Deep Reinforcement Learning based Adaptive Real-Time Path Planning for UAV

Jiankang Li
School of Automation Science and Electrical Engineering
Beihang University
Beijing, China
jorkerli1755@buaa.edu.cn

Yang Liu
School of Automation Science and Electrical Engineering
Beihang University
Beijing, China
qwertyliuyang@buaa.edu.cn

***Abstract*—Real-time path planning typically aims to obtain a collision-free and shorter path with lower computational complexity for UAVs in unknown environment. Apart from the above basic objective, kinematic constraints and the smoothness of path should be further considered especially for fixed-wing UAVs restricted by their maneuverability. In this paper, we propose an adaptive real-time path planning method based on Deep Reinforcement Learning. Taking the sensor data of obstacles nearby and the target's position relative to the UAV as the decision information, and designing the action satisfying kinematic constraints of fixed-wing UAV, the proposed method can plan a feasible path for fixed-wing UAV in real-time. Experimental results show that the adaptive action devised combining with greedy reward, granularity reward and smoothness reward can accelerate the convergence speed of the algorithm and enhance the smoothness of the planned path.**

Keywords: adaptive, real-time path planning, deep reinforcement learning, UAV, fixed-wing

I. Introduction

Today, Unmanned Aerial Vehicles (UAVs) have been widely applied in military and civilian tasks. As a basic task for UAVs, real-time path planning aims to obtain a path satisfying the desired requirements from the origin to the predefined destination without knowing environmental information in advance. Regarding the requirements, collision-free path to ensure the safety of UAV, shorter path to save fuel and time and lower computational complexity to generate a path in real-time are the typical performance pursued by real-time path planning algorithms. Moreover, the consideration of kinematic constraints and the smoothness of path are also significant requirements, especially for fixed-wing UAVs restricted by their maneuverability. If a planned path demands many agile or abrupt maneuvers, it would be difficult or even unfeasible to track. In this paper, the research effort focuses on real-time path planning with the kinematic constraints of fixed-wing UAVs, e.g., the limitations of yaw rate and flight speed.

For real-time path planning, a multitude of methods have been proposed, including map-based methods and map-less methods. Dynamic A* (D*) [1], Lifelong Planning A* (LPA*) [2] and D* Lite [3], as the classic map-based algorithms, search the optimal path of a graph in which the latest detected information of obstacles is updated. However, when confronted with high-dimensional data on obstacles detected by airborne sensors (e.g. radar, camera, range finder, etc.) from unknown environments, the map-based methods exhibits some limitations owing to the difficulties in mapping. It can be challengeable to both extract the effective information of obstacles from high-dimensional sensor data and construct an appropriate map that balances well the accuracy and the complexity of the representation of obstacles. Furthermore, when the unknown environment for real-time path planning is dynamic, it will make the constructed map lose timeliness and become inaccurate. In terms of the map-less methods, instead of mapping first and then re-planning, the most common way utilizes the reactive behavior where the planner of real-time path planning makes decisions directly based on the obtained information of obstacles, the target, etc., which helps them bypass the challenges caused by mapping. Nevertheless, for traditional reactive methods such as Bug algorithm [4], Artificial Potential Field [5] and Fuzzy Logic [6], they still exist the limitation of extracting the high-dimensional sensor data effectively.

Deep Reinforcement Learning (DRL), integrating the powerful perception and representation ability of deep neural network to deal with high-dimensional decision information and the learning ability for decision-making of reinforcement learning through trial-and-error, has been applied in [7], [8]. As a learning-based reactive algorithm for real-time path planning, DRL demonstrates the potential to overcome the aforementioned challenges. For this reason, we resort to an excellent DRL algorithm, Deep Q-Network (DQN) [9], to address real-time path planning in unknown environment.

In terms of fixed-wing UAVs, in order to make the planned path meet their kinematic constraints, the yaw rate, as an action of DRL agent, is designed to satisfy the constraint of the maximum yaw rate. In addition, the time step, regarded as the other action of the agent, is designed to be variable, compared to the fixed time step, which is utilized to provide an adaptive action to cope with the environment under different granularities, where the agent can take a longer time step when confronts with a relatively open environment, whereas a shorter time step can be taken in a relatively narrow envi-

This work was supported by the National Natural Science Foundation of China under Grant 61772055 and Grant 61872169.

2767-6684/21/$31.00 ©2021 IEEE
DOI 10.1109/DSA52907.2021.00077

ronment. Obviously, variable time step can effectively avoid the contradiction between the loss of path accuracy in a short time step and the reduction of algorithm efficiency in a long one.

This work mainly contributes to the following two aspects:

- Based on DRL, the adaptive real-time path planning approach is proposed, which not only takes consideration of the kinematic constraints of fixed-wing UAV, but also has the adaptive ability to deal with the environment under different granularities.
- In the reward design, in addition to basic reward, greedy reward is designed to accelerate the convergence speed of the algorithm, while granularity reward and smoothness reward are introduced to reduce the switching times of maneuvers and enhance the smoothness of the planned path.

The remainder of this paper is organized as follows. Section II investigates the related work. Section III presents the preliminaries of kinematic constraints of fixed-wing UAV and DQN algorithm. Section IV introduces our approach including decision information, adaptive action and reward design. Experiments under specially designed scenarios are implemented and analyzed in Section V. Finally, conclusions and future work are presented Section VI.

II. Related work

There is a large body of work on real-time path planning based on the detected high-dimensional environmental data. In this paper, the literature investigated mainly focuses on learning-based methods owing to their excellent performance in feature extraction. One class of learning-based methods for real-time path planning is Imitation Learning (IL). Pfeiffer et al. [10] present a target-oriented end-to-end navigation model for a robotic platform. By imitating those expert demonstrations generated by an existing motion planner, the navigation model can learn the complex mapping from raw 2D-laser range findings and a target position to the required steering commands for the robot. In literature [11], a deep LSTM model constructed by Long Short-Term Memory networks imitates the paths generated by a static path planning algorithm A* and shows that it can improve the global optimization quality of the path in real-time path planning.

Reinforcement Learning, as another class of learning-based methods, has been successfully employed to address real-time path planning problems. For mobile robots, Tai et al. [12] present an end-to-end mapless motion planner to navigate the nonholonomic mobile robot, which is trained by an asynchronous DRL algorithm without any manually designed features and prior demonstrations. Taking the kinematic model and constraint conditions of mobile robots into account, Yan et al. [13] propose a DRL based path planning method for mobile robot under continuous action space. In terms of UAVs, Sampedro et al. [14] pursue the reactive navigation with fast collision avoidance capabilities in which an artificial potential field formulation is adopted to guide the design of reward for DRL agent. Wang et al. [15] formulate navigation task in large-scale complex environments as a Partially Observable Markov Decision Process (POMDP). The navigation model directly maps UAVs' raw sensory measurements into control signals and is solved by a novel online DRL algorithm within the actor-critic framework. Zhao et al. [16] present an actor-critic model incorporating LSTM networks and a coordination strategy to identify and resolve the problem of collision avoidance for a variable number of fixed-wing UAVs in limited airspace. In reference [17], a DRL based path planning algorithm oriented to large-scale and dynamic environments is proposed, in which an action selection strategy to reduce the meaningless exploration and an adaptive experience replay mechanism based on the frequency of failure are developed to improve the stability and learning efficiency of Deep Q-Network and Deep Recurrent Q-Network. Cui et al. [18] present a 2-layer path planning method where the higher layer deals with the local information whereas the lower layer deals with the global information. After obtaining the planned path, B-spline curve approach is applied for online path smoothing. Different from the above work which focuses on mobile robots or quadrotor vehicles, or ignores the constraints of fixed-wing UAVs on maneuverability, we develop a real-time path planning method with attention to kinematic constraints of fixed-wing UAVs.

III. Preliminaries

A. Kinematic Constraints of UAV

In this paper, we assume UAV flies at a fixed height, i.e., the activities of UAV are restricted to the x - y plane. Consequently, as shown in Fig. 1, a 2-dimensional fixed earth coordinate system can be used to describe UAV's absolute position (x_t, y_t) and orientation θ_t at time t, which constitutes the state of UAV, denoted as $\begin{bmatrix} x_t & y_t & \theta_t \end{bmatrix}^{\mathrm{T}}$. Given control input $(\omega_t, v_t, \Delta t)$ including the yaw rate ω_t, the flight speed v_t and the time step Δt, we can calculate the increment of orientation and position respectively:

$$\Delta\theta_t = \omega_t \Delta t$$

$$\begin{matrix} \Delta x_t \\ \Delta y_t \end{matrix} = \begin{cases} v_t/\omega_t[\sin(\Delta\theta_t - \theta_t) + \sin\theta_t], \\ v_t/\omega_t[\cos(\Delta\theta_t - \theta_t) - \cos\theta_t], \end{cases} \text{if } \omega_t \neq 0$$

$$\begin{matrix} v_t \Delta t \cos\theta_t \\ v_t \Delta t \sin\theta_t, \end{matrix} \qquad \text{if } \omega_t = 0$$

and obtain the kinematic equation or the state transition of UAV, formulated as:

$$\begin{bmatrix} x_{t'} \\ y_{t'} \\ \theta_{t'} \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} + \begin{bmatrix} \Delta x_t \\ \Delta y_t \\ \Delta\theta_t \end{bmatrix}$$

where $t' = t + \Delta t$ denotes the next decision-making time.

In order to meet the kinematic constraints of fixed wing UAV and make flight path feasible for it to track, the following restrictions are imposed on the control input:

- Considering the turning ability of fixed wing UAV, the yaw rate ω_t is limited by the maximum yaw rate $\omega_{\max}$:

$$\omega_t \in [-\omega_{\max}, \omega_{\max}] \tag{1}$$

- The flight speed v_t needs to be within a reasonable range:

$$v_t \in [v_{\min}, v_{\max}] \tag{2}$$

- Considering the detection range of airborne sensors, it is too dangerous for UAV to go out of detection range under one decision-making in unknown environment. Thus, the time step Δt is limited by UAV's the maximum detection range $d_{\max}$ and the flight speed v_t:

$$\Delta t \in (0, \frac{d_{\max}}{v_t}) \tag{3}$$

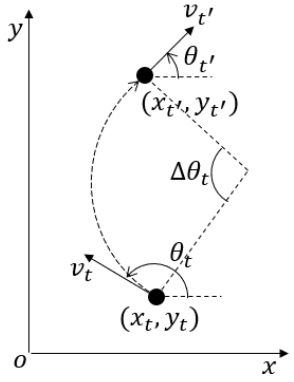

Fig. 1. The state transition of UAV.

B. Measurements

Definition 1 (Path smoothness): The path smoothness is a measurement of the adjustment amplitude of yaw rate in turning operations that UAV has made during the whole flight from origin to target. We define PS as the path smoothness:

$$\mathrm{PS} = 1 - \frac{\sum_{k=1}^{N} |\omega_{t_k} - \omega_{t_{k-1}}|}{2\omega_{\max} \cdot N} \in [0, 1] \tag{4}$$

where ω_{t_0} is the initial yaw rate and set to 0, N is the change times of yaw rate. A larger PS means a smoother path. Especially, when PS is 1, it indicates that the planned path is a straight line.

Definition 2 (Success rate): The success rate measures the proportion of the successful scenarios where the path planner can obtain a feasible path from origin to destination in all test scenarios. We define SR as the success rate:

$$\mathrm{SR} = \frac{M_{\text{success}}}{M_{\text{test}}} \tag{5}$$

where M_{success} and M_{test} denote the number of the successful scenarios and all test scenarios, respectively.

Definition 3 (Path length): The path length measures the distance flying from origin to destination along the planned path, defined as:

$$\mathrm{PL} = \sum_{k=1}^{N} v_{t_k} \Delta t_k \tag{6}$$

Definition 4 (Average step length): The average step length is a measurement of the average distance interval between two consecutive plannings in a flight mission, defined as:

$$\mathrm{ASL} = \frac{\mathrm{PL}}{N} = \frac{\sum_{k=1}^{N} v_{t_k} \Delta t_k}{N} \tag{7}$$

Generally, a smaller ASL means that higher planning frequency and better real-time performance are required for path planning algorithms.

C. DQN

The sequential decision problems, addressed by reinforcement learning, are generally modeled as Markov Decision Process (MDP) in which at each time step t, the agent executes an action A_t at state S_t, which results in its transition to a next state S_{t+1} and receives a reward R_{t+1} characterizing the feedback of the environment toward the action. The agent improves its policy for decision-making through trial-and-error in the environment. Concretely, the policy $\pi = p(a|s)$ of the agent is optimized by maximizing the expectation of cumulative future reward for each state-action pair (s, a):

$$q_\pi(s, a) = \mathbb{E}_\pi[\sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1} | S_t = s, A_t = a]$$

where $q_\pi(s, a)$ is the action-value function regarding the policy π, and $\gamma \in [0, 1]$ is the discount factor determining the agent's horizon.

DQN, as a typical DRL algorithm, utilizes the deep neural network to approximate the action-value function $q(s, a)$ to avoid the dimensional explosion of state space faced by traditional tabular RL, which results in the parameterized action-value function $q(s, a; w)$ termed the q-network. The update rule for the q-network's parameters w is as follows:

$$w \leftarrow w + \alpha[r + \gamma \max_{a'} q(s', a'; w^-) - q(s, a; w)] \nabla_w q(s, a; w)$$

where α is the learning rate. Besides the q-network, a target network $q(s, a; w^-)$ with lower update frequency than the q-network is used to enhance the stability of target value $r + \gamma \max_{a'} q(s', a')$. The training data is randomly sampled from the replay memory that stores the experiences generated in the interaction between the agent and the environment, which removes correlations in the experiences and helps the training of RL more stable. When interacting with the environment, ϵ-greedy policy is typically taken as the agent's behavior policy in DQN.

IV. Approach

In this section, we describe the design of state, action and reward in the MDP formulated by real-time path planning. To distinguish the *state* of UAV, the term *state* used in MDP below is referred as *decision information*.

A. Decision Information

Confronted with unknown environment, UAV to make a decision in real-time shall at least be capable of obtaining the following two kinds of information: the observation of obstacles around UAV and the positional relationship between target and UAV, which are used to ensure the safety of the planned path and avoid UAV getting lost, respectively.

In terms of the observation of obstacles around UAV, we use the distance data from n_{O} range finders which detect objects from the degree $-\theta_{\mathrm{O}}^{\max}$ to $\theta_{\mathrm{O}}^{\max}$ of UAV within the maximum range $d_{\mathrm{O}}^{\max}$ to characterize it, denoted as $s_{\mathrm{O}}^t = \left(d_1^t, d_2^t, ..., d_{n_{\mathrm{O}}}^t\right)$ and depicted in Fig. 2.

And for the positional relationship between target and UAV, we use polar coordinates of target's position in the drone-centric body coordinate system to represent it, denoted as $s_{\mathrm{T}}^t = (d_{\mathrm{T}}^t, \theta_{\mathrm{T}}^t)$ and depicted in Fig. 2. Typically, the range of target azimuth θ_{T}^t is set as $(-\pi, \pi]$.

To summarize, we can obtain the decision information s_t used for the agent (i.e., UAV) decision-making at time t by composing the above two kinds of information:

$$s_t = \left(s_{\mathrm{O}}^t, s_{\mathrm{T}}^t\right) = \left(d_1^t, d_2^t, ..., d_{n_{\mathrm{O}}}^t, d_{\mathrm{T}}^t, \theta_{\mathrm{T}}^t\right)$$

where n is the number of range finders used.

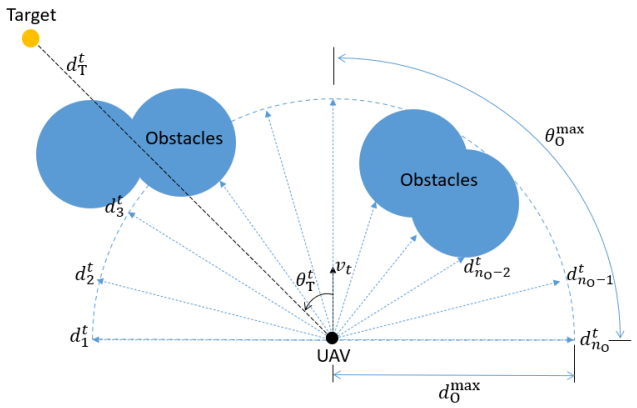

Fig. 2. Decision information.

B. Adaptive Action

As described in section III-A, the control input consists of the yaw rate ω_t, the flight speed v_t and the time step Δt, satisfying the constraints (1), (2) and (3) respectively. For simplicity, we assume UAV flies at a fixed speed, i.e., $v_t = v_{\min} = v_{\max} = v_0$. Excluding the flight speed, we take the yaw rate and the time step as the agent's action, which controls UAV's maneuvering direction and distance respectively when executing an action. Since the nature that DQN is a discrete action RL algorithm, we design the discrete action set that satisfies the corresponding kinematic constraints for the yaw rate and the time step, respectively.

- Regarding the yaw rate ω_t, as demonstrated in Fig. 3, the action set A_ω for the agent is designed as n_ω discrete values:

$$\omega_t \in A_\omega = \{\frac{k\omega_{\max}}{n'_\omega} | -n'_\omega \le k \le n'_\omega, k \in \mathbb{Z}\}$$

  where $n'_\omega = \frac{n_\omega - 1}{2}$ and $n_\omega = |A_\omega|$ is a odd number greater than 1. For a given time step, the flight distance to perform each action is the same at a constant flight speed.
- As for the time step, instead of using a fixed time step, a variable time step is utilized to provide an adaptive action to deal with the environment with different granularity, where the agent can take a longer time step when confronts with a relatively open environment; otherwise, it is the opposite. Concretely, we design $n_{\Delta t}$ linearly increasing multiples of basic time step Δt_0 as the time step action set $A_{\Delta t}$:

$$\Delta t \in A_{\Delta t} = \{k\Delta t_0 | k = 1, 2, ..., n_{\Delta t}\}$$

  where $\Delta t_0 = \frac{\frac{\pi}{4}}{\omega_{\max}}$ and $n_{\Delta t}\Delta t_0 < \frac{d_{\max}}{v_t}$. Taking Δt_0 as a time step action, UAV will rotate one-eighth of a circle when flying at the maximum yaw rate $\omega_{\max}$, as shown in Fig. 3.

Combining the action set of yaw rate and time step, we get the 2-dimensional action set A of the agent:

$$A = A_\omega \times A_{\Delta t} = \{(\omega_t, \Delta t) | \omega_t \in A_{\Delta t}, \Delta t \in A_{\Delta t}\}$$

where the size n_A of the action set A is $n_\omega \cdot n_{\Delta t}$.

It should be noted that DQN does not support multidimensional action directly. We eliminate such gap by building a bijective mapping f between the 2-dimensional action set A and the 1-dimensional action set $A' = \{1, 2, ..., n_A\}$:

$$f : A \to A'$$

When an action $a \in A$ is passed from the agent to the DQN, it will be converted to the corresponding action $a' \in A'$ by f. And when an action $a' \in A'$ is passed from the DQN to the agent, it will be converted to the corresponding action $a \in A$ by f^{-1}.

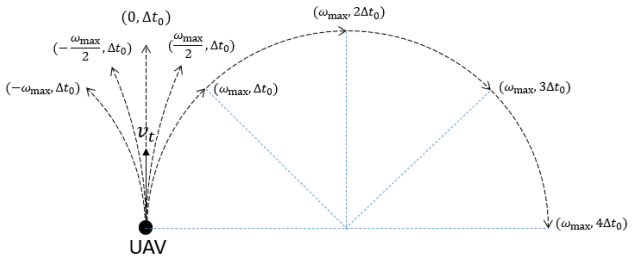

Fig. 3. Adaptive action where $n_\omega = 5$ and $n_{\Delta t} = 4$.

C. Reward Design

Considering that reward plays a key role in the effect and efficiency of RL algorithm, basic reward and greed reward are first provided to achieve the basic objective of real-time path planning and accelerate the convergence speed of the algorithm, respectively. Besides, smoothness reward and granularity reward are designed to improve the smoothness of the planned path and increase the granularity of action.

- Basic reward R_{bs} is used to tell the agent where the target is and where the obstacles are, designed as: a positive reward r_{T} will be given when UAV reaches the target, whereas a punishment (i.e., negative reward) r_{O} will be given when UAV collides with an obstacle.
- Greed reward R_{gd} is based on the greedy strategy that only utilizes the information of the target but ignores the information of obstacles to guide UAV to fly towards the target directly, which can make the agent reduce some aimless wandering during the training process and accelerate the convergence speed of RL algorithm. To this aim, the greed reward is designed as:

$$r_{d_{\mathrm{T}}} \cdot \mathrm{sgn}(d_{\mathrm{T}}^{t^-} - d_{\mathrm{T}}^{t}) + r_{\theta_{\mathrm{T}}} \cdot \mathrm{sgn}(\theta_{\mathrm{T}}^{t^-} - \theta_{\mathrm{T}}^{t})$$

  where sgn is the signum function, d_{T}^{t} (θ_{T}^{t}) and $d_{\mathrm{T}}^{t^-}$ ($\theta_{\mathrm{T}}^{t^-}$) denote the target distance (the target azimuth) at current time t and last time t^- respectively.
- Smoothness reward R_{sm} is utilized to reduce unnecessary switching of the yaw rate between two consecutive actions and increases the smoothness of the planned path, which makes it easier to be tracked by UAV. For this purpose, we design the smoothness reward as:

$$r_{\omega} \cdot \frac{|\omega_t - \omega_{t^-}|}{2\omega_{\max}}$$

  where ω_t and ω_{t^-} denote the yaw rate at current time t and last time t^- respectively.
- Granularity reward R_{gn} is designed to encourage the agent to take a larger time step that can effectively reduce the number of decision-makings and switching of the control input, which conducive to reduce the training time and improve the smoothness of path. Thus, the granularity reward is formulated as:

$$r_{\Delta t} \cdot \frac{\Delta t}{\Delta t_{\max}}$$

  where $\Delta t_{\max}$ is the element with largest value in $A_{\Delta t}$.

V. Experiment

A. Experimental Setup

We construct two types of scenes for real-time path planning, depicted in Fig. 4. Each scene is 700×700 in size and crowded with circular obstacles generated randomly. In addition, the boundary of the scene is also regarded as an obstacle. For each scenario, the origin is placed at one of four corners of a scene, and the target is placed at the opposite corner of the same scene. Thus, one scene can be used to generate 4 scenarios, as demonstrated in Fig. 4. In our experiment, 50 scenes (i.e., 200 scenarios) and 100 scenes (i.e. 400 scenarios) are utilized for training and testing the agent's policy respectively.

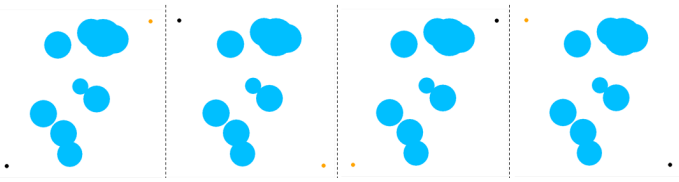

Fig. 4. 4 scenarios based on a scene (blue circles, black dot and orange dot denote obstacles, the origin and the target respectively).

The action-value function $q(s, a; w)$ in DQN is parameterized by the deep neural network, whose structure is demonstrated in Fig. 5. First, the observation of obstacles s_{O}^{t} is fed to the convolution neural network module (CNN Module) for preprocessing, and then its output is concatenated with the target information s_{T}^{t}, which is fed to the fully connected network module (FC Module) and outputs all the action values corresponding to the decision information s_t. In s_t, the variables representing the distance $d_1^t, d_2^t, ..., d_{n_{\mathrm{O}}}^t, d_{\mathrm{T}}^t$ and the angle θ_{T}^t are normalized to $[0, 1]$ and $[-1, 1]$ respectively.

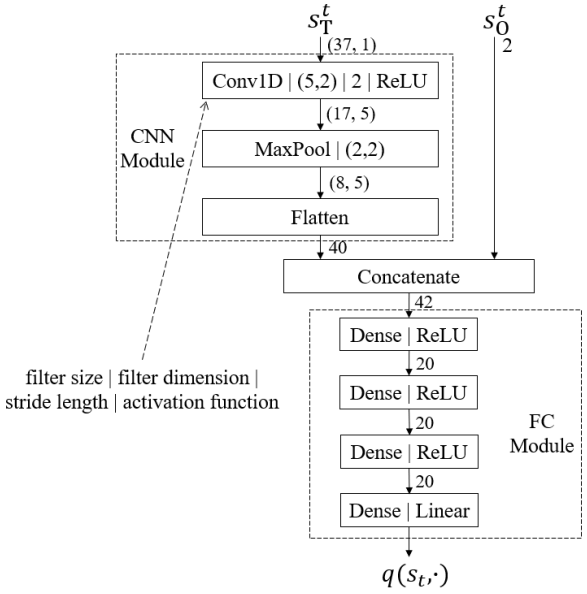

Fig. 5. The network structure for $q(s, a; w)$.

The parameters of UAV's kinematics and airborne sensors are listed in Table I, and the designed rewards are listed in Table II. The action set sizes n_{ω} and $n_{\Delta t}$ are set to 5 and 4 respectively.

The agent is trained by traversing the 200 scenarios for training 30 times. While the training, the neural network parameters are performed an update using 64 samples extracted randomly from the replay memory every executing 5 actions, in which the Adam optimizer [19] with the learning rate 0.001 is employed, and the parameters of the target network are replaced by the parameters of the evaluate network every

executing 5 updates. The size of the replay memory is set to 500. The discount factor γ and the proportion ϵ of random exploration in ϵ-greedy behavior policy are set to 0.9 and 0.1 respectively. Our neural network model implementation is based on Google's TensorFlow framework [20]. Using these libraries, all the models have been trained on a GPU Nvidia GeForce GTX 970.

TABLE I
THE PARAMETERS OF KINEMATIC CONSTRAINTS AND AIRBORNE SENSORS

Parameter	$\omega_{\max}$	v_0	$\theta_{\mathrm{O}}^{\max}$	$d_{\mathrm{O}}^{\max}$	n_{O}
Value	$\frac{\pi}{60}$rad/s	0.05km/s	$\frac{\pi}{2}$rad	11.5km	37

TABLE II
DESIGNED REWARDS

Parameter	r_{T}	r_{O}	$r_{d_{\mathrm{T}}}$	$r_{\theta_{\mathrm{T}}}$	r_{ω}	$r_{\Delta t}$
Value	10	-10	0.2	0.2	-0.1	0.1

B. Simulations and Discussions

The main objective of the simulation experiments presented in this section is to evaluate the performance for the proposed DRL based adaptive real-time path planner and demonstrate the contribution of each part of the designed rewards to the results. In the following, the statistical data are the results of 15 repeated experiments.

RQ1: What is the performance of the proposed method?

To demonstrate the performance of our method, Bi-level Programming (BLP) [21], as an advanced real-time path planning algorithm with kinematic constraints, is selected as a benchmark for comparison of our method. It should be noted that our method is different from BLP in the way of obtaining obstacle information. BLP is a map-based method mainly applied to those scenarios with virtual threats, in which UAV can obtain the complete information of obstacles including their position and shape within the detection range, whereas in our method, only relatively limited obstacle information is observed by airborne range sensors.

Table III demonstrates the statistical results of BLP and our method with the measurement of success rate, path length, path smoothness and average step length. Even if less environmental information is used, our method has a higher success rate compared with BLP (96.82% vs 95.00%), and they perform relative close regarding path length and path smoothness. Specifically, in all test scenarios, the average length of path planned by our method is 8.39% longer than that of BLP, and in terms of smoothness, our method achieves 85.98% of BLP. In addition, the average step length our method takes is much smaller than that of BLP. This difference is reasonable and mainly due to the weaker capability of sensing environment. It is dangerous to take a big step in relatively limited perception, and more environmental data needs to be collected to identify obstacles and channels and understand the shape of obstacles.

The path planning results of two randomly selected test scenarios are shown in the Fig. 6, which demonstrates that when UAV enters a dangerous area (within the circle of black dotted line in the Fig. 6), small steps are taken to increase the sampling frequency of sensors to collect more environmental information. However, in relatively simple environments, long steps are taken to provide a smoother path and help to reduce the training time of the DQN algorithm. The above behaviors show that the agent obtained by our method has good adaptive ability in real-time path planning.

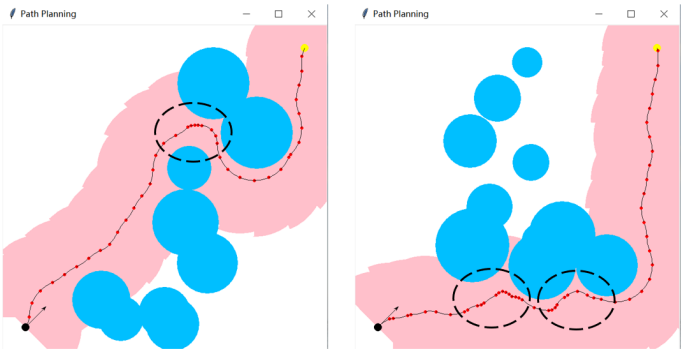

Fig. 6. Two randomly selected test scenarios and their planned paths (the pink shadow represents the detection area during flight).

RQ2: What is the contribution of each part of the designed rewards to the results?

The designed rewards consist of basic reward R_{bs}, greed reward R_{gd}, smoothness reward R_{sm} and granularity reward R_{gn}. Since the basic reward is fundamental and indispensable, we focus on the role of the remaining three parts of the reward in our proposed approach. Table IV demonstrates the measurement under different types of rewards.

- The comparison between $R_{\mathrm{bs}} + R_{\mathrm{gd}}$ and R_{bs} is depicted in the Fig. 7, which shows that greed reward can significantly improve the success rate and shorten the path length for each scenario under the same number of training episodes. On averaging over scenario, the success rate increases from 42.30% to 94.83% and the path length decrease from 139.45 to 96.62 after adding greed reward R_{gd}. Due to the significant effect of this kind of reward, $R_{\mathrm{bs}} + R_{\mathrm{gd}}$ is used as a comparison in subsequent experiments.
- The comparison between $R_{\mathrm{bs}} + R_{\mathrm{gd}} + R_{\mathrm{sm}}$ and $R_{\mathrm{bs}} + R_{\mathrm{gd}}$ is depicted in the Fig. 8, and shows that the success rate and the path length are almost the same, while the smoothness of path is improved in most scenarios.
- The comparison between $R_{\mathrm{bs}}+R_{\mathrm{gd}}+R_{\mathrm{gn}}$ and $R_{\mathrm{bs}}+R_{\mathrm{gd}}$ is depicted in the Fig. 9. Similar to R_{sm}, R_{gd} improves the average step length while almost unchanging the success rate and the path length. But the difference is that granularity reward has an additional contribution to the smoothness of path.
- The comparison between $R_{\mathrm{bs}} + R_{\mathrm{gd}} + R_{\mathrm{sm}} + R_{\mathrm{gn}}$ and $R_{\mathrm{bs}}+R_{\mathrm{gd}}$ is shown in the Fig. 10. By combining R_{sm} and R_{gn}, both path length and path smoothness are improved.

TABLE III
COMPARISON BETWEEN BLP AND OUR METHOD

Measurement	**Success rate**	**Path length** (km)	**Path smoothness**	**Average step length** (km)
BLP	95.00%	87.41	0.983	14.317
Our method	96.82%	94.74	0.845	2.724

TABLE IV
MEASUREMENT UNDER DIFFERENT TYPES OF REWARDS

Reward Type	R_{bs}	$R_{\mathrm{bs}}+R_{\mathrm{gd}}$	$R_{\mathrm{bs}}+R_{\mathrm{gd}}+R_{\mathrm{sm}}$	$R_{\mathrm{bs}}+R_{\mathrm{gd}}+R_{\mathrm{gn}}$	$R_{\mathrm{bs}}+R_{\mathrm{gd}}+R_{\mathrm{sm}}+R_{\mathrm{gn}}$
Success rate	42.30%	94.83%	95.63%	96.52%	96.82%
Path length	139.45	96.62	96.50	96.16	94.99
Path smoothness	83.57%	81.62%	82.90%	85.67%	84.45%
Average step length	2.06	2.37	2.54	2.73	2.72

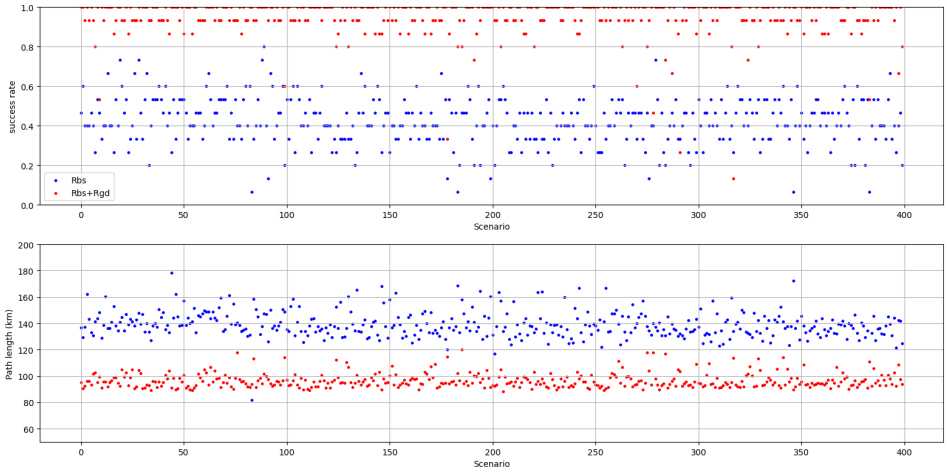

Fig. 7. $R_{\mathrm{bs}}+R_{\mathrm{gd}}$ vs R_{bs}.

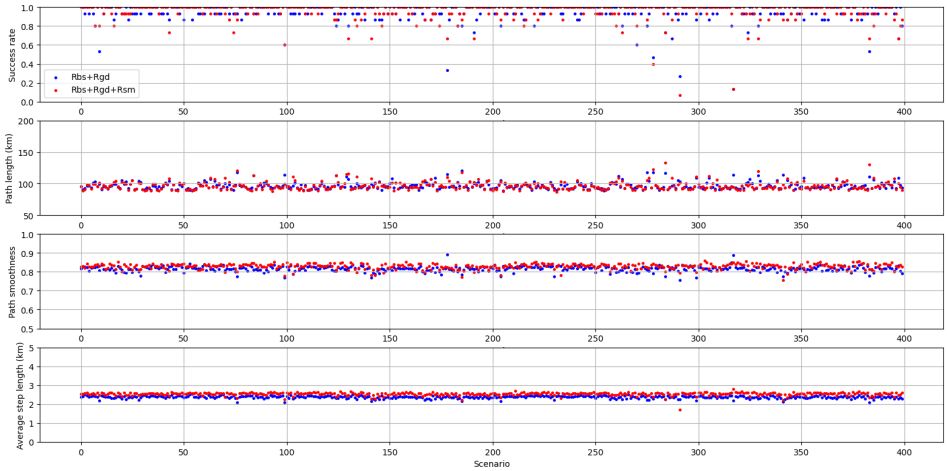

Fig. 8. $R_{\mathrm{bs}}+R_{\mathrm{gd}}+R_{\mathrm{sm}}$ vs $R_{\mathrm{bs}}+R_{\mathrm{gd}}$.

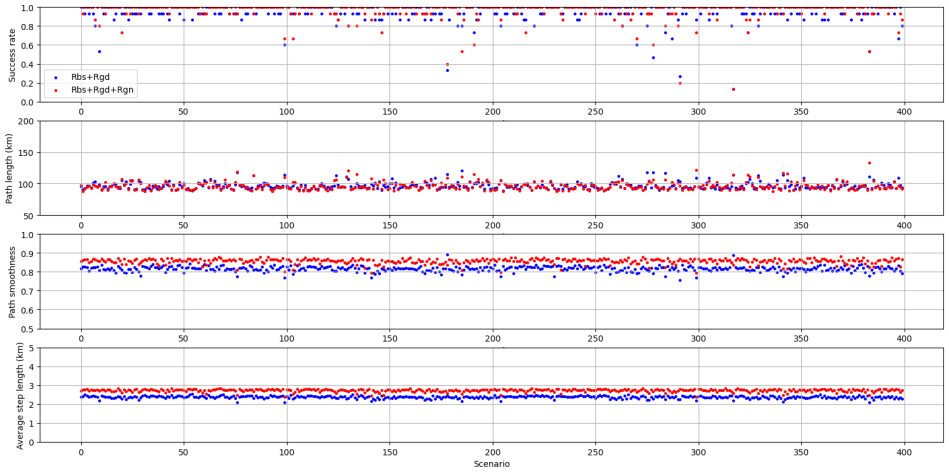

Fig. 9. $R_{\mathrm{bs}} + R_{\mathrm{gd}} + R_{\mathrm{gn}}$ vs $R_{\mathrm{bs}} + R_{\mathrm{gd}}$.

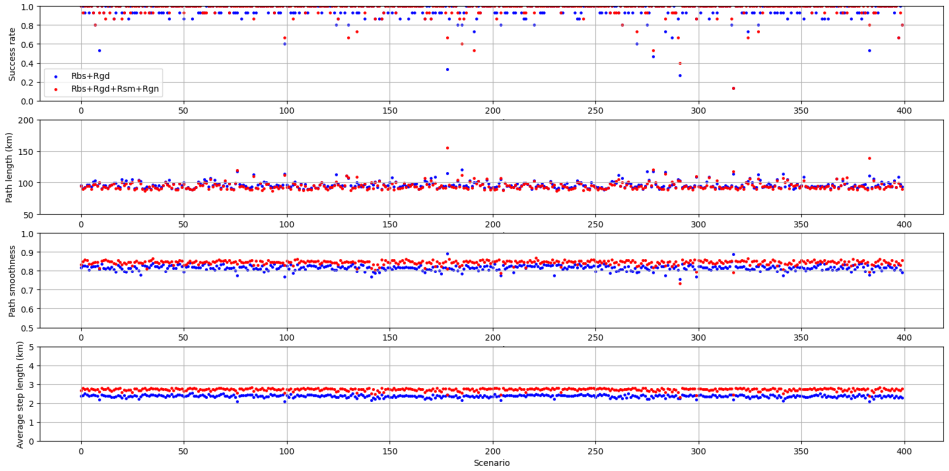

Fig. 10. $R_{\mathrm{bs}} + R_{\mathrm{gd}} + R_{\mathrm{sm}} + R_{\mathrm{gn}}$ vs $R_{\mathrm{bs}} + R_{\mathrm{gd}}$.

VI. Conclusions and Future Work

In this work, an adaptive real-time path planning method based on DRL is proposed. Taking the observation of obstacles around UAV and the positional relationship between target and UAV as the decision information, and designing the action with the consideration of the kinematic constraints of fixed-wing UAV, the proposed method can plan a feasible path for fixed-wing UAV in real-time from the origin to the target. Furthermore, combining with greedy reward, granularity reward and smoothness reward in the reward design, the designed action with adaptive ability to deal with the environment under different granularities contributes to accelerating the algorithm's convergence speed and enhancing the smoothness of the planned path. At last, it is worth mentioning that the actions and rewards designed in our method are not limited to DQN algorithm and can be applied to other DRL algorithms.

Considering that real-time path planning tasks can not be regarded as approximately satisfying the Markov property well when confronted with the scenarios containing U-shaped obstacles, future work aims to add historical decision information in decision-making to alleviate such problem. In addition, integrating the proposed method with a global planner will be considered to take advantage of the benefits of both methods.

References

[1] A. Stentz, "The focused d* algorithm for realtime replanning," *Proc. of IJCAI-95*, 1995.

[2] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong planning A," *Artif. Intell.*, vol. 155, no. 1-2, pp. 93–146, 2004. [Online]. Available: https://doi.org/10.1016/j.artint.2003.12.001

[3] S. Koenig and M. Likhachev, "D* lite," *Aaai/iaai*, pp. 476–483, 2002.

[4] I. Kamon, E. Rimon, and E. Rivlin, "Tangentbug: A range-sensor-based navigation algorithm," *International Journal of Robotic Research - IJRR*, vol. 17, pp. 934–953, 09 1998.

[5] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proceedings of the 1991 IEEE International Conference on Robotics and Automation, Sacramento, CA, USA, 9-11 April 1991*. IEEE Computer Society, 1991, pp. 1398–1404. [Online]. Available: https://doi.org/10.1109/ROBOT.1991.131810

[6] A. Pandey, R. K. Sonkar, K. K. Pandey, and D. R. Parhi, "Path planning navigation of mobile robot with obstacles avoidance using fuzzy logic controller," in *2014 IEEE 8th International Conference on Intelligent Systems and Control (ISCO)*, 2014, pp. 39–41.

[7] N. Prasad, L.-F. Cheng, C. Chivers, M. Draugelis, and B. Engelhardt, "A reinforcement learning approach to weaning of mechanical ventilation in intensive care units," 04 2017.

[8] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, pp. 1–12, 02 2016.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nat.*, vol. 518, no. 7540, pp. 529–533, 2015. [Online]. Available: https://doi.org/10.1038/nature14236

[10] M. Pfeiffer, M. Schaeuble, J. I. Nieto, R. Siegwart, and C. Cadena, "From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots," in *2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, Singapore, May 29 - June 3, 2017*. IEEE, 2017, pp. 1527–1533. [Online]. Available: https://doi.org/10.1109/ICRA.2017.7989182

[11] H. Yao, Y. Liu, and X. Zhang, "Developing deep LSTM model for real-time path planning in unknown environments," in *7th International Conference on Dependable Systems and Their Applications, DSA 2020, Xi'an, China, November 28-29, 2020*. IEEE, 2020, pp. 219–225. [Online]. Available: https://doi.org/10.1109/DSA51864.2020.00039

[12] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017, Vancouver, BC, Canada, September 24-28, 2017*. IEEE, 2017, pp. 31–36. [Online]. Available: https://doi.org/10.1109/IROS.2017.8202134

[13] T. Yan, Y. Zhang, and B. Wang, "Path planning for mobile robot's continuous action space based on deep reinforcement learning," 06 2018, pp. 42–46.

[14] C. Sampedro, H. Bavle, A. Rodriguez-Ramos, P. de la Puente, and P. Campoy, "Laser-based reactive navigation for multirotor aerial robots using deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2018, Madrid, Spain, October 1-5, 2018*. IEEE, 2018, pp. 1024–1031. [Online]. Available: https://doi.org/10.1109/IROS.2018.8593706

[15] C. Wang, J. Wang, Y. Shen, and X. Zhang, "Autonomous navigation of uavs in large-scale complex environments: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2124–2136, 2019. [Online]. Available: https://doi.org/10.1109/TVT.2018.2890773

[16] Y. Zhao, J. Guo, C. Bai, and H. Zheng, "Reinforcement learning-based collision avoidance guidance algorithm for fixed-wing uavs," *Complex.*, vol. 2021, pp. 8 818 013:1–8 818 013:12, 2021. [Online]. Available: https://doi.org/10.1155/2021/8818013

[17] R. Xie, Z. Meng, L. Wang, H. Li, K. Wang, and Z. Wu, "Unmanned aerial vehicle path planning algorithm based on deep reinforcement learning in large-scale and dynamic environments," *IEEE Access*, vol. 9, pp. 24 884–24 900, 2021. [Online]. Available: https://doi.org/10.1109/ACCESS.2021.3057485

[18] Z. Cui and Y. Wang, "UAV path planning based on multi-layer reinforcement learning technique," *IEEE Access*, vol. 9, pp. 59 486–59 497, 2021. [Online]. Available: https://doi.org/10.1109/ACCESS.2021.3073704

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[20] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. J. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Józefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. B. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *CoRR*, vol. abs/1603.04467, 2016. [Online]. Available: http://arxiv.org/abs/1603.04467

[21] W. Liu, Z. Zheng, and K. Cai, "Bi-level programming based real-time path planning for unmanned aerial vehicles," *Knowl. Based Syst.*, vol. 44, pp. 34–47, 2013. [Online]. Available: https://doi.org/10.1016/j.knosys.2013.01.011