Journal of Intelligent & Robotic Systems
https://doi.org/10.1007/s10846-019-01073-3

Towards Real-Time Path Planning through Deep Reinforcement Learning for a UAV in Dynamic Environments

Chao Yan[1] · Xiaojia Xiang[1] · Chang Wang[1]

Received: 9 January 2019 / Accepted: 25 July 2019

Abstract
Path planning remains a challenge for Unmanned Aerial Vehicles (UAVs) in dynamic environments with potential threats. In this paper, we have proposed a Deep Reinforcement Learning (DRL) approach for UAV path planning based on the global situation information. We have chosen the STAGE Scenario software to provide the simulation environment where a situation assessment model is developed with consideration of the UAV survival probability under enemy radar detection and missile attack. We have employed the dueling double deep Q-networks (D3QN) algorithm that takes a set of situation maps as input to approximate the Q-values corresponding to all candidate actions. In addition, the ε-greedy strategy is combined with heuristic search rules to select an action. We have demonstrated the performance of the proposed method under both static and dynamic task settings.

Keywords Unmanned aerial vehicle (UAV) · Path planning · Reinforcement learning · Deep Q-network · STAGE scenario

1 Introduction

Unmanned Aerial Vehicles (UAVs) have been widely used in many applications including landscape mapping, search and rescue, surveillance, attacking terrorists, et al. [1–3]. Among various capabilities of the UAVs, autonomous path planning is fundamental to guarantee task accomplishment [4]. Although there has been significant progress in making UAV's operation increasingly autonomous [5], path planning in dynamic environments is still challenging for a UAV because there leaves little time for it to avoid unexpected flying obstacles such as birds or other aircrafts [6].

In this paper, we deal with a more challenging problem that considers UAV path planning with potential enemy threats, e.g., in anti-terrorist tasks. Specifically, we assume that enemy radars can detect the UAV within a certain range, and surface-to-air missiles (SAM) can influence the UAV's survival probability with regard to the distance, so that the UAV must learn to solve the task while ensuring its safety. This problem is difficult because both the number and the location of the threats are not known to the UAV before the task is given. The UAV has to develop an adaptive policy to react to environmental dynamics.

Reinforcement learning (RL) enables an agent to autonomously learn an optimal policy to maximize cumulative rewards through trial-and-error interactions with its environment [7]. Model-free RL methods have become popular in the field of path planning. The Q-learning algorithm [8] has been widely used for path planning. For example, an adaptive and random exploration (ARE) approach [9] enables UAV navigation while avoiding obstacles. A path planning method combines an improved version of the Q-learning algorithm with heuristic searching rules for mobile robots in a dynamic environment [10]. In [11], a distinct derived learning method based on Q-learning and cyclic error correction has been proved effective for mobile robot navigation. In our previous work, we have used the extended classifier systems (XCS) that combines Q-learning and genetic algorithms (GA) to enable affordance-based navigation for a humanoid robot NAO in changing environments [12]. We have also improved the performance of Q-learning algorithm with an action selection strategy and a Q-function initialization method, which has been applied to UAV path planning in an antagonistic environment [13]. However, the Q-learning algorithm stores all state-action pairs in a Q-table, and it has limitation to deal with high-dimensional state representation that needs to be handled in this paper.

✉ Xiaojia Xiang
xiangxiaojia@nudt.edu.cn

Chao Yan
yanchao17@nudt.edu.cn

Chang Wang
wangchang07@nudt.edu.cn

[1] College of Intelligence Science and Technology, National University of Defense Technology, Changsha 410073, China

Published online: 07 September 2019

Deep Neural Network (DNN) has been used to approximate the Q-function, and Deep reinforcement Learning (DRL) [14] algorithms such as the Deep Q- Networks (DQN) [15, 16] has been proposed. In this way, the high-dimensional state of raw images can be taken as the input of the DQN, and the output is an action to be selected by the agent. For example, using only raw grayscale images and the scores in the game as input, the agent trained with the DQN algorithm and the experience replay technique can achieve human-level performance in playing a number of Atari games [16]. This method has been widely used in robotic applications.

Not much related work has been done for UAV path planning taking the DRL approach. An effective lazy training method can improve the training efficiency and stability of deep Q-network [17]. The Neural Q-Learning algorithm (NQL) and the extended deep Q-networks take the BP network as a Q-function approximator, and have been shown to gain better performance and higher success rate [18]. However, the above methods need to be further tested in more complex environments rather than the simply maze environment or pure numerical simulations. In our previous work [6], we have proposed a framework that integrates a saliency-based obstacle detection module using deep convolutional neural networks (CNN) along with an RL module using an actor-critic structure for a UAV to avoid flying obstacles. However, the proposed method is computational expensive that it can hardly achieve real-time performance.

In this paper, we assume that the global situational data is available for the UAV, provided by a C4ISR system or other UAVs. The main contributions are as follows:

- We have proposed a fast situation assessment model that translates the global environmental states into sequential situation maps to represent dynamic enemy threats.
- We have proposed a deep reinforcement learning framework for UAV path planning. This framework employs the dueling double deep Q-networks to predict Q-values of all candidate actions, which are used by the action selection policy that combines the ε-greedy strategy with heuristic search rules.

The paper is organized as follows. Section 2 proposes a DRL-based system framework for UAV path planning, and Section 3 introduces the details of the framework. Experiment results are discussed in Section 4. Finally, Section 5 concludes the paper and outlines our plans for future work.

2 System Framework

In this section, we propose a novel DRL-based framework to solve the path planning problem for a UAV in dynamic environments.

The system framework consists of an Environment and an Agent (see Fig. 1). We develop the Environment in the STAGE Scenario (see Section 3.1 for details). The Agent consists of three modules: the situation assessment module, the Dueling Double Deep Q-networks (D3QN) module, and the action selection module. Specifically, the *Stage Interface* plug-in reads the situation data generated by the STAGE Scenario through the shared memory, and then it sends the data to the Agent via the UDP socket. The situation assessment module uses the received data to build a situation map. Then, the D3QN module uses the stacked situation map to predict the Q-values corresponding to all valid actions. The action selection module selects an action according to the predicted Q-values, and the selected action number is sent to *Stage Interface*. Finally, the preset path data of the UAV in shared memory will be updated, and then the UAV will fly to the new position using the selected action.

3 DRL-Based Path Planning

In this section, we introduce the details of the proposed framework. Section 3.1 briefly discusses the Environment, i.e., the STAGE Scenario for data generation and the *Stage Interfere* plug-in for data acquisition. Section 3.2 discusses the situation assessment module. The Markov Decision Process (MDP) model is given in Section 3.3. The D3QN module is illustrated in Section 3.4 and the action selection policy is described in Section 3.5. Finally, we summarize the workflow of the proposed approach in Section 3.6.

3.1 Environment

Reinforcement learning algorithms typically require many episodes of trial-and-error training through the interaction between the agent and the environment [7]. The STAGE Scenario [19] is an ideal software tool for developing the environment for a UAV along with enemy entities (see Figs. 1 and 5). It has rich tactical database and provides authoritative and classical built-in models and modules for behavioral simulation as well as tactical/campaign simulation of battlefield entities such as radars and missiles. It also supports users to implement secondary developments.

We select necessary attributes to represent the situation data, e.g., the UAV location, the target location, and the enemy location et al. We develop the *Stage Interface* plug-in to acquire the situation data in real time, and implement the functions using external commands to drive the simulated entities. This plug-in interacts with the simulation data generated in the STAGE Scenario by the shared memory. When the simulation is running, the *Stage Interface* plug-in can access and modify the data in the memory through the interface.

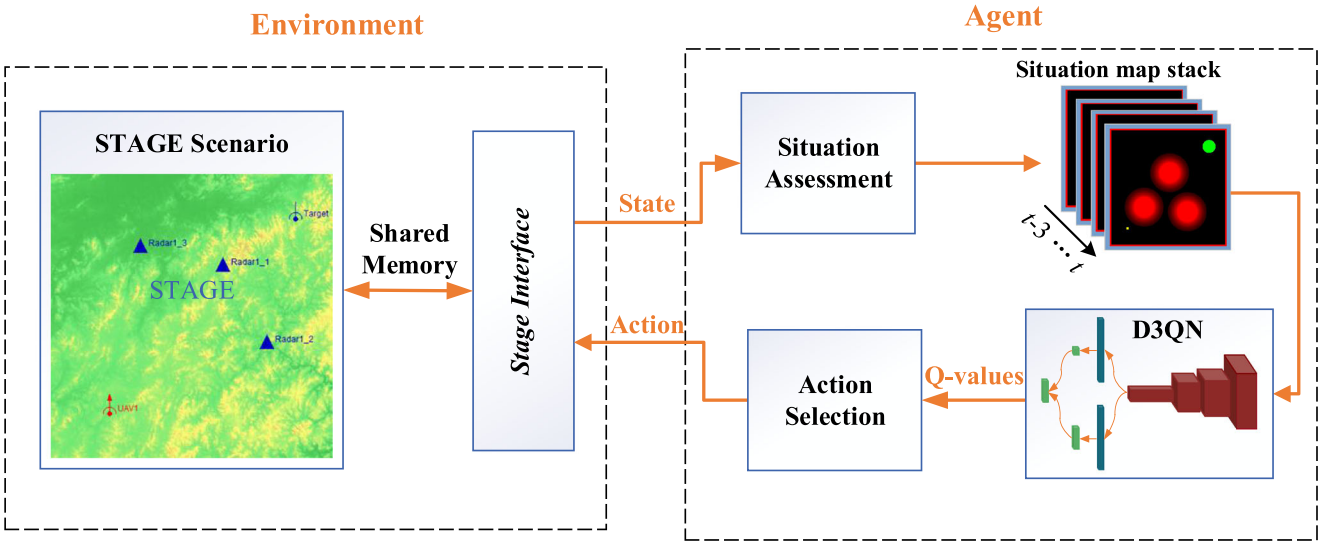


Fig. 1 The system framework of UAV path planning through deep reinforcement learning

3.2 Situation Assessment

Situation assessment (SA) is the basis of decision-making and path planning [20]. In this paper, the SA module is the front-end function of the UAV path planning framework (see Fig. 1).

3.2.1 Situation Assessment Model

In the literature, various threat assessment models have been proposed in the air combat scenario [21–23]. The distance between the UAV and the enemy entities is typically used for threat assessment. In this paper, we construct the situation assessment model using the distance between the UAV and the center of an enemy defense unit consisting of an early warning radar and a Surface-to-Air Missile (SAM). Specifically, once the UAV is detected by radar, it will be destroyed by a SAM with a certain probability which is calculated with regard to their distance. This calculation considers the maximum detection range of the radar, the maximum radius of missile killing zone, and the maximum range of no-escape zone. As a result, the threat level of each defense unit to the UAV is formulated as follows:

$$T_p = \begin{cases} 0 & D \geq R_{R\max} \\ 0.5\, e^{-\frac{D-R_{M\max}}{R_{R\max}-R_{M\max}}} & R_{M\max} \leq D < R_{R\max} \\ 2^{-\frac{D-R_{Mk\max}}{R_{M\max}-R_{Mk\max}}} & R_{Mk\max} \leq D < R_{M\max} \\ 1 & D < R_{Mk\max} \end{cases} \quad (1)$$

where D is the distance between the UAV and the defense unit; $R_{R\ \max}$ denotes the maximum detection range of the radar; $R_{M\ \max}$ represents the maximum radius of the missile killing zone; $R_{Mk\ \max}$ indicates the maximum range of the no-escape zone.

The threat level T_p indicates the probability of being hit by a SAM. We assume that the hit probabilities of different missiles are independent. Therefore, the overall threat of all defense units to the UAV can be calculated by the following equation [24]:

$$T_s = 1 - \prod_{i=1}^{k} \left(1 - T_p^i\right) \quad (2)$$

where k denotes the number of defense units in the task area of the UAV; T_p^i represents the damaged probability of the UAV passing the i-th defense unit.

3.2.2 Situation Map

A situation map shows the overview of a tactical situation at a particular time and it helps make tactical decisions [25]. However, it is a challenging task to build a situation map of the task area for the UAV. In this paper, we use an RGB color model to represent the situation analysis results of the proposed SA model, and then we build a situation map for the UAV. The main steps for creating a situation map are as follows:

- Step 1: Acquire the situation data from the STAGE Scenario via the *Stage Interface* plug-in;
- Step 2: Calculate the synthetic threat value T_s for each position using the situation assessment model;
- Step 3: Convert each synthetic threat value T_s to the pixel value of a color channel to build a situation map according to the following equation:

$$C = C_{\min} + \frac{T_s - T_{\min}}{T_{\max} - T_{\min}} \left(C_{\max} - C_{\min}\right) \quad (3)$$

where $C_{\max}$ and $C_{\min}$ denote the maximum and minimum pixel value of a color channel, respectively; $T_{\max}$ and $T_{\min}$ represent the maximum and minimum synthetic threat values based on the situation data.

In this paper, there are one reconnaissance target (i.e., the UAV's destination) and three defense entities in the task area of the UAV. We set the maximum pixel value $C_{\max} = 255$, the

minimum pixel value $C_{\min} = 0$, and then we convert the threat value to the pixel value of the Red channel. A resulting situation map is shown in Fig. 2.

In Fig. 2, The small yellow rectangle denotes the location of the UAV, the green circular area represents the target area, the radius of which is determined by the UAV's detection range. The task succeeds if the UAV arrives at the target area. The red circular area and the red bounding rectangle indicate the threat area and task area boundary, respectively. The darker the color, the greater the probability of being shot down by missiles. We will use the situation map as input for the path planning module. The details will be given in Section 3.4.

3.3 MDP Model

The problem of UAV path planning can be formulated as an MDP model. The three elements of this model are described as follows.

3.3.1 State

The environmental state of our path planning MDP model is represented by the situation map constructed via the situation assessment model proposed in Section 3.2.

3.3.2 Action

In order to reduce the computational burden of UAV path planning, we divide the task area into grids, and assume that the UAV can move freely within the 8 grids surrounding it. In this way, we design 8 basic actions. As shown in Fig. 3, action number 0, 1 … 7 denote the North, North-East … North-West, respectively.

3.3.3 Reward

In reinforcement learning, the reward function affects the training effective directly. Therefore, designing a reasonable reward function is very important. The reward function we constructed for the MDP model of UAV path planning is shown in Table 1.

In Table 1, the reward value r_T is related to the synthetic threat value calculated according to Eq. (2). If the threat value T_s exceeds the threshold value T_σ, it can be considered that the UAV is bound to be shot down by SAM. Thus, the expression of r_T is defined as follows:

$$r_T = \begin{cases} -10T_s - 0.5 & 0 \le T_s < T_\sigma \\ -50 & T_\sigma \le T_s \le 1 \end{cases} \tag{4}$$

As can be seen from Table 1, when the UAV (i) reaches the target area, a big positive reward (+200) is obtained; on the other hand, a big negative reward (−50) is arranged if the UAV (ii) enters the threat area and is destroyed by missiles, (iii) flies over its maximum flight range or, (iv) flies out of the map boundary. When the UAV satisfies one of the above four conditions, an episode will be terminated and the next episode will be started in the training process. Otherwise, a small negative reward (−0.5) is designed to motivate the UAV to arrive at the target position as soon as possible.

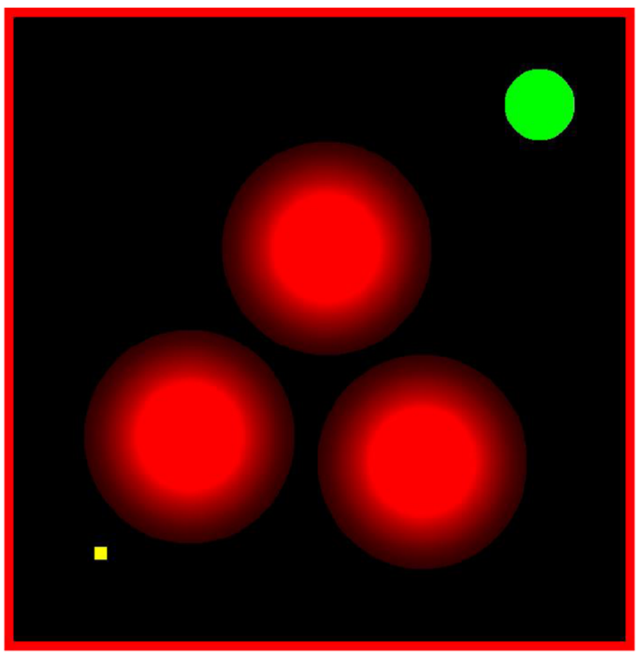


Fig. 2 An example of the situation map created using the situation assessment model

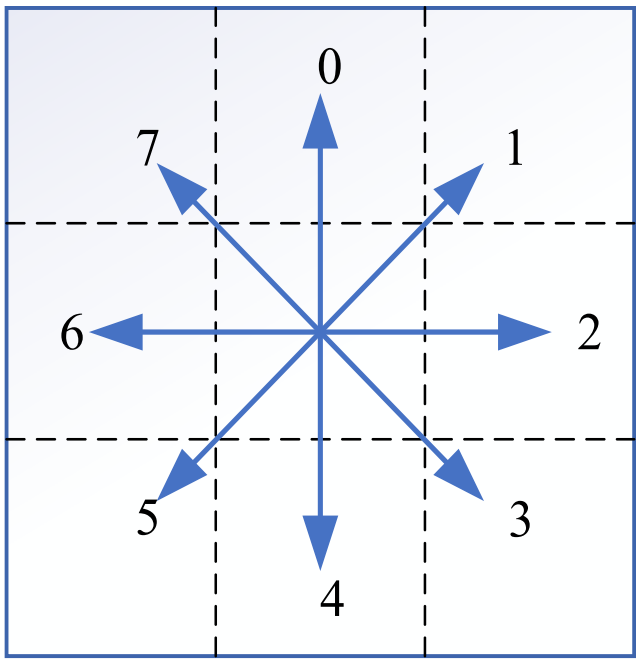


Fig. 3 8 discrete actions. Number 0 denotes the North, 1 represents the North-East, 2 indicates the East, and so on

Table 1 The setting of reward function

UAV State	Reward Value
Within the target area	+200
Within the threat area	r_T
Beyond the maximum range	−50
Outside the map boundary	−50
Otherwise	−0.5

3.4 Q-Function Approximation Using D3QN

As deep Q-networks (DQN) [16] can deal with high dimensional states of raw images, DQN-based models have been widely used for obstacle avoidance or path planning [17, 26]. In this section, we use the techniques of double Q-learning [27] and dueling architectures [28] to improve the performance of DQN. In other words, we use Dueling Double Deep Q-networks (D3QN) to approximate the Q-values of all candidate actions based on which the optimal action is selected.

3.4.1 Double Q-Network and Dueling Architecture

In order to improve its stability, DQN uses a separate target network to generate the target Q-value y_t^{DQN} [16]:

$$y_t^{\mathrm{DQN}} = r_{t+1} + \gamma \max_{a'} Q\left(s_{t+1}, a'; \theta^-\right) \tag{5}$$

where θ^- is the parameter of the target network. However, the Q-values are still overestimated [29]. Eq. (5) can be written as follows [27]:

$$y_t^{\mathrm{DQN}} = r_{t+1} + \gamma Q\left(s_{t+1}, \operatorname*{argmax}_{a'} Q\left(s_{t+1}, a'; \theta^-\right); \theta^-\right) \tag{6}$$

The max operator in DQN uses the same values for both action selection and action evaluation. In contrast, double DQN (DDQN) decompose the action selection from the action evaluation to prevent the overestimation problem, as follows [27]:

$$y_t^{\mathrm{DDQN}} = r_{t+1} + \gamma Q\left(s_{t+1}, \operatorname*{argmax}_{a'} Q\left(s_{t+1}, a'; \theta\right); \theta^-\right) \tag{7}$$

where θ is the parameter of the online network for action selection while θ^- is the parameter of the target network for estimating the values.

In both DQN and DDQN, there are only a single stream of fully-connected layers following the feature vectors extracted by convolutional neural networks. Once given the current state, these networks will evaluate the Q-values of all action-state pairs. However, it is unnecessary to estimate the value of each candidate action for all states. The dueling architecture [28] has been proposed to further boost the performance of DQN and DDQN. This architecture constructs two streams of fully-connected layers (see Fig. 4) to separate the estimation of the state value function and the action advantage function, which are finally combined to approximate the Q-values for each valid action via a special aggregating operation [28]. Denote by V the state value function estimated by one stream of fully-connected layers, and A the advantage function produced by another stream. Then, the above aggregating operation can be described as:

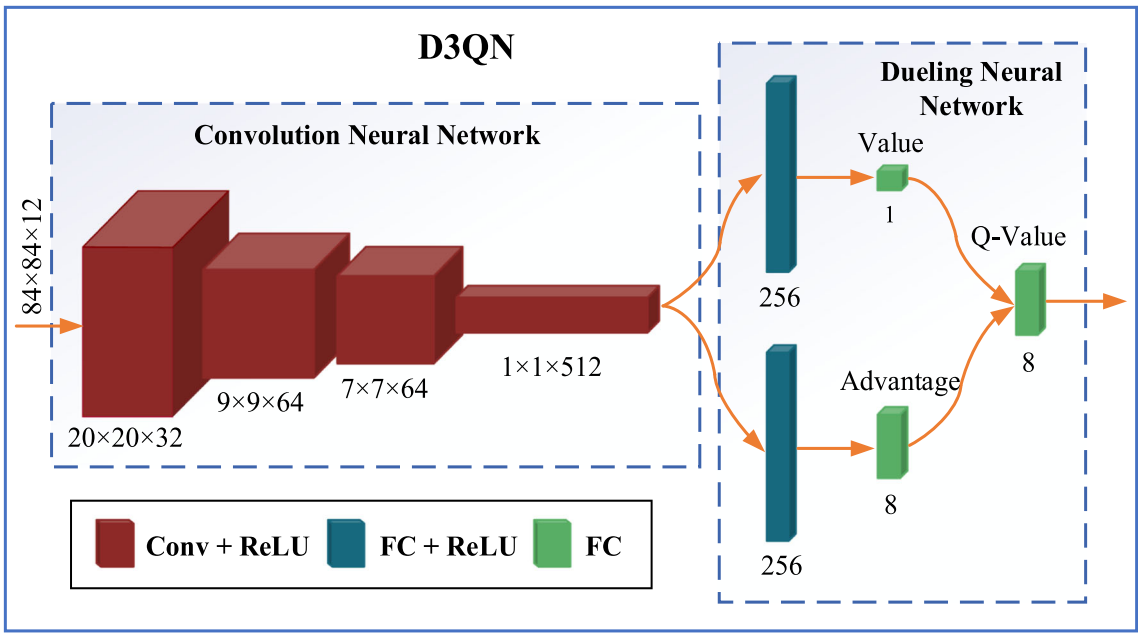


Fig. 4 The architecture of the D3QN. The input of the network is an 84 × 84 × 12 situation map stack constructed by the situation assessment model. It is followed by four convolutional layers and two fully-connected layers for two streams of dueling architecture

$$Q(s,a) = V(s) + \left(A(s,a) - \frac{1}{|\mathcal{A}|} \sum_{a'} A\left(s,a'\right) \right) \tag{8}$$

where $|\mathcal{A}|$ is the cardinal of A, which is equal to the size of the action set designed in Section 3.3.2.

The dueling architecture can be easily combined with DQN or DDQN, and several experiments have demonstrated that this combination leads to better results than the benchmarks [27].

3.4.2 Network Structure

To precisely estimate the state-action value function (Q-function), we establish a D3QN as illustrated in Fig. 4. The input of this network is the stacked last N frames situation map constructed according to the proposed situation assessment model, in which $N = 4$, although the D3QN is robust to different values of N (for instance, 3 or 5) [16]. Besides, its output is the Q-values of all valid actions.

The proposed networks consist of two sub-networks: a convolutional neural network and a dueling neural network. More specifically, the convolutional neural network contains four convolutional layers. The first convolutional layer (Conv1) has 32 filters of 8×8 with stride 4, followed by the second convolutional layer (Conv2) with 64 filters of 4×4 with stride 2 [30]; the third convolutional layer (Conv3) convolves 64 filters of 3×3 with stride 1, followed by the final convolutional layer (Conv4) with 512 filters of 7×7 with stride 1. We note that each of the convolutional layers is followed by a Rectified Linear Unit (ReLU) [31] activation function layer. Additionally, the dueling neural network consists of two fully-connected layers (FC1, FC2) for two streams of dueling architecture. The value and advantage streams both have a fully-connected layer with 256 hidden units, both followed by a ReLU layer. The final fully-connected layer of the value stream has one output, and the advantages stream has eight outputs, which is consistent with the number of valid actions designed in Section 3.3.2 [28]. The parameters of the above layers are given in Table 2.

3.5 Action Selection

In this paper, we combine the ε-greedy strategy with heuristic search rules to select actions for balancing the exploration and exploitation in order to improve the learning efficiency of the D3QN in the training stage based on [10].

At first, the UAV Agent selects actions from the action set randomly. To reduce the blindness of the Agent, heuristic search rules are used in the training stage. Specifically, when the Agent randomly selects an action from the action set with the possibility $1 - \varepsilon$, these rules should be satisfied according to the current location of the Agent and its destination (see Algorithm 1). For instance, assume that ($long_c$, lat_c) denotes the UAV location, ($long_t$, lat_t) represents its destination, if the destination is located to the southeast of the UAV, i.e. $long_c < long_t$ and $lat_c > lat_t$, then the UAV Agent will randomly select actions to execute from action 1 to action 5 without the need of trying every possible action from the entire action set (line 7).

In addition, pre-training episodes (*pre_episode*) are conducted before the formal training to accumulate and save experience into the replay memory for training the D3QN. In this process, the ability of D3QN to approximate Q-function is inadequate and not credible. To increase the possibility of reaching the target area and the gain more successful experience for the formal training, the UAV Agent selects an action randomly from the action set that satisfies the heuristic search rules, instead of selecting actions according to the predicted Q-values.

Algorithm 1 Action selection policy

Input: Q: the predicted state-action values of all valid actions;
($long_c$, lat_c): UAV location; ($long_t$, lat_t): destination;
episode_num: the current episode number of training;
pre_episode: the total episode number of pre-training;
Output: a: the selected action;

1. Generate p randomly, $p \in (0, 1)$
2. **if** $p > \varepsilon$ **and** *episode_num* > *pre_episod* **then**
3. $a = \text{argmax}(Q(s, a_i))$
4. **else**
5. **if** $long_c < long_t$ **then**
6. **if** $lat_c > lat_t$ **then**
7. $a \leftarrow$ random(1, 2, 3, 4, 5)
8. **else**
9. $a \leftarrow$ random(7, 0, 1, 2, 3)
10. **end if**
11. **else**
12. **if** $lat_c > lat_t$ **then**
13. $a \leftarrow$ random(3, 4, 5, 6, 7)
14. **else**
15. $a \leftarrow$ random(5, 6, 7, 0, 1)
16. **end if**
17. **end if**
18. **end if**

3.6 Summary of the Proposed Method

In this paper, we adopt the dueling double deep Q-networks (D3QN) algorithm [28] to train the UAV agent for path planning. In each training episode, the workflow of the proposed method is as follows:

Table 2 D3QN parameters

Name of layers	Size of filters or number of neurons	Stride
Conv1	(8, 8, 32)	4
Conv2	(4, 4, 64)	2
Conv3	(3, 3, 64)	1
Conv4	(7, 7, 512)	1
FC1 for advantage	256	–
FC1 for value	256	–
FC2 for advantage	8	–
FC2 for value	1	–

- Step 1: Empty replay memory D with capacity N; initialize the initial network parameters θ and the target network weights θ^- randomly;
- Step 2: Set the UAV to the initial position in the STAGE Scenario;
- Step 3: Capture situation data from STAGE via the *Stage Interface* plug-in, and then build the situation map according to the proposed situation assessment model as the current state s;
- Step 4: Select an action a using the proposed action selection policy;
- Step 5: Execute the selected action a, construct the new situation map as the subsequent state s' based on the newly obtained situation data, and observe the immediate reward r according to Table 1;
- Step 6: Store transition tuple (s, a, r, s') in D, and replace the oldest tuple if $\|D\| > N$;
- Step 7: Sample a minibatch of N_b tuples (s, a, r, s') from D randomly;
- Step8: Construct target values, one for each of the N_b tuples:

$$y_j = \begin{cases} r & \text{if } r = -50 \text{ or } 200 \\ r + \gamma Q\left(s', \underset{a'}{\operatorname{argmax}}\, Q\left(s', a'; \theta\right); \theta^-\right) & \text{otherwise} \end{cases} \tag{8}$$

- Step 9: Update initial network parameters θ through performing a gradient descent step with loss L:

$$L = \frac{1}{N_b} \sum_j \left\| y_j - Q(s, a; \theta) \right\|^2 \tag{9}$$

- Step 10: Modify target network parameters θ^- toward initial network weights θ every N^- steps with update rate τ, rather than directly copying the weights:

$$\theta^- \leftarrow \tau\theta + (1-\tau)\theta^- \tag{10}$$

- Step 11: Reset current state $s \leftarrow s'$;
- Step 12: Terminate the current episode and start the next episode if the new state of UAV satisfies the termination conditions mentioned in Section 3.3.3. Otherwise, return to perform Step 4.

4 Experiments and Analysis

In this section, different scenarios are established based on STAGE and several experiments are conducted to evaluate the feasibility and effectiveness of our proposed D3QN algorithm for UAV path planning. Our experiments are composed of two phases: off-line training and on-line testing/utilizing. To be specific, we get the optimal parameters of D3QN by training in a static scenario firstly. After that, a dynamic environment is built for further training. Finally, the optimal path of UAV is generated by our proposed D3QN algorithm with the trained parameters. In this way, we effectively offload the on-line computation to an off-line learning procedure.

4.1 Experimental Platform

As shown in Fig. 1, the proposed framework for UAV path planning consists of an Environment and an Agent. Thus, the implementation platform of experiments in our study also consists of two parts: PC1 for running the Environment and PC2 for running the Agent. PC1 and PC2 are connected through an RJ45 patch cable and communicated via UDP socket.

Throughout the whole experiment, PC1 used for running the STAGE Scenario and the *Stage interface* plug-in is a laptop equipped with an Intel Core i7-4710MQ CPU and the Windows 7 operating system. However, as mentioned above, our experiments are composed of two phases: off-line training

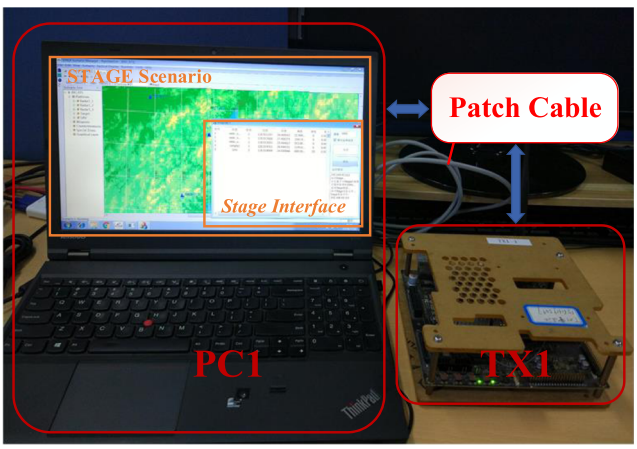
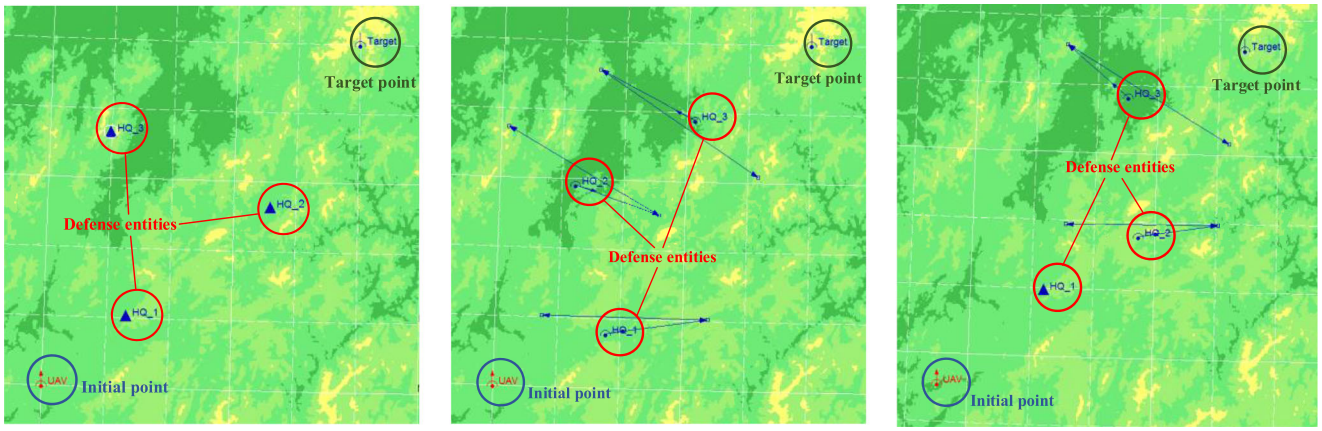


Fig. 5 The experimental platform consists of two main parts: a laptop equipped with an Intel Core i7-4710MQ CPU (PC1), and an NVIDIA Jenson TX1. They are connected through an RJ45 patch cable. Notice that TX1 shown in this figure is used for testing while another laptop equipped with an NVIDIA GTX 980 GPU (not shown in this figure) is devoted for training



(a) The static scenario for training **(b)** The dynamic scenario for training **(c)** The unseen scenario for testing

Fig. 6 Two simulation scenarios for training and one simulation scenario for testing. Notice that the circles shown in the above figure are added in post-processing for highlighting the entities. In the static scenario (**a**), there are one UAV, one reconnaissance target, and three stationary defense entities in the task area of UAV. Besides, in the dynamic scenario (**b**) and the testing scenario (**c**), the blue lines represent the desired trajectory of moving defense entities rather than its actual trajectory. Due to the turning radius, they do not always overlap. In addition, there are one stationary defense entity and two moving defense entities in the testing scenario. Note that the UAV's target point and initial position, and the deployment of enemy's defense entities are both different from the scenarios

and on-line testing. Due to the training phase requires more computing resources than the testing phase, PC2 used in the two experimental phases should be different. Specifically, the Agent is trained on another laptop equipped with an NVIDIA GTX 980 GPU, but tested on an NVIDIA Jenson TX1, which is used as onboard computer platform of UAV. Both devices (PC2) have an Ubuntu 14.04 operating system installed. The experimental platform used for testing in this paper is shown in Fig. 5.

The *Stage Interface* plug-in is developed using C++ language for acquiring situation data from the STAGE Scenario. The main implementation steps of this plug-in are as follows [13]: using the key instruction statement "*pd* = *stage_pd_attach*()" to establish a connection with the shared memory through a common data interface. After the STAGE simulation launch, the pointer named as "*pd*" obtains the shared memory address and the data in memory can be accessed and modified by using "*pd* - > "instruction. For example, data of the *i*-th entity in the simulation can be obtained by using "*pd* -> *ent[i]*" instruction. Additionally, the Agent is programmed using Python language. To build and train the dueling double deep Q-network (D3QN) for approximating the Q-values of all valid actions more easily, we used TensorFlow library in Python.

4.2 Scenario Description

In order to verify the performance of our proposed algorithm in static environments, we establish a static simulated scenario in STAGE firstly. As seen in Fig. 6(a), the task area of UAV over 23° N ~ 30° N, 115° E ~ 122° E with one reconnaissance target and three stationary defense entities of enemy. More precisely, the defense entities are located at (27.67° N, 117.02° E), (26.58° N, 119.41° E), and (25.11° N, 117.29° E), respectively. The task of UAV is to reach the destination position (29.00° N, 121.00° E) safely from the initial position (24.00° N, 116.00° E) with the optimal routes.

On the basis of the static scenario above, a dynamic scenario is shown in Fig. 6(b) is built for further training. Concretely, we remain the initial position and the target position unchanged and set fixed waypoints for the defense entities of enemy, and then the defense entities can make reciprocating motion along with these waypoints with a constant

Table 3 Setting of experiment parameters

Parameter	Value
Side length of girds	10 (km)
$R_{R\ \max}$	120 (km)
$R_{M\ \max}$	90 (km)
$R_{Mk\ \max}$	60 (km)
Radius of target area	40 (km)
max_episode	10,000
max_step	500
pre_episode	100
Replay memory capacity N	50,000
Minibatch size N_b	32
Discount factor γ	0.972
Initial network learning rate	0.0001
Target network update rate τ	0.001
Update frequency $\bar{N}$	8
Initial exploration	1
Final exploration	0.1
annealing_episode	2000
Threshold of threat value T_σ	1

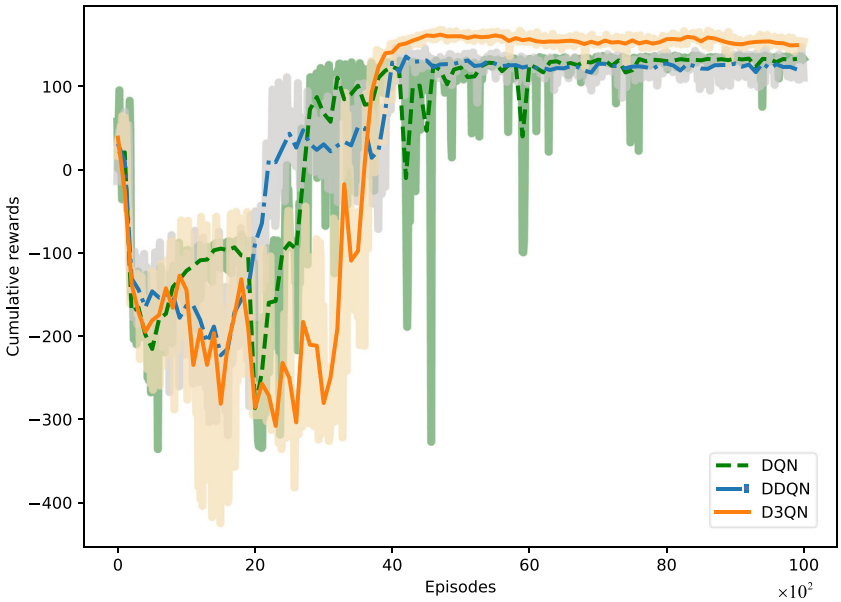


Fig. 7 Smoothed curves of cumulative rewards obtained from the static environment with the three algorithms

speed. We will start further training our D3QN in this dynamic scenario with the trained parameters in the above static scenario.

Additionally, to verify the generalization ability of our trained D3QN model, a new scenario with one stationary defense entity and two moving defense entities is established for testing experiments. As shown in Fig. 6(c), unlike the static scenario and the dynamic scenario we built for training experiments, the desired trajectory and the movement velocity of moving defense entities, the location of UAV's reconnaissance target (28.50° N, 120.50° E), and the initial position of UAV (23.50° N, 115.50° E) in the testing environment are all changed. The optimal path for the UAV in this scenario will be generated using our proposed D3QN path planner with the trained parameters.

4.3 Parameter Settings

As mentioned in Section 4.2, the task area of UAV is located in mid-latitude regions, so it can be considered that the length of a degree of longitude is roughly equal to a degree of latitude (about 100 km). For simplicity, we consider the length of one degree of longitude/latitude is equal to 100 km, and decompose the task area into grids with a side length of 10 km (i.e., 0.1 degrees of longitude/latitude). In the SA stage of experiments, without loss of generality, we set the parameters of

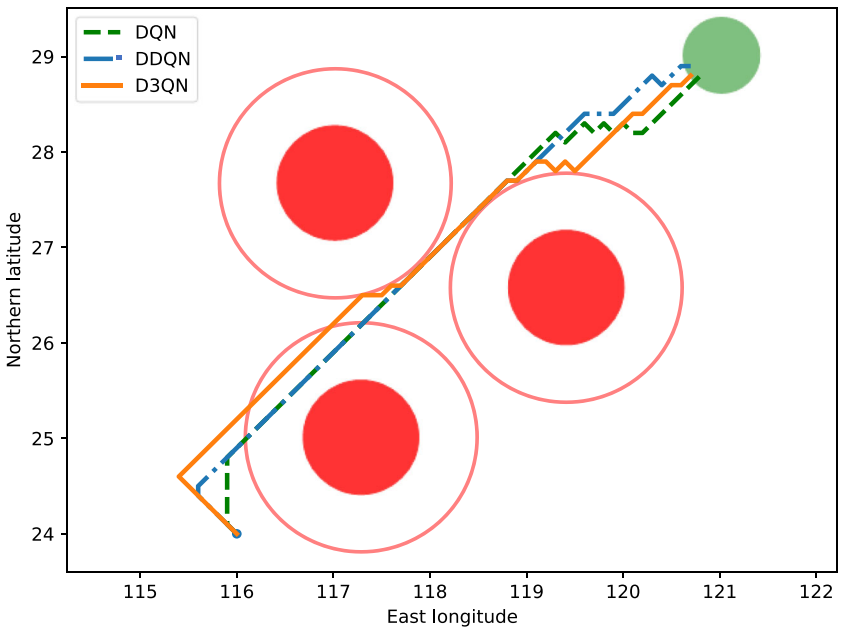


Fig. 8 Diagram of optimal trajectory planned by the three algorithms in the static scenario. Note that the area within the light-red circle (includes the dark-red filled area) represents the threat area, while the green circular area denotes the target area

Table 4 Comparison of the optimal trajectory generated by the three algorithms of DQN, DDQN and D3QN

Algorithm	Step	Cumulative reward
DQN	57	121.35
DDQN	56	122.77
D3QN	59	155.38

enemy defense units, $R_{R\ \max} = 120$ km, $R_{M\ \max} = 90$ km, $R_{Mk\ \max} = 60$ km; besides, the detection range of UAV is regarded as 40 km, which is equal to the radius of the target area.

During training, the maximum steps (*max_step*) of UAV is set to 500, which means that the UAV is deemed to fly over its maximum flight range if it fails to reach target position within 1000 steps in one episode; the threshold of threat value T_σ is set to 1. that is, it is considered that the UAV is destroyed by missiles if it enters the threat area where the threat value greater than or equal to 1. The initial network parameters are trained using Adam optimizer [32] with 0.0001 learning rate, 32 minibatch size; the target network weights are adjusted towards initial network every 8 steps with 0.001 update rate. In addition, the exploration rate ε, the proposed action selector's parameter, is annealed linearly from 1 to 0.1 over a period of 2000 (*annealing_episode*) episodes, and then fixed at 0.1 thereafter [16]. It should be mentioned that before the formal training begins, 100 episodes pre-training (*pre_episode*) are conducted to collect experience into the experience buffer. At the start of each pre-training episode, the initial position of UAV is randomly set among the task area, that increases the randomization and variety of training data [26]. In summary, our proposed D3QN algorithm is trained for a total of 10,000 episodes, the empirical values of essential parameters are provided in Table 3. We note that several experiments have been done with different parameter settings, but the results are quite stable. This illustrates that the proposed D3QN algorithm is robust to different parameter configurations.

4.4 Results and Analysis

4.4.1 Training in the Static Environment

In order to analyze the effectiveness of our revised D3QN algorithm and the superiority of dueling network and double Q-learning techniques in UAV path planning, a comparative experiment is conducted in this section. Specifically, the proposed D3QN algorithm, DDQN [27] and DQN [16] are adopted respectively with the same parameters for UAV path planning under the same static scenario described in Section 4.2, and then the results are compared. We note that DDQN is similar to DQN, but has a different way to express the target Q-values based on Eq. (5) and Eq. (7). The difference between D3QN and DDQN is that D3QN separates the fully-connected layers into two streams to estimate the state value function and advantage function respectively, which are then combined to produce the Q-values using Eq. (8), while DDQN constructs a single stream of the fully-connected layers to approximate the Q-values directly.

The smoothed curves of the cumulative rewards acquired by the three algorithms in the training period are shown in Fig. 7. Although the learning curves of DQN and DDQN converge to similar values, the cumulative rewards obtained by DDQN model is more stable than DQN. This means that the double Q-learning technique can relieve the overestimation issue and improve the training stability. Additionally, compared with DDQN, the cumulative rewards of D3QN increase slower in the early stages, but becomes higher finally. The reason is that, the separation of the single stream Q-network can learn the Q-function more efficiently. The above

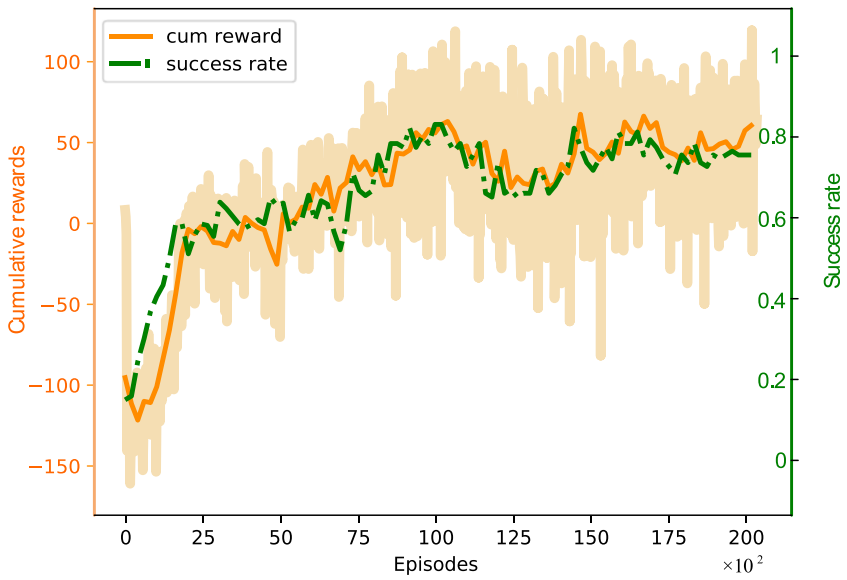


Fig. 9 Smoothed curves of cumulative rewards obtained from the dynamic scenario and success rate of UAV path planning. Notice that two curves use different y-axes

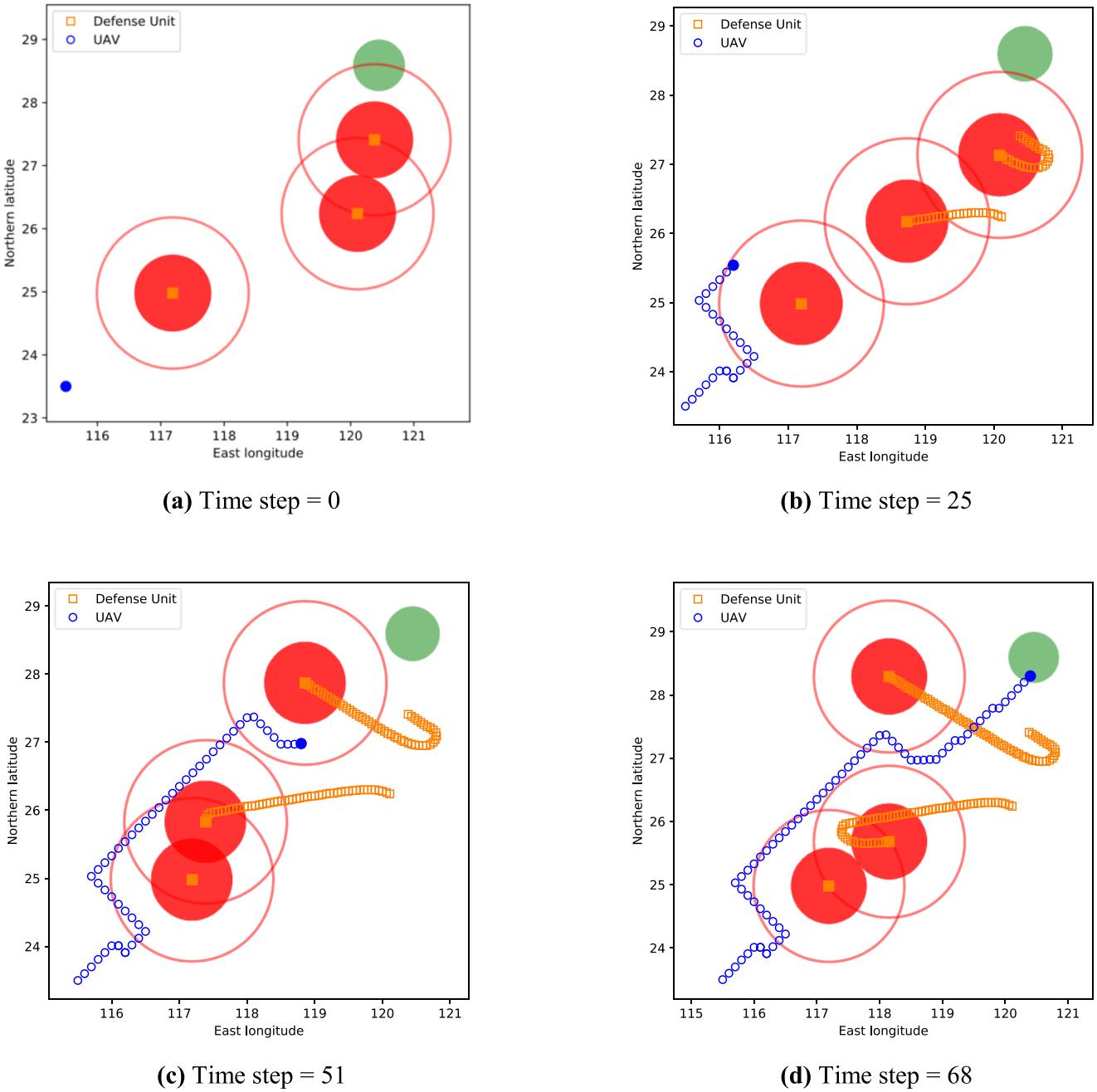


Fig. 10 A typical case that UAV successfully arrives at the target point with an acceptable safe trajectory generated by the trained D3QN model. It should be noted this figure only displays several representative and key time steps

results illustrate that D3QN outperforms DQN and DDQN in both the learning performance and the learning stability.

Table 5 Test results with different model snapshots

Model	Number of successful flights	Success rate (%)
16,000	132	66.00
18,000	138	69.00
20,000	134	67.00
Mean	134.67	67.33

The optimal trajectory planned by the three algorithms is illustrated in Fig. 8. Both the area within the light-red circle and the dark-red filled area denote the threat area, while the green circular area indicates the target area. Notice that we consider that the UAV is destroyed by defense entities of enemy if and only if the UAV enters the dark-red filled area. Intuitively, all three algorithms can successfully find a safe trajectory for the UAV. In addition to qualitative results shown above, quantitative results are given in Table 4 for comparing the performance of the three methods.

The optimal path generated by D3QN needs 59 steps to achieve the target with the cumulative reward of 155.38, but DQN and DDQN spends 57 and 56 steps, respectively, obtaining rewards of 121.35 and 122.77. In other words, D3QN can generate a safer path with higher rewards than DQN and DDQN, although this path takes more steps to reach the target area. The above results illustrate that both DDQN and D3QN perform well to plan paths for the UAV in the given static scenario, but D3QN is the best. Therefore, we only choose D3QN for further training of our path planner in the following dynamic scenario.

4.4.2 Training in the Dynamic Environment

In the dynamic scenario described in Section 4.2, we further train our path planning Agent 20,000 episodes. Beyond that, other parameters required for this experiment are all consistent with Table 3. It is worth mentioning that the initial parameters of the D3QN are assigned with the optimal weights trained in the above static scenario, rather than randomly. Besides, the learning rate of initial network and the update rate of target network are respectively fixed at 5×10^{-5} and 5×10^{-4} in the last 5000 training episodes. The curves of cumulative rewards obtained from the dynamic environment and success rate of UAV path planning are shown in Fig. 9.

As can be seen, the average of cumulative rewards that the UAV obtained from the dynamic environment in the last 5000 episodes only reaches 50.93, substantially below the cumulative rewards, as shown in Fig. 7, acquired from the static scenario. This is mainly attributed to the complexity and the variability of the dynamic environment increases the difficulty of path planning and prevent the UAV from obtaining higher rewards. Despite this, the success rate of UAV path planning in the last 5000 episodes is still up to 80.37% in average, which verifies the attractive performance of our proposed approach on path planning under dynamic battlefield environments.

4.4.3 Testing in the Unseen Scenario

In the scenario shown in Fig. 6(c), we conduct a testing experiment to verify the generalization ability of our trained D3QN model. More specifically, in this environment, we use the D3QN model with the optimal weights trained in the above dynamic scenario for UAV real-time path planning. It is worth noting that, in this testing experiment, the exploration rate ε and the pre-training steps are both set to 0, which means that the UAV entirely exploits Q-values predicted by our trained D3QN model to select actions and make decisions.

To evaluate the proposed method qualitatively, first of all, simulation results of testing experiments recording the movement sequence of UAV and enemy's defense units are represented in Fig. 10. In this figure, the solid blue circle and the hollow blue circle denote the current position of UAV and its historic movement sequences, respectively. Similarly, the solid orange squares and the hollow orange squares respectively indicate the current positions of enemy's defense units and their historic movement sequences. It is clear that the UAV reaches the target area successfully while avoiding the defense units of enemy, which illustrates that the trained D3QN path planner can generate acceptable safe trajectories in this testing environment.

Additionally, to avoid randomness and get quantitative results, the trained D3QN path planner is tested 200 episodes in each epoch with loading different model snapshots saved after a certain number of training episodes (e.g., 16,000, 18,000, or 20,000 episodes; see Table 5). The statistical results of the above experiments are shown in Table 5. As can be seen, although the new testing scenario is previously unseen in the training process, the UAV still successfully reaches the target area 134.67 times in average, and the mean of success rate is also up to 67.33%. The above results demonstrate the generalization ability and the application potential of our proposed path planning approach.

5 Conclusion

In this paper, we have proposed a deep reinforcement learning (DRL) method for UAV path planning in dynamic environments with potential enemy threats. The STAGE Scenario has been used to construct the DRL training environment that contains enemy defense units such as radars or missiles. Based on the global situation information in the UAV's task area, a fast assessment model has been proposed to construct a situation map corresponding to the overall threat values of each enemy unit to the UAV. Using a sequence of situation maps, an improved dueling double deep Q-networks (D3QN) algorithm can predict Q-values of all candidate actions. Compared with the DDQN algorithm, the proposed algorithm can achieve higher cumulative rewards and success rates. In addition, an action selection policy combining the ε-greedy strategy with heuristic search rules has been suggested for the UAV to avoid potential threats. Finally, the generalization ability and the real-time performance of the proposed method has been proved in both static and dynamic navigation tasks. However, we have made an assumption that the global situational data is provided to the UAV. This information is not always easily available for a single UAV performing a given task. Additionally, we have demonstrated the performance of the proposed method by several experiments conducted in simulated environments only. In the future, we will consider a team of UAVs that can communicate and share situational data among them, and further verify the feasibility and the generalization of our method in real world environments. Accordingly, we will develop the corresponding situation assessment model and learning algorithms in a decentralized way.

References

1. Tran, L.D., Cross, C.D., Motter, M.A., Neilan, J.H., Qualls, G., Rothhaar, P.M., Trujillo, A., Allen, B.D.: Reinforcement learning with autonomous small unmanned aerial vehicles in cluttered environments. In: Proceedings of AIAA Aviation Technology, Integration, and Operations Conference, 2899 (2015)
2. Faessler, M., Fontana, F., Forster, C., Mueggler, E., Pizzoli, M., Scaramuzza, D.: Autonomous, vision-based flight and live dense 3D mapping with a quadrotor micro aerial vehicle. J. Field. Rob. **33**, 431–450 (2016)
3. Scherer, S., Rehder, J., Achar, S., Cover, H., Chambers, A., Nuske, S., Singh, S.: River mapping from a flying robot: state estimation, river detection, and obstacle mapping. Auton. Robot. **33**, 189–214 (2012)
4. Xie, L., Wang, S., Markham, A., Trigoni, N.: Towards monocular vision based obstacle avoidance through deep reinforcement learning. arXiv:1706.09829(2017)
5. Ross, S., Melik Barkhudarov, N., Shankar, K.S., Wendel, A., Dey, D., Bagnell, J.A., Hebert, M.: Learning monocular reactive UAV control in cluttered natural environments. In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA), pp. 1765–1772 (2013)
6. Ma, Z., Wang, C., Niu, Y., Wang, X., Shen, L.: A saliency-based reinforcement learning approach for a UAV to avoid flying obstacles. Robot. Auton. Syst. **100**, 108–118 (2018)
7. Sutton, R.S., Barto, A.G.: Reinforcement Learning: an Introduction. MIT Press, Cambridge (1998)
8. Watkins, C.J., Dayan, P.: Q-learning. Mach. Learn. **8**(3–4), 279–292 (1992)
9. Zhao, Y., Zheng, Z., Zhang, X., Liu, Y.: Q learning algorithm based UAV path learning and obstacle avoidance approach. In: Proceedings of Chinese Control Conference (CCC), pp. 3397–3402 (2017)
10. Li, S., Xu, X., Zuo, L.: Dynamic path planning of a mobile robot with improved Q-learning algorithm. In: Proceedings of IEEE International Conference on Information and Automation, pp. 409–414 (2015)
11. Tang, R., Yuan, H.: Cyclic error correction based Q-learning for mobile robots navigation. Int. J. Control. Autom. Syst. **15**, 1790–1798 (2017)
12. Wang, C., Hindriks, K.V., Babuska, R.: Robot learning and use of affordances in goal-directed tasks. In: Proceeding of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2288–2294 (2013)
13. Yan, C., Xiang, X.: A path planning algorithm for UAV based on improved Q-learning. In: Proceedings of IEEE International Conference on Robotics and Automation Sciences, pp. 46–50 (2018)
14. Li, Y.: Deep Reinforcement Learning: an Overview. arXiv: 1701.07274(2017)
15. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv:1312.5602(2013)
16. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature. **518**(7540), 529–533 (2015)
17. Wu, J., Shin, S., Kim, C.G., Kim, S.D.: Effective lazy training method for deep Q-network in obstacle avoidance and path planning. In: Proceedings of IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 1799–1804 (2017)
18. Zhou, B., Wang, W., Wang, Z., Ding, B.: Neural Q learning algorithm based UAV obstacle avoidance. In: Proceedings of IEEE/CSAA Guidance, Navigation and Control Conference, pp. 961–966 (2018)
19. Wang, Y., Peng, D.: A simulation platform of multi-sensor multi-target track system based on STAGE. In: Proceedings of World Congress on Intelligent Control and Automation, pp. 6975–6978 (2010)
20. Deng, Y.: A threat assessment model under uncertain environment. Math. Probl. Eng. **2015**, 1–12 (2015)
21. Gao, Y., Xiang, J.: New threat assessment non-parameter model in beyond-visual-range air combat. Journal of System Simulation. **18**, 2570–2572 (2006)
22. Xiao, B., Fang, Y., Hu, S., Wang, L.: New threat assessment method in beyond-the-horizon range air combat. Syst. Eng. Electron. **31**, 2163–2166 (2009)
23. Ernest, N., Cohen, K., Kivelevitch, E., Schumacher, C., Casbeer, D.: Genetic fuzzy trees and their application towards autonomous training and control of a squadron of unmanned combat aerial vehicles. Unmanned Systems. **3**(03), 185–204 (2015)
24. Wen, N., Su, X., Ma, P., Zhao, L., Zhang, Y.: Online UAV path planning in uncertain and hostile environments. Int. J. Mach. Learn. Cybern. **8**, 469–487 (2017)
25. Kim, Y.J., Hoffmann, C.M.: Enhanced battlefield visualization for situation awareness. Comput. Graph. **27**, 873–885 (2003)
26. Tai, L., Liu, M.: Towards cognitive exploration through deep reinforcement learning for mobile robots. arXiv:1610.01733(2016)
27. Van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double Q-learning. In: Proceedings of AAAI Conference on Artificial Intelligence, pp. 2094–2100 (2015)
28. Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., De Freitas, N.: Dueling network architectures for deep reinforcement learning. In: Proceedings of International Conference on Machine Learning (ICML), pp. 1995–2003 (2016)
29. Van Hasselt, H.: Double Q-learning. In: Advances in Neural Information Processing Systems, pp. 2613–2621 (2010)
30. Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., De Freitas, N.: Sample efficient Actor-Critic with experience replay. arXiv:1611.01224(2016)
31. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proceedings of International Conference on Machine Learning (ICML), pp. 807–814 (2010)
32. Kingma, D.P., Ba, J.: Adam: a Method for Stochastic Optimization. arXiv: 1412.6980(2014)

Chao Yan is a graduate student in the College of Intelligence Science and Technology at National University of Defense Technology, Changsha, China. He received the B.S. degree (outstanding graduates) in the School of Information and Control Engineering from China University of Mining and Technology, Xuzhou, China, in 2017. His research interests include the applications of reinforcement learning techniques for UAV path planning.

Xiajia Xiang is an associate professor in the College of Intelligence Science and Technology, NUDT. He received the B.E., M.S., and Ph.D. degrees in automatic control from National University of Defense Technology (NUDT), Changsha, China, in 2003, 2007, and 2016, respectively. He currently works in the field of mission planning, autonomous and cooperative control of unmanned systems.

Chang Wang is now an assistant professor in the College of Intelligence Science and Technology at National University of Defense Technology. He received the B.S. degree from University of Science and Technology of China, and the M.S. degrees from National University of Defense Technology, in 2007 and 2009, respectively. He received the Ph.D. degree from Delft University of Technology on the topic of robot learning. His research interests include developmental learning, reinforcement learning, multi-agent systems, and human-robot collaboration.