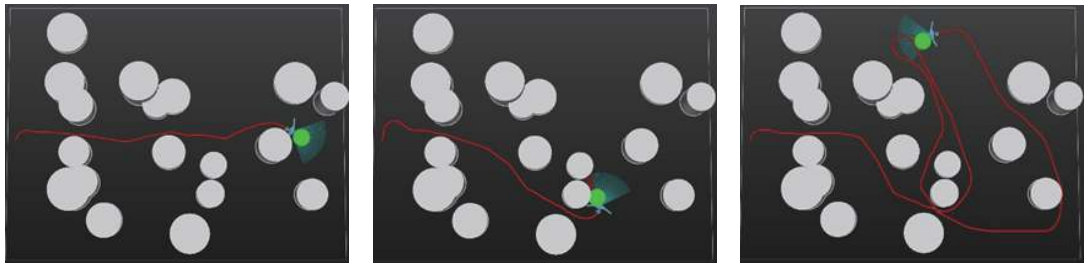
(b3) DQN algorithm in environments with a high -speed target



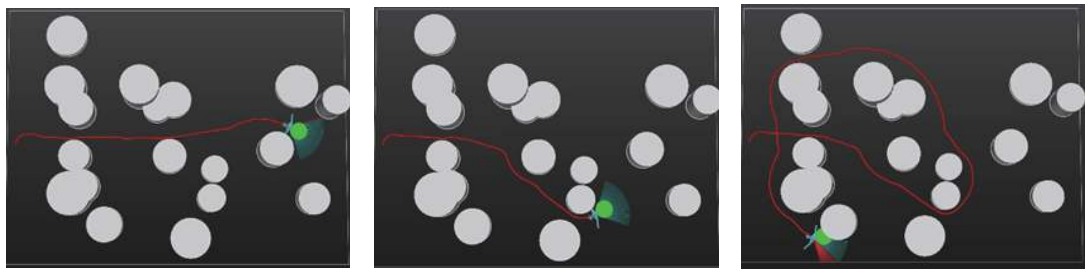
(c1) DQN with DA1 algorithm in environments with a low-speed target

(c2) DQN with DA1 algorithm in environments with a medium -speed target

(c3) DQN with DA1 algorithm in environments with a high -speed target



(d1) DQN with DA2 algorithm in environments with a low-speed target (d2) DQN with DA2 algorithm in environments with a medium -speed target (d3) DQN with DA2 algorithm in environments with a high -speed target



(e1) DQN with DA3 algorithm in environments with a low-speed target (e2) DQN with DA3 algorithm in environments with a medium -speed target (e3) DQN with DA3 algorithm in environments with a high -speed target

Fig. 12 Trajectories of the UAV with different controllers while facing the moving target

Fortunately, APF, DQN with DA2, and DQN with DA3 eventually catch up and hit the target, although it takes a lot of detours ((a3), (d3), (e3)). This experiment turns the target search mission (trained one) into a target track mission (test one), but as we can observe, the UAV adapts it very well without any retraining process. The same conclusion can be obtained from the trajectory parameters of the four controllers in Table 6. We can also find that the traditional APF method works well in all three circumstances, even when the target moves at a high-speed. This result illustrates a fact that the APF method is not sensitive to the target speed, it even provides better adaptability than DQN and DQN with DA1.

Table 6 Trajectory parameters of the four controllers

Controller	Low-speed target (10 m/s)		Medium-speed target (15 m/s)		High-speed target (20 m/s)	
	Flight time/s	Path length/m	Flight time/s	Path length/m	Flight time/s	Path length/m
APF	24.5	490	20.8	416	48.1	962
DQN	22.0	440	30.8	616	Crash	Crash
DQN with DA1	19.3	386	25.6	512	Crash	Crash
DQN with DA2	17.3	346	16.0	320	61.9	1 238
DQN with DA3	16.9	338	14.6	292	43.9	878

From all the four exploiting tests, we can conclude that the DRL is a powerful method for providing the UAV with AMC capabilities in unknown dynamic environments and the DA policy can further enhance the efficiency of the traditional DRL algorithms.

6. Conclusions

This paper presents a DRL-based end-to-end motion controller for UAVs to autonomously fly across dynamic unknown environments. The framework uses four modules working together to maintain a stable motion-control of the UAV.

Two DA-based learning strategies are introduced into traditional DRL to construct novel DRL algorithms for motion planning. An improved LGVF algorithm is used to handle the trajectory-tracking problem and turn high-level actions into guidance-control commands for the UAV. A general UAV mission environment is built in this study for the controller's training and testing.

The training experiments demonstrate that the algorithms in this paper make great contributions to the performance improvement of UAVs, and the testing experiments show that our DRL-based controller provides the UAV with good adaptability to different dynamic environments.

In future research, we plan to extend the autonomous

motion-control problem to a 3D space, that is, we intend to provide the UAV with altitude control. Moreover, we intend to control the UAV in a continuous action space and use policy-based DRL algorithms to develop intelligent controllers.

References

[1] STEVENS R C, SADJADI F A, et al. Small unmanned aerial vehicle (UAV) real-time intelligence, surveillance and reconnaissance (ISR) using onboard pre-processing. Proc. of the SPIE, 2008: 6967.

[2] DARRAH M, NILAND W, STOLARIK B, et al. UAV cooperative task assignments for a SEAD mission using genetic algorithms. Proc. of AIAA Guidance, Navigation, and Control Conference and Exhibit, 2006. DOI: 10.2514/6.2006-6456.

[3] TOMIC T, SCHMID K, LUTZ P, et al. Toward a fully autonomous UAV: research platform for indoor and outdoor urban search and rescue. IEEE Robotics & Automation Magazine, 2012, 19(3): 46–56.

[4] SHAKHATREH H, SAWALMEH A, AL-FUQAHA A, et al. Unmanned aerial vehicles (UAVs): a survey on civil applications and key research challenges. IEEE Access, 2019, 7: 48572–48634.

[5] WAN K F, GAO X G, HU Z J, et al. A RDA-based deep reinforcement learning approach for autonomous motion planning of UAV in dynamic unknown environments. Proc. of the 4th International Conference on Control Engineering and Artificial Intelligence, 2020. DOI: 10.26914/c.cnkihy.2020.002792.

[6] IMANBERDIYEV N, FU C, KAYACAN E, et al. Autonomous navigation of UAV by using real-time model-based reinforcement learning. Proc. of the 14th International Conference on Control, Automation, Robotics and Vision, 2016: 1−6.

[7] WAN K F, GAO X G, HU Z J, et al. Robust motion control for UAV in dynamic uncertain environments using deep reinforcement learning. Remote Sensing, 2020, 12(4): 640–661.

[8] CHEE K Y, ZHONG Z W. Control, navigation and collision avoidance for an unmanned aerial vehicle. Sensors and Actuators A: Physical, 2013, 190(1): 66–76.

[9] PANAGOU D. A distributed feedback motion planning protocol for multiple unicycle agents of different classes. IEEE Trans. on Automatic Control, 2016, 62(3): 1178–1193.

[10] ISRAELSEN J, BEALL M, BAREISS D, et al. Automatic collision avoidance for manually tele-operated unmanned aerial vehicles. Proc. of the IEEE International Conference on Robotics and Automation, 2014: 6638−6643.

[11] YANG X, DING M Y, ZHOU C. Fast marine route planning for UAV using improved sparse A* algorithm. Proc. of the 4th International Conference on Genetic and Evolutionary Computing, 2011: 190−193.

[12] DUAN H B, PEI L. UAV path planning. Berlin: Springer, 2014.

[13] WAN K F, GAO X G, LI B. Using approximate dynamic programming for multi-ESM scheduling to track ground moving targets. Journal of Systems Engineering and Electronics, 2018, 29(1): 74–85.

[14] YANG Q M, ZHANG J D, SHI G Q. Modeling of UAV path planning based on IMM under POMDP framework. Journal of Systems Engineering and Electronics, 2019, 30(3): 545–554.

[15] DONG Z N, ZHANG R L, CHEN Z J, et al. Study on UAV path planning approach based on fuzzy virtual force. Chinese Journal of Aeronautics, 2010, 23(3): 341–350.

[16] BREZOESCU A, ESPINOZA T, CASTILLO P, et al. Adaptive trajectory following for a fixed-wing UAV in presence of crosswind. Journal of Intelligent & Robotic Systems, 2013, 69(1/4): 257–271.

[17] FADLULLAH Z M, TAKAISHI D, NISHIYAMA H, et al. A dynamic trajectory control algorithm for improving the communication throughput and delay in UAV-aided networks. IEEE Network, 2016, 30(1): 100–105.

[18] GEE T, JAMES J, MARK W, et al. Lidar guided stereo simultaneous localization and mapping (SLAM) for UAV outdoor 3-D scene reconstruction. Proc. of the International Conference on Image and Vision Computing, 2016: 1−6.

[19] PANCHPOR A A, SHUE S, CONRAD JM, et al. A survey of methods for mobile robot localization and mapping in dynamic indoor environments. Proc. of the Conference on Signal Processing & Communication Engineering Systems, 2018: 138−144.

[20] ALTAN A, BAYRAKTAR K, HACIOGLU R, et al. Simultaneous localization and mapping of mines with unmanned aerial vehicle. Proc. of the 24th Signal Processing and Communication Application Conference, 2016: 1433−1436.

[21] FU C H, OLIVARES M, SUAREZ R, et al. Monocular visual-inertial slam-based collision avoidance strategy for fail-safe UAV using fuzzy logic controllers. The International Journal of Robotics Research, 2014, 73(4): 513–533.

[22] SUTTON R S, BARTO A G. Reinforcement learning: an introduction. 2nd ed. Cambridge, US: MIT Press, 2017.

[23] KERSANDT K. Deep reinforcement learning as control method for autonomous UAV. Barcelona, Spain: University of Catalonia, 2017

[24] VOLODYMYR M, KORAY K, David S, et al. Human-level control through deep reinforcement learning. Nature, 2015, 518(7540): 529–533.

[25] HASSELT H V, GUEZ A, SILVER D. Deep reinforcement learning with double Q-learning. Proc. of the 30th AAAI Conference on Artificial Intelligence, 2016. arXiv: 1509.06461.

[26] WANG Z Y, SCHAUL T, MATTEO H, et al. Dueling network architectures for deep reinforcement learning. Proc. of the 33rd International Conference on Machine Learning, 2016, 48: 1995–2003.

[27] FAN D D, THEODOROU E, REEDER J, et al. Model-based stochastic search for large scale optimization of multi-agent UAV swarms. Proc. of the IEEE Symposium Series on Computational Intelligence, 2017. arXiv: 1803.01106.

[28] WANG C, WANG J, SHEN Y, et al. Autonomous navigation of UAVs in large-scale complex environments: a deep reinforcement learning approach. IEEE Trans. on Vehicular Technology, 2019, 68(3): 2124–2136.

[29] ALEJANDRO R R, CARLOS S, HRIDAY B, et al. A deep reinforcement learning strategy for UAV autonomous landing on a moving platform. Journal of Intelligent & Robotic Systems, 2018, 93: 351–366.

[30] KOCH W, MANCUSO R, WEST R, et al. Reinforcement learning for uav attitude control. ACM Trans. on Cyber-Physical Systems, 2019, 3(2): 1–21.

[31] LILLICRAP T T, HUNT J J, PRITZEL A, et al. Continuous control with deep reinforcement learning. arXiv, 2015. arXiv:

1509.02971.
[32] SCHULMAN J, LEVINE S, MORITZ P, et al. Trust region policy optimization. Proc. of the 32nd International Conference on Machine Learning, 2015. arXiv:1502.05477.
[33] SCHULMAN J, WOLSKI F, DHARIWAL P, et al. Proximal policy optimization algorithms. arXiv, 2017. arXiv:1707.06347.
[34] HERNANDEZ E G, ARANDA E. Convergence and collision avoidance in formation control: a survey of the artificial potential functions approach. ALKHATEEB F, MAGHAYREH A E, DOUSH A I, ed. Multi-agent systems-modeling, control, programming, simulations and applications. InTechOpen, 2011: 103−126.
[35] LUCA A D, ORIOLO G. Local incremental planning for nonholonomic mobile robots. Proc. of the IEEE International Conference on Robotics and Automation, 1994. DOI: 10.1109/ROBOT.1994.351003.
[36] LOIZOU S G, KYRIAKOPOULOS K J. Navigation of multiple kinematically constrained robots. IEEE Trans. on Robotics, 2008, 24(1): 221–231.
[37] KIM J O, KHOSLA P K. Real-time obstacle avoidance using harmonic potential functions. IEEE Trans. on Robotics and Automation, 1992, 8(3): 338–349.
[38] LAWRENCE D A, FREW E W, PISANO W. Lyapunov guidance vector fields for autonomous unmanned aircraft flight control. Journal of Guidance, Control, and Dynamics, 2012, 31(5): 1220–1229.
[39] FREW E W, LAWRENCE D A, MORRIS S. Coordinated standoff tracking of moving targets using Lyapunov guidance vector. Journal of Guidance, Control, and Dynamics, 2008, 31(2): 290–306.
[40] PARK S. Circling over a target with relative side bearing. Journal of Guidance Control & Dynamics, 2016, 39(6): 1–7.
[41] JUNG W, LIM S, LEE D, et al. Unmanned aircraft vector field path following with arrival angle control. Journal of Intelligent & Robotic Systems, 2016, 84: 311–325.
[42] NAMHOON C, YOUDAN K, SANGHYUK P. Three-dimensional nonlinear differential geometric path-following guidance law. Journal of Guidance, Control, and Dynamics, 2015, 38(12). DOI: https://doi.org/10.2514/1.G001060.
[43] SAMUELSON W, ZECKHAUSER R. Status quo bias in decision making. Journal of Risk and Uncertainty, 1988, 1: 7–59.
[44] DIXON C. Controlled mobility of unmanned aircraft chains to optimize network capacity in realistic communication environments. Colorado, US: University of Colorado, 2010.
[45] QUINTERO S, COLLINS G, HESPANHA J. Flocking with fixed-wing UAVs for distributed sensing: a stochastic optimal control approach. Proc. of the American Control Conference, 2013. DOI: 10.1109/ACC.2013.6580133.
[46] WU G F, GAO X G, FU X W. Mobility control of unmanned aerial vehicle as communication relay in airborne multi-user systems. Chinese Journal of Aeronautics, 2019, 32(6): 1520–1529.
[47] HORGAN D, QUAN J, BUDDEN D, et al. Distributed prioritized experience replay. arXiv, 2018. arXiv:1803.00933.
[48] IVANOV S, YAKONOV A. Modern deep reinforcement learning algorithms. arXiv preprint, arXiv:1906.10025, 2019.
[49] HU Z J, WAN K F, GAO X G, et al. A dynamic adjusting reward function method for deep reinforcement learning with adjustable parameters. Mathematical Problems in Engineering, 2019: 7619483.

Biographies

WAN Kaifang was born in 1987. He received his B.E. degree in detection homing and control technology from Northwestern Polytechnical University (NWPU), Xi'an, in 2010. He received his Ph.D. degree in system engineering in 2016 from NWPU. Now he is an assistant researcher of the Key Laboratory of Aerospace Information Perception and Photoelectric Control of the Ministry of Education, NWPU. His current research interests include sensor management application, multi-agent theory, approximate dynamic programming, and reinforcement learning theory.
E-mail: wankaifang@nwpu.edu.cn

LI Bo was born in 1978. He received his B.S. degree in electronic information technology and his M.S. and Ph.D. degree in systems engineering from Northwestern Polytechnical University (NWPU), Xi 'an, in 2000, 2003, and 2008, respectively. He was a postdoctoral fellow with NWPU from 2008 to 2010. He is currently an associate professor with the School of Electronics and Information, NWPU. His current research interests include intelligent command and control, deep reinforcement learning, and uncertain information processing.
E-mail: Libo803@nwpu.edu.cn

GAO Xiaoguang was born in 1957. She received her B.E. degree in detection homing and control technology from Northwestern Polytechnical University (NWPU) in 1982. She completed her master degree in system engineering from NWPU in 1986. She received her Ph.D. degree from NWPU in 1989. She is currently a professor and the head of the Key Laboratory of Aerospace Information Perception and Photoelectric Control of the Ministry of Education, NWPU. Her research interests are machine learning theory, Bayesian network theory, and multi-agent control application.
E-mail: cxg2012@nwpu.edu.cn

HU Zijian was born in 1996. He received his B.E. degree in detection guidance and control technology from Northwestern Polytechnical University (NWPU), Xi 'an in 2018. He is currently pursuing his Ph.D. degree in the College of Electronic and Information, NWPU. His current research interests include reinforcement learning theory and the applications of reinforcement learning in UAV control and fire control systems.
E-mail: huzijian@mail.nwpu.edu.cn

YANG Zhipeng was born in 1995. He received his B.S. degree in electrical engineering and automation from Hubei University of Technology, Wuhan, in 2016. He is currently a postgraduate student with the School of Electronics and Information, Northwestern Polytechnical University. His current research interest is intelligent maneuver decision for unmanned systems.
E-mail: yzp@mail.nwpu.edu.cn