Three-dimensional Path Planning Method of Agent Based on Fluid Disturbance Algorithm and PPO

Zhixin Gu, Keyi Jia and Kaihong Xu

Abstract—**In response to the dynamic challenges posed by movable obstacles and unpredictable environmental shifts in three-dimensional (3D) settings, we propose a hybrid algorithm that combines the fluid disturbance algorithm with Proximal Policy Optimization (PPO). This innovative approach utilizes an enhanced fluid disturbance method to generate an initial path scheme, ensuring adherence to the agent's kinematic model and constraints. Subsequently, PPO's interactive learning capabilities are employed to optimize and adjust the fluid disturbance parameters, thereby refining the initial path for optimal decision-making. The design of this algorithm aims to leverage the strengths of both constituent algorithms while compensating for their individual weaknesses, ultimately enhancing path planning efficiency, ensuring smooth paths, and improving obstacle avoidance success rate and pathfinding. Simulation results consistently demonstrate that this algorithm meets specified requirements across different random seeds, showcasing robust stability and adaptability in 3D dynamic environments with significant practical value.**

Index Terms—**Path planning, Fluid disturbance algorithm, Proximal policy optimization, Agent.**

I. Introduction

The path planning for autonomous agents represents a significant research focus, with broad applications in national defense, emergency response, logistics, and traffic management[1]. While the path planning in static environments has been extensively studied and achieved a certain level of maturity, navigating dynamic environments with moving obstacles has emerged as a critical topic in contemporary research. Addressing autonomous obstacle avoidance and path planning in such contexts is not only a current research priority but also a key challenge in advancing the era of intelligent systems.

Some researchers have endeavored to enhance traditional obstacle avoidance algorithms to address the challenge of dynamic obstacle avoidance for autonomous agents. For instance, Yong Zhang et al.[2] have enhanced the Artificial Potential Field (APF) method, achieving path planning and tracking for unmanned vehicles. Na Liu et al.[3] have introduced an optimized A* algorithm for the path planning of Automated Guided Vehicles (AGVs), while Meilin Kang et al.[4] have adapted the Rapidly-exploring Random Tree (RRT) algorithm for robotic path planning in intricate orchard settings. Swarm intelligence algorithms have also been harnessed, with Thi Kien Dao et al.[5] refining Particle Swarm Optimization (PSO) for AGV path planning and Fengcai Huo et al.[6] proposing an improved Ant Colony Optimization (ACO) algorithm for mobile robots. Luo Yuan et al.[7] have contributed to the field by enhancing the Grey Wolf Optimization (GWO) algorithm for unmanned delivery robots. Nonetheless, these algorithms, while predominantly effective in 2D spaces, confront scalability issues in 3D contexts, where they may exhibit increased computational complexity and diminished path smoothness.

In response to the challenges posed by 3D environments, scholars are actively exploring innovative solutions. Honglun Wang et al.[8] have introduced a suite of fluid calculation methods inspired by the natural phenomenon of water flowing around obstacles, leading to the development of the Interfered Fluid Dynamical System (IFDS). This approach is renowned for its computational efficiency and the smoothness it offers in planned paths, making it highly applicable across various domains. However, the algorithm's reliance on multiple parameters presents both an opportunity and a challenge as finding their optimal configuration is crucial for achieving optimal performance. The adaptability of these parameters plays a pivotal role in effectively navigating through complex 3D spaces.

In addition, many researchers have increasingly integrated Deep Reinforcement Learning (DRL) into agent path planning. For example, Yang Xiao et al.[9] optimized the Deep Q-Network (DQN) algorithm for ship navigation, and Yuanhang Chen et al.[10] enhanced the Soft Actor-Critic (SAC) algorithm for autonomous vehicle obstacle avoidance. Moreover, Shuhuan Wen et al.[11] refined the PPO algorithm for multi-robotic pathfinding. Compared to previous algorithms, DRL's strength lies in its ability to approximate optimal actions and extract features aided by deep neural networks[12]. However, the algorithm's dependency on experiential learning can result in convergence delays. To ameliorate this issue, we propose an integrated IFDS-PPO approach designed to expedite the learning process and enhance path planning efficiency.

II. Agent and Environmental Modeling

A. Kinematic model and constraints of the agent

Given that the agent's control system maintains its posture and velocity stability, the agent's kinematic model[13] under these conditions is presented as formula (1).

$p=(x,y,z)$ denotes the agent's position in 3D space; γ is the trajectory inclination angle; φ indicates the trajectory deviation angle; V represents the agent's velocity; n_x n_y n_z are the control inputs of the agent, representing the overload along the x,y,z axis in the coordinate system; g

Manuscript received July 3, 2024; revised January 18, 2025.

This work was supported in part by the fund project: National Key Research and Development Plan of China (2022YFD1301104).

Zhixin Gu is an associate professor of Northeast Forestry University in Harbin, Heilongjiang Province, China. (e-mail: gzx@nefu.edu.cn).

Keyi Jia is a postgraduate student of Northeast Forestry University in Harbin, Heilongjiang Province, China. (e-mail: 945629100@qq.com).

Kaihong Xu is a professor of Northeast Forestry University in Harbin, Heilongjiang Province, China. (corresponding author, phone: 13936548825; e-mail: 323224311@qq.com).

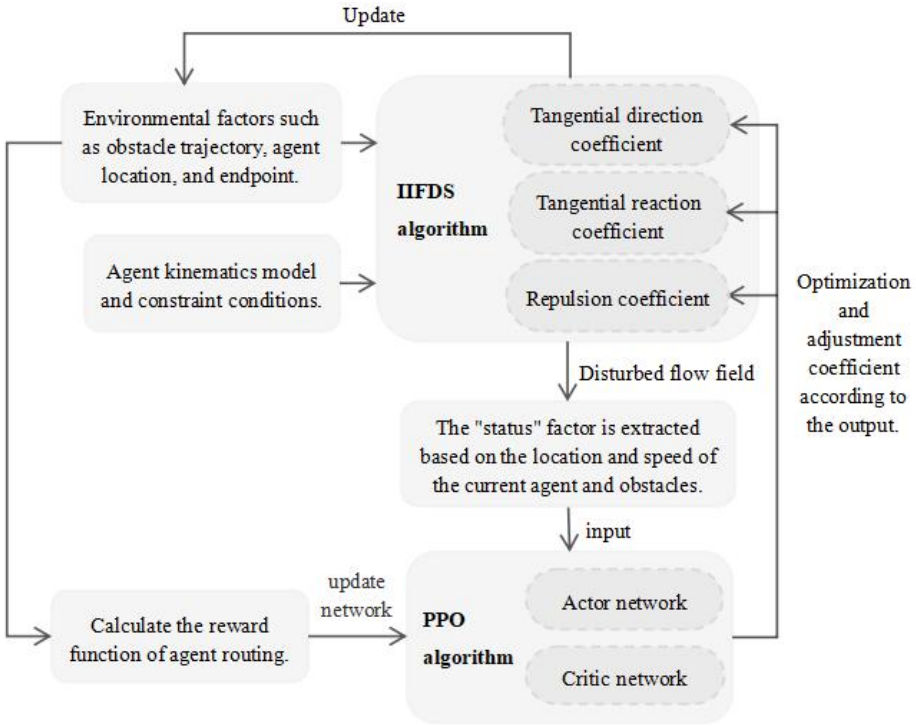

Fig. 1. Architecture diagram of algorithm implementation.

denotes gravitational acceleration.

$$\begin{cases} \dot{x} = V\cos\gamma\cos\varphi \\ \dot{y} = V\cos\gamma\sin\varphi \\ \dot{z} = V\sin\gamma \\ \dot{V} = (n_x - \sin\gamma)g \\ \dot{\gamma} = \dfrac{(n_z - \cos\gamma)g}{V} \\ \dot{\varphi} = \dfrac{n_y g}{V\cos\gamma} \end{cases} \tag{1}$$

Concurrently, the aforementioned kinematic model is constrained by the following conditions[13] as follows: $\gamma_{\min} \le \gamma \le \gamma_{\max}$; $|\dot{\gamma}| \le \dot{\gamma}_{\max}$; $|\dot{\varphi}| \le \dot{\varphi}_{\max}$; $n_{i\min} \le n_i \le n_{i\max}$ ($i = x, y, z$).

B. Obstacle model

To facilitate real-time trajectory planning for the agent, obstacles are commonly approximated as standard convex polyhedra, including spheres, cylinders, cones, and parallelepipeds[14], with the geometric equation defined as follows:

$$\Gamma(P) = \left(\frac{x - x_0}{a + R_t}\right)^{2l} + \left(\frac{y - y_0}{b + R_t}\right)^{2m} + \left(\frac{z - z_0}{c + R_t}\right)^{2n} \tag{2}$$

In the formula, $a, b, c > 0$ and $l, m, n > 0$ determine the scale and geometry of the obstacle respectively. Specifically, when conditions $l = m = n = 1$ and $a = b = c$ are satisfied, the obstacle is a sphere with a radius of a (in this paper, all obstacles are represented as spheres); $p = (x_0, y_0, z_0)$ is the center coordinate of the obstacle; R_t indicates the agent's safety radius during navigation. $\Gamma(P) < 1$, $\Gamma(P) = 1$, and $\Gamma(P) > 1$ represent the obstacle's interior, boundary, and exterior regions respectively. Throughout trajectory planning, it is crucial to prevent agents from entering the obstacle's interior or coming into contact with its boundary.

III. The Framework of IIFDS-PPO2 Algorithm

A. Overall overview

The IFDS is recognized for its computational efficiency and the smoothness of its planned paths, yet it is not without flaws, such as susceptibility to local minima and a complex parameter configuration. Conversely, PPO is lauded for its autonomous learning capabilities, operational simplicity, and manageable computational demands, but it can be hindered by slow convergence and a propensity for deadlocks. This study, therefore, proposes an integration of PPO with IFDS to synergize their strengths and mitigate their weaknesses, thereby enhancing the overall path planning performance.

To mitigate the susceptibility of the IFDS algorithm to local minima, we propose an enhancement, termed the Improved IFDS (IIFDS). This enhancement is integrated with the agent's kinematic model and constraints to enhance the practicality and reliability of path planning. Subsequently, IIFDS is synergized with the PPO algorithm, leveraging the former for generating disturbed flow fields indicative of navigable paths within the environment, and the latter for fine-tuning the reaction and direction coefficients of obstacles. The integrated approach calculates a comprehensive disturbance matrix to refine the agent's trajectory, ensuring real-time obstacle avoidance in an iterative process. The overall framework is depicted in Fig. 1.

B. Path planning based on IIFDS algorithm

1) IFDS principle

The IFDS algorithm emulates the natural phenomenon of fluid flow around obstacles, utilizing a vector representing the direct path from the agent's starting point to its destination as the initial flow field. This field is designed such that, in unobstructed scenarios, the agent can follow any

streamline to its target. However, when obstacles are present, this initial flow is disturbed. The IFDS algorithm quantifies this disturbance, generating a disturbance matrix that accounts for the obstacles' influence. By applying this matrix, the initial flow field is adjusted, resulting in a modified flow field that delineates the agent's obstacle-avoiding trajectory.

Let the agent's position be denoted by P, its velocity by v_0, and the end point of path planning by $P_d = [x_d, y_d, z_d]^T$. In an unobstructed environment, the flow velocity of streamline in the initial flow field is designated as $u(P)$ (called initial flow velocity):

$$u(P) = \begin{bmatrix} \frac{v_0(x_d - x)}{\|P - P_d\|} & \frac{v_0(y_d - y)}{\|P - P_d\|} & \frac{v_0(z_d - z)}{\|P - P_d\|} \end{bmatrix}^T \quad (3)$$

In the presence of obstacles, their impact on the initial flow velocity is expressed by the disturbance matrix:

$$M = \sum_{i=1}^{I} \omega_i M_i \quad (4)$$

In the formula, ω_i is the weight coefficient of the obstacle *i*, and M_i is the corresponding disturbance matrix, formulated as follows:

$$M_i = E + \frac{1}{|\Gamma_i|^{\frac{1}{\rho_i}} n_i^T n_i} \left(n_i^T n_i E - 2 n_i n_i^T \right) \quad (5)$$

The formula is bifurcated into two components: The first, *E*, represents the attraction matrix, which is a third-order identity matrix; The second, constitutes the repulsion matrix, with ρ_i being the repulsion coefficient for obstacle *i*, and n_i representing the unit normal vector of obstacle *i*.

Subsequently, the initial flow velocity is adjusted based on the disturbance matrix *M*, and $\overline{u}$, the flow velocity of the disturbed flow field can be obtained:

$$\overline{u} = Mu \quad (6)$$

Finally, the next path point can be obtained by integrating $\overline{u}$:

$$P_{t+1} = P_t + \overline{u} \cdot \Delta T \quad (7)$$

In the formula, ΔT is the time interval between *t* and *t* + 1. By iterating the above series of operations, a set of discrete path planning points is derived, which, when connected, form the streamline of disturbed flow field, thus constituting the final planned path.

2) IFDS in dynamic environment

The preceding discussion pertains to the IFDS path planning within static environments. In the context of dynamic environments, the relative velocity between the agent and moving obstacles can be factored into the model, effectively converting the dynamic path planning problem into a static one for resolution[15].

In the presence of a dynamic spherical obstacle, its future state at time *t* + 1 is projected based on its current state at time *t*, with the obstacle's reference velocity at time *t* + 1 designated as *v*:

$$v = e^{\frac{-1}{\lambda}(\Gamma(P)-1)} v^s \quad (8)$$

In the formula, $\lambda > 0$, and the larger it is, the earlier the agent starts to avoid obstacles; v^s Indicates the predicted value of the actual moving speed of the obstacle.

Concurrently, the relative initial flow field must be defined, with its velocity designated as $u - v$; similarly, the velocity of the relative disturbed flow field is $\overline{u} - v$. Within the framework of the relative initial flow field, the dynamic obstacle can be treated as stationary, thereby allowing us to deduce the actual velocity of the disturbed flow field:

$$\overline{u} = M(u - v) + v \quad (9)$$

In the presence of multiple moving obstacles within the dynamic environment, equation (8) can be changed to (10).

$$v_i = e^{\frac{-1}{\lambda}(\Gamma_i(P)-1)} \omega_i v_i^s \quad (10)$$

Subsequently, the resultant velocity can be obtained after weighted summation of the velocities of multiple obstacles :

$$v = \sum_{i=1}^{I} \omega_i e^{\frac{-1}{\lambda_i}(\Gamma_i(P)-1)} v_i \quad (11)$$

Finally, the flow velocity of the actual disturbed flow field is obtained by incorporating the calculated values into equation (9).

3) IIFDS

The IFDS algorithm encounters limitations, notably a propensity for local minima, attributed to an incomplete specification of the disturbance matrix *M*. To address this, the algorithm can be enhanced by incorporating an additional "tangential matrix"[16] into matrix *M*.

Taking obstacle *i* as an example, take two mutually perpendicular vectors in the tangent plane perpendicular to its unit normal vector n_i, denoted as $l_{i,1} \perp l_{i,2}$.

Define axes x', y', z' as $l_{i,1}$, $l_{i,2}$ and n_i respectively to construct the coordinate system $o' - x'y'z'$, then any unit tangent vector within the tangent plane can be expressed in this coordinate system as follows:

$$l_i' = [\cos\theta_i \quad \sin\theta_i \quad 0]^T \quad (12)$$

Where, $\theta_i \in [-\pi, \pi]$ and it is the angle between this vector and the vector $l_{i,1}$ (or x' axis), termed the tangential direction coefficient.

Subsequently, by conducting a coordinate transformation, the expression l_i of l_i' in $o - xyz$ can be ascertained. Consequently, the tangential matrix can be articulated as:

$$M_i^{\perp} = \frac{l_i n_i^T}{|\Gamma_i|^{\frac{1}{\sigma_i}} \|l_i\| \|n_i\|} \quad (13)$$

Equation (14) defines the disturbance matrix *M* for the IIFDS, incorporating three key adjustable parameters: the repulsion coefficient ρ_i, the tangential reaction coefficient σ_i, and the tangential direction coefficient θ_i.

$$M_i = E - \frac{n_i n_i^T}{|\Gamma_i|^{\frac{1}{\rho_i}} n_i^T n_i} + \frac{l_i n_i^T}{|\Gamma_i|^{\frac{1}{\sigma_i}} \|l_i\| \|n_i\|} \quad (14)$$

Incorporating tangential matrix into *M* has been demonstrated to enhance the distribution of streamlines without compromising their inherent characteristics, effectively addressing the issue of local minima.

C. Implementation of PPO algorithm

The PPO algorithm[17] is adept at handling scenarios with continuous actions and high-dimensional states, training the

agent to maximize cumulative rewards by optimizing its policy. The algorithm processes the agent's current state through a neural network to determine actions and their expected rewards. Subsequently, it updates the agent's state based on these actions and refines the neural network's weight parameters using an objective function that incorporates the rewards and actions, thereby enhancing the policy to achieve higher overall rewards.

1) Neural network

In the PPO algorithm, the Actor network is a neural network used to generate actions (a), which takes the environment states (s) as input and outputs the probability of each action. Essentially, the Actor's goal is to select actions based on the current policy and in collaboration with the critic network, refine the policy to optimize performance. The network structure[18] of Actor in this paper is shown in Fig. 2.

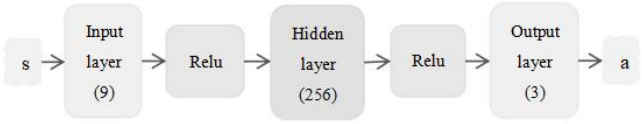

Fig. 2. Actor network architecture.

The function of the Critic network is to evaluate the efficacy of the current policy and furnish the actor network with the value estimates and feedback for policy refinement. By estimating the value of a given state (or state-action), the network helps determine the long-term return of an agent taking an action in that state. It usually outputs a state value function (V(s)) or an action value function (Q(s, a)). Its network structure in this paper is shown in Fig. 3.

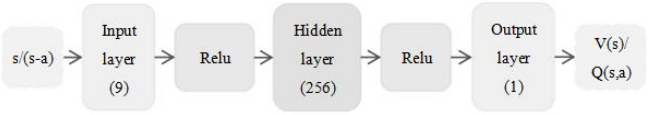

Fig. 3. Critic network architecture.

2) The variant PPO algorithm

One of the core of PPO algorithm is the policy gradient update, with two prevalent approaches: PPO-Penalty and PPO-Clip. These methods primarily serve to constrain the magnitude of policy gradient adjustments, leading to distinct neural network parameter update strategies. Empirical evidence suggests that PPO-Clip is more effective and less complex than PPO-Penalty. Consequently, this study employs the PPO-Clip method[19], also referred to as the PPO2 algorithm, for its implementation.

The objective of PPO2 is to refine the policy while constraining the scope of policy updates to prevent excessive alterations that could lead to significant policy shifts. The neural network parameters are adjusted based on the subsequent formula:

$$L^{Clip}(\theta)=\hat{E}_t\left[\min\left(r_t(\theta)\hat{A}_t, clip\left(r_t(\theta),1-\varepsilon,1+\varepsilon\right)\hat{A}_t\right)\right] \quad (15)$$

Where θ is the policy parameter; $\hat{E}_t$ is the empirical expectation of the time step; r_t is the probability ratio between the new and the old policy; $\hat{A}_t$ is the advantage estimation at time t (GAE); ε is a hyper-parameter. With this formula, it can be seen that the essence of PPO2 is to limit the difference between the old and new actions to [1 - ε , 1 + ε].

3) Parameter settings

The key parameter settings for the PPO2 algorithm in this paper are presented in Table I.

TABLE I
KEY PARAMETERS VALUES FOR THE PPO2 ALGORITHM

Parameter name	Settings	Parameter name	Settings
Optimizer	Adam	Episode	4000
Learning rate	1e-4	Clipping parameter	0.25
Discount factor	0.99	Epochs	8
Experience replay buffer	2^{12}	Batch size	128

D. Implementation of IIFDS - PPO2 algorithm

1) State value function

Given the complexity of the algorithmic environment in this study, which encompasses critical elements such as agents, dynamic obstacles, and end points, the position of agent cannot be simply utilized as a state input in the neural network. Accordingly, this paper takes the relative coordinates and velocities among the agent, obstacles and end point as the input state, which has three parts:

① The positional discrepancy between the agent and the closest point on the obstacle's boundary;

② The positional discrepancy between the agent and the destination;

③ The velocity of the obstacles.

2) Action value function

This paper's algorithm employs an indirect approach to action selection, wherein the Actor network's output is sampled from a normal distribution and subsequently normalized to the interval (-1, 1) using the hyperbolic tangent function (tanh). To regulate the unidirectional movement of the agent, these values are subsequently transformed to a predefined range (0.1 to 3), which is crucial for constraining the agent's trajectory within operational limits.

Through the above steps, we can get the action value, which in this algorithm is $(\rho_0,\ \sigma_0,\ \theta)$, it is closely related to the values of the repulsion coefficient, tangential reaction coefficient and tangential direction coefficient in IIFDS, and this operation is also the key to merging the IIFDS with the PPO2 algorithm.

3) Reward function

In reinforcement learning projects, the design of the reward function is as critical as the algorithm configuration, given its essential role in updating parameters associated with states and actions. The reward function delineates the objectives that the agent aims to achieve during learning; it not only influences the efficiency of the agent's learning but also directly affects the agent's performance in complex environments and the optimization of its ultimate policy. The reward function is designed as follows:

·$reward = 0$

·if intersects with an obstacle

· $reward += \frac{d_{a\to o} - R}{R} - 1$

·else (no intersection)

· if $d_{a\to o} < R + 0.4$

· $tempR = R + 0.4$

· $reward += \frac{d_{a\to o} - tempR}{tempR} - 0.3$

· if $d_{a\to g} > threshold$

· $reward += -\frac{d_{a\to g}}{d_{s\to g}}$

· else

· $reward += -\frac{d_{a\to g}}{d_{s\to g}} + 3$

·Return *reward*

Where $d_{a\to o}$, $d_{a\to g}$ and $d_{s\to g}$ are respectively the distance from the agent to the obstacle center, the distance from the agent to the end point, and the distance from the starting point to the end point; R is the radius of the obstacle; 0.4 is the threat area; *threshold* is the threshold value, which is set as 0.2 in this paper, that is, when $d_{a\to g}$ is less than 0.2, it is regarded as reaching the end point.

IV. Simulation and Analysis

A. Comparative analysis settings

This paper compares the proposed method with two novel algorithms in this domain, namely APF-PPO2 and IIFDS-SAC. The concept of the APF algorithm shares similarities with IFDS, where the gravitational field in APF is analogous to the initial flow field in IFDS, and the repulsive force field corresponds to the disturbed flow field. Additionally, the Soft Actor-Critic (SAC)[20], as an algorithm based on policy optimization and value function learning, has emerged as one of the prominent algorithms in the field of reinforcement learning in recent years.

The three algorithms will be trained and verified under identical conditions to ensure the comparability of the experimental outcomes. Additionally, different random seeds will be employed during algorithm training to effectively enhance control, reproducibility, and impartiality in the experiment setup.

B. Environment settings

1) Training environment

This study constructs nine distinct scenarios with single dynamic obstacles (the code names are ①~⑨). The obstacles' trajectories encompass a variety of paths such as straight lines, arcs, circles, ellipses, and wave patterns. To optimize obstacle utilization in agent path planning without venturing beyond the planning area, the paper confines the trajectories of certain obstacles, like those moving in straight lines or waves, to cycle within a designated region. This approach enhances obstacle engagement in path planning, ensuring both the complexity of the simulation environment and the practicality of the path planning outcomes.

2) Test environment

This study employs a multi-dynamic obstacle environment to assess the performance of the algorithms. The test environment comprises a combination of single dynamic obstacles from the preceding section, with six distinct configuration states selected. Each simulation features three dynamic spherical obstacles. During the path planning assessment, the algorithm traverses six multi-dynamic obstacle test environments, accumulating the reward function scores for each. The algorithm's path planning efficacy is ultimately determined by averaging the reward function values across these six environments.

C. The effect of path planning

Upon observing 10 sets of random seeds, it was observed that the algorithms' performances across different seeds exhibited minimal variance. Consequently, this study presents a comparative analysis featuring only the best group of random seed data in each algorithm together for comparative analysis.

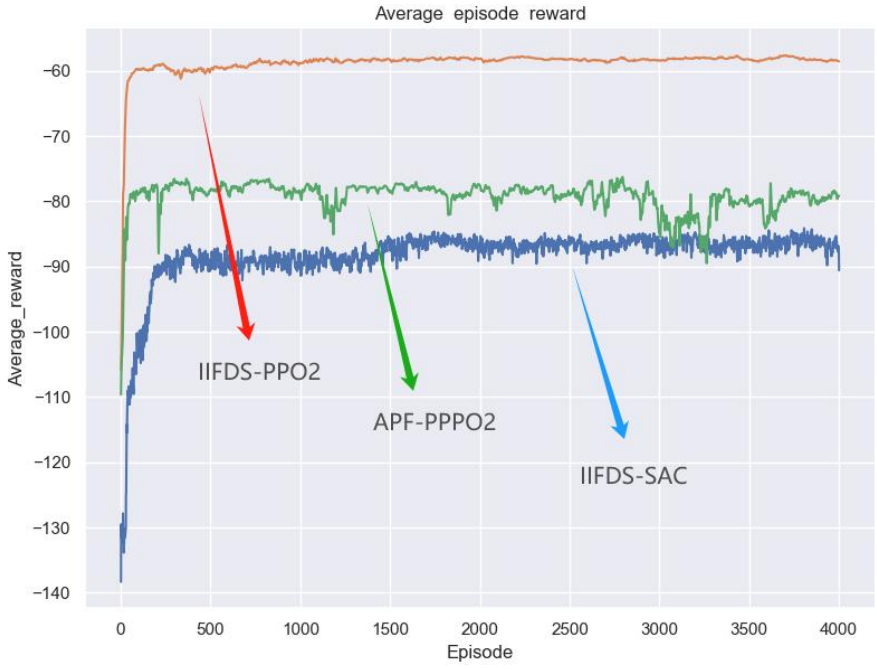

Fig. 4. Episode - Average reward.

1) Comparison and analysis of average reward value

The reward function serves as a critical metric for assessing the performance of path planning algorithms. As indicated by the analysis of the reward function, a higher reward value signifies that the agent receives more rewards after executing a specific action in a given state, implying that the action is more beneficial for achieving the goal. As shown in Fig. 4, a comparison of the average reward values among three algorithms in a multi-obstacle environment reveals significant performance differences. The proposed algorithm demonstrates exceptional performance, with its average reward value converging rapidly within approximately 100 episodes and stabilizing at a level of around -56, showing a marked advantage over other algorithms. This result not only confirms the rapid convergence characteristic of the proposed algorithm but also reflects its outstanding pathfinding capability in complex environments.

2) Comparison and analysis of loss function

The loss function is a pivotal indicator in training processes and algorithmic performance, with relevant graphical representations provided in in Fig. 5. Here, loss_V and loss_Q denote the value function loss and the Q-function loss, respectively, assessing the discrepancy between the estimated and actual values of the functions. Meanwhile, loss_pi represents the policy gradient loss, quantifying the deviation of the current policy from the target behavior.

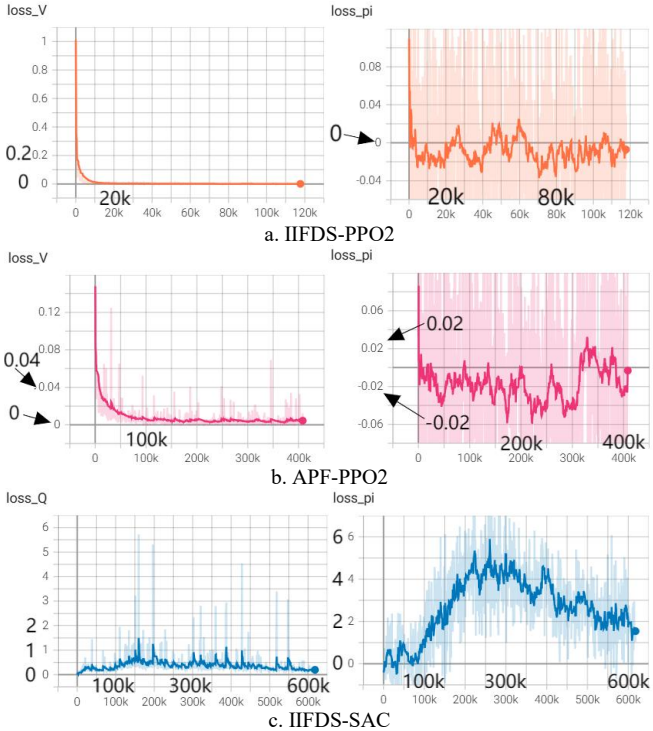

Fig. 5. The loss function images of each algorithm.

Fig. 5 displays smoothed images to provide a clearer view of the performance and trends for each loss function. In general, the loss values of each loss function exhibit a downward trend as the training steps. Notably, the loss_pi of the IIFDS-SAC algorithm fluctuates greatly, but also has a tendency to converge.

By observing the absolute values of each loss function, it can be seen that IIFDS-PPO2 algorithm emerges the most effective in value estimation and policy refinement, with a more rapid convergence rate. Furthermore, the IIFDS-PPO2 algorithm's loss curve is notably smoother, suggesting superior training stability.

3) Effect and analysis of path planning in single dynamic obstacle environment

Under the condition that the parameters for all three algorithms are identical, we conducted 4000 episodes to select the batch that demonstrated enhanced path planning performance. The critical parameters were meticulously logged to optimize the configuration for the respective DRL models. The efficacy of the agent's path planning was then evaluated in single dynamic obstacle settings using the trained algorithms. This paper showcases and dissects four exemplary simulation outcomes to illustrate the agent's pathfinding capabilities.

In the evaluation of the IIFDS-PPO2 algorithm, the agent successfully avoided collisions. Conversely, the APF-PPO2 algorithm resulted in collisions within environment ①, while the agent under IIFDS-SAC algorithm encountered obstacles in environment ⑧. Despite these differences, all agents managed to reach their designated targets, with the IIFDS-PPO2 algorithm leading in terms of arrival time, followed by APF-PPO2 and IIFDS-SAC respectively.

The performance metrics, including reward values and path lengths, for the agent across four distinct environments using the three algorithms are summarized in Tables II and III.

Table II reveals that the IIFDS-PPO2 algorithm achieves the optimal reward, indicating its superior performance in obstacle avoidance. This is further corroborated in Fig. 6, which illustrates the agent's proactive strategy in obstacle avoidance, thereby enhancing safety. Additionally, a comprehensive analysis of Fig. 6 and Table III reveals that the IIFDS-SAC algorithm exhibits longer path lengths and more frequent directional changes compared to other algorithms, suggesting its inferior path planning capabilities. In contrast, while the IIFDS-PPO2 and APF-PPO2 algorithms show similarities in terms of path smoothness and length, the occurrence of collisions with the APF-PPO2 algorithm raises safety concerns. Therefore, considering all factors, the IIFDS-PPO2 algorithm outperforms other algorithms in terms of obstacle avoidance, path efficiency, and overall navigability in a single dynamic obstacle scenarios, which holds significant practical implications.

TABLE II
REWARD VALUE OF THE PLANNED PATH UNDER SINGLE DYNAMIC OBSTACLE ENVIRONMENT ①, ②, ④ AND ⑧

Environment	Algorithm		
	IIFDS-PPO2	APF-PPO2	IIFDS-SAC
①	-50.782606	-84.429145	-80.894087
②	-57.988665	-80.980666	-88.849673
④	-56.344449	-83.409442	-83.343272
⑧	-57.507263	-88.855071	-105.137645

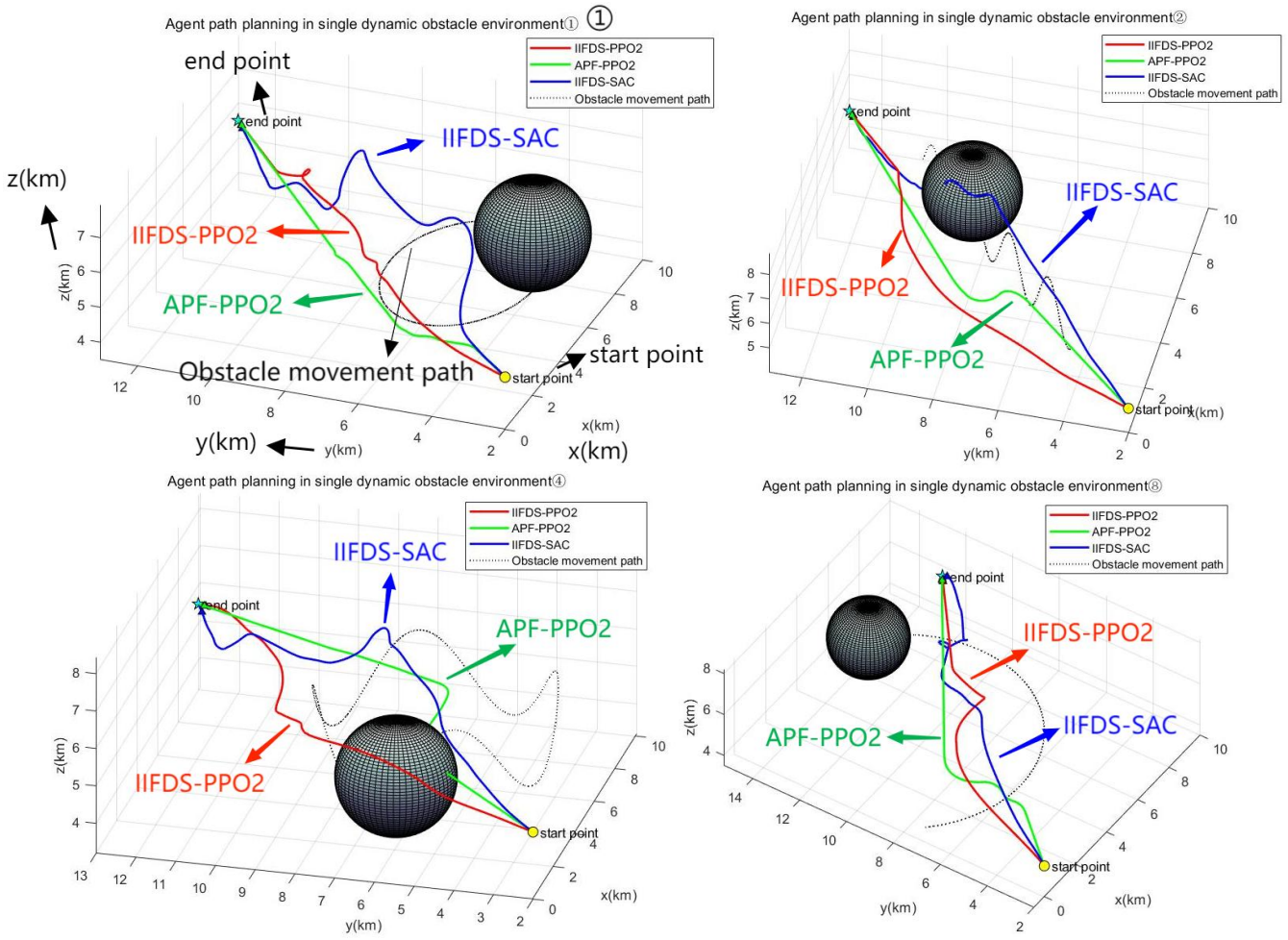

Fig. 6. Effect of path planning in single dynamic obstacle environment①, ②, ④ and ⑧.

TABLE III
THE LENGTH OF THE PLANNED PATH UNDER SINGLE DYNAMIC OBSTACLE ENVIRONMENT ①, ②, ④ AND ⑧

Environment	Algorithm		
	IIFDS-PPO2	APF-PPO2	IIFDS-SAC
①	16.603384	15.568597	19.086940
②	16.033619	16.503827	17.766747
④	16.881620	16.228919	17.550134
⑧	16.182155	16.528663	19.801096

TABLE IV
STATISTICAL RESULTS OF AVERAGE DATA OF EACH INDICATOR

Algorithm	IIFDS-PPO2	APF-PPO2	IIFDS-SAC
reward value	-57.597645	-75.907476	-84.813530
path length	16.214076	15.336330	17.579840
steps	125	153	180
collision probability	0/6	2/6	0/6
probability of entering the threat zones	0/6	2/6	1/6

4) Effect and analysis of path planning in multiple dynamic obstacles environment

Considering the complexity of real-world environments often surpassing that of single dynamic obstacle scenarios, it is essential to assess the adaptability of trained algorithms to settings with multi-dynamic obstacles. The corresponding simulation results are depicted in Fig. 7. It should be noted that, consistent with the arrival order observed in single obstacle environments, the IIFDS-PPO2 and IIFDS-SAC algorithms' agents rank first and last, respectively.

Table IV presents the average statistical data of key path planning indicators across six environments as illustrated in Fig. 7, which include reward value, path length, number of steps, collision probability, and the probability of entering the threat zones of obstacles. This data facilitate a comprehensive analysis of the path planning performance of each algorithm under varying conditions.

By analyzing Fig. 7 and Table IV, it can be concluded that the paths planned by the APF-PPO2 algorithm, while having fewer turns and the shortest average path length, are associated with lower average reward and a high collision risk. This indicates that the APF-PPO2 algorithm sacrifices safety performance in order to ensure the brevity and directness of the path, leading to unnecessary cost losses in practice and diminishing practical significance. In contrast, although the IIFDS-PPO2 algorithm has a slightly longer average path length than the APF-PPO2 algorithm, its zero collision rate and the fact that it never enters the threat zone of obstacles indicate a good balance between pathfinding safety and efficiency. This is also reflected in its higher average reward and fewer steps, suggesting that the IIFDS-PPO2 algorithm is a safer and more sustainable path

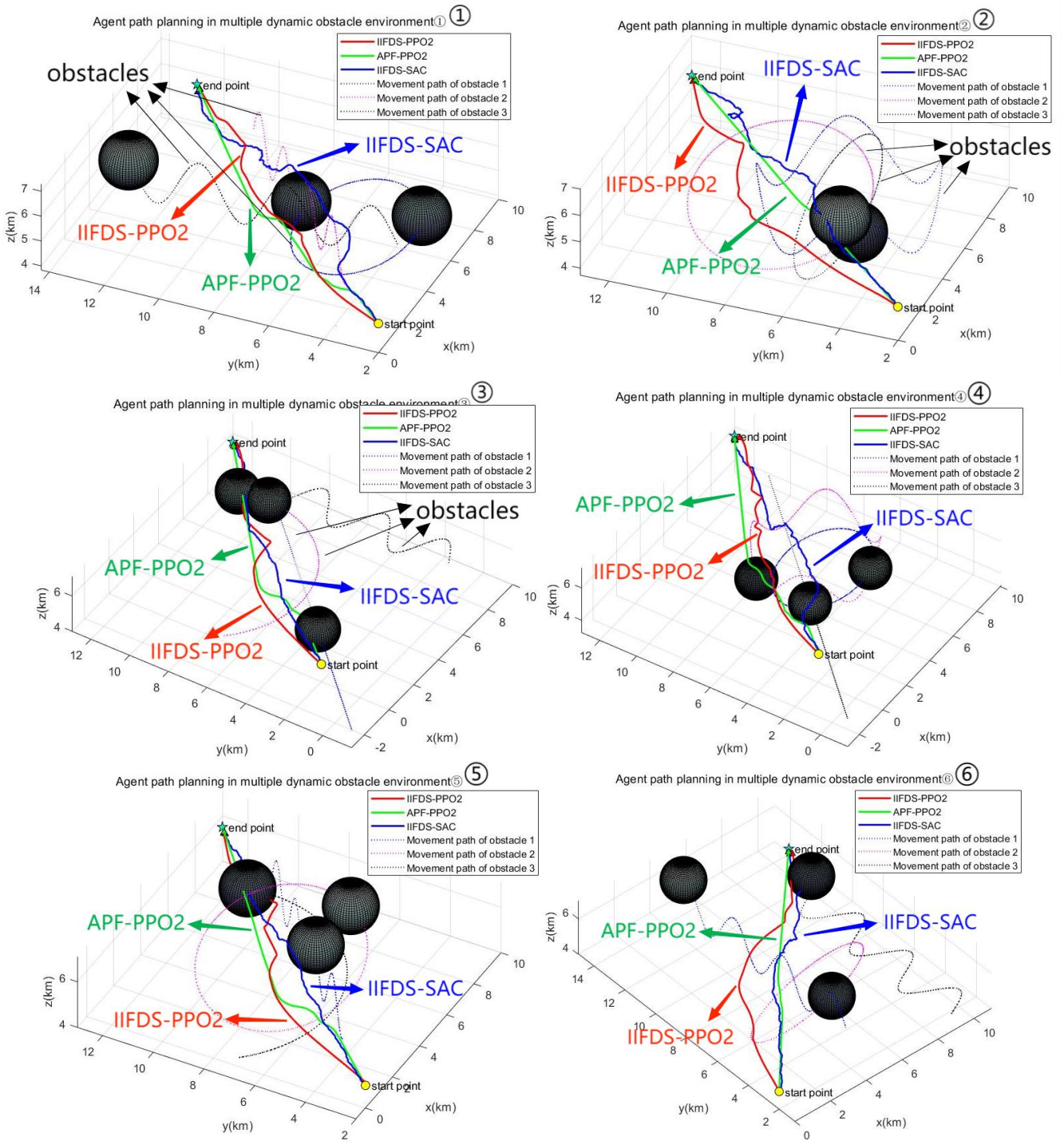

Fig. 7. Effect of path planning in multiple dynamic obstacle environments.

planning method.

Additionally, although the IIFDS-SAC algorithm plans paths with relatively high safety, it occasionally enters obstacle threat zones during pathfinding and performs poorly in terms of average reward, path length, and steps. By examining the loss function in Fig. 5, it is indicates that the algorithm's training stability and exploratory ability require enhancement when confronted with complex environments.

V. Conclusion

This study integrates the IIFDS and PPO2 algorithms to tackle the challenge of path planning in dynamic 3D obstacle environments. By combining the obstacle avoidance capabilities of IIFDS with the interactive learning abilities of PPO2, the approach achieve real-time, dynamic obstacle avoidance. In summary, the IIFDS-PPO2 algorithm capitalizes on the complementary strengths of both: the sophisticated obstacle-avoidance capabilities of the IIFDS and the interactive adaptability of the PPO2. This synergy effectively mitigates the slow convergence and susceptibility to deadlocks of the PPO2 algorithm, while also tackling the parameter tuning challenges of the IIFDS algorithm.

To substantiate the efficacy of the IIFDS-PPO2 algorithm, this paper conducted empirical analyses and comparative evaluations against the APF-PPO2 and IIFDS-SAC. The findings consistently demonstrate the superiority of the IIFDS-PPO2 algorithm, highlighting its robustness, real-time responsiveness, superior path smoothness, and enhanced reliability in obstacle avoidance. Moreover, the IIFDS-PPO2

algorithm exhibits swifter convergence, which is pivotal for applications demanding rapid decision-making in dynamic and uncertain conditions.

REFERENCES

[1] Xiaoliang G, Liyuan Y. Overview of agent path planning algorithm, J. Science and Technology Information,2019,17(13):23-24. DOI: 10.16661/j.cnki.1672-3791.2019.13.023.

[2] Zhang Y, Liu K ,Gao F, et al.Research on Path Planning and Path Tracking Control of Autonomous Vehicles Based on Improved APF and SMC, J.Sensors,2023,23(18):

[3] Liu N, Ma C, Hu Z, et al.Workshop AGV path planning based on improved A* algorithm. J. Mathematical bio-sciences and engineering:MBE,2024,21(2):2137-2162.

[4] Kang M, Chen Q, Fan Z, et al.A RRT based path planning scheme for multi-DOF robots in unstructured environments, J. Computers and Electronics in Agriculture,2024,218108707-.

[5] Dao K T, Ngo G T, Pan S J, et al.Enhancing Path Planning Capabilities of Automated Guided Vehicles in Dynamic Environments:Multi-Objective PSO and Dynamic-Window Approach J.Biomimetics,2024,9(1):

[6] Huo F, Zhu S, Dong H, et al.A new approach to smooth path planning of Ackerman mobile robot based on improved ACO algorithm and B-spline curve, J. Robotics and Autonomous Systems, 2024, 175104655-.

[7] Yuan L, Qiong Q, Zhangfang H, et al.Path Planning for Unmanned Delivery Robots Based on EWB-GWO Algorithm, J. Sensors,2023, 23(4):1867-1867.

[8] Honglun W, Wentao L, Peng Y, et al.Three-dimensional path planning for unmanned aerial vehicle based on interfered fluid dynamical system, J.Chinese Journal of Aeronautics,2015, 28(01): 229-239.

[9] Yang X, Han Q. Improved DQN for Dynamic Obstacle Avoidance and Ship Path Planning, J.Algorithms,2023,16(5):

[10] Yuanhang C, Shaofang W. Framework of Active Obstacle Avoidance for Autonomous Vehicle Based on Hybrid Soft Actor-Critic Algorithm, J. Journal of Transportation Engineering, Part A: Systems, 2023,149(4):

[11] Shuhuan W, Zeteng W, Di Z, et al.A multi-robot path-planning algorithm for autonomous navigation using meta-reinforcement learning based on transfer learning, J.Applied Soft Computing Journal,2021,110

[12] Yong L. Research on Autonomous Obstacle Avoidance Strategy of UAV Based on Fluid Disturbance Algorithm, D.Donghua University, 2022.DOI:10.27012/d.cnki.gdhuu.2022.001517.

[13] Jianfa W, Honglun W, Yanxiang W, et al.UAV Reactive disturbed Fluid path Planning, J.Acta Automatica Sinica,2023,49(02):272-287. DOI:10.16383/j.aas.c210231.

[14] Siyi F, Ming C, Nian D. Obstacle avoidance route planning and simulation of agricultural plant protection UAV based on fluid disturbance algorithm, J.Agricultural Engineering Technology,2018, 38(09):51-54. DOI:10.16815/j.cnki.11-5436/s.2018.09.009.

[15] Yao P, Wang H, Liu C. 3-D dynamic path planning for UAV based on interfered fluid flow, C.Proceedings of 2014 IEEE Chinese Guidance, Navigation and Control Conference,Aug.2014,Yantai,China,997-1002.

[16] Peng Y, Honglun W, Zikang S. UAV feasible path planning based on disturbed fluid and trajectory propagation, J.Chinese Journal of Aeronautics,2015,28(04):1163-1177.

[17] schulman J, Wolski F, Dhariwal P, et al..Proximal policy optimization algorithms, J. 2017.DOI:10.48550/arXiv.1707.06347.

[18] Kangxiong C, Lei L. Uav Path planning algorithm based on disturbed fluid and TD3, J.Electric Light and Control,2024,31(01):57-62.

[19] Ke Y, Yiting Z, Zhi Z, et al.Design of sequence decision guidance algorithm for space controller based on PPO2, J.Software,2023, 44(07):5-12+65.

[20] Haarnoja T, Zhou A, Hartikainen K, et al.Soft Actor-Critic Algorithms and Applications, J. 2018.DOI:10.48550/arXiv.1812. 05905.