machines

MDPI

Article

Path Planning of Unmanned Aerial Vehicle in Complex Environments Based on State-Detection Twin Delayed Deep Deterministic Policy Gradient

Danyang Zhang, Zhaolong Xuan *, Yang Zhang, Jiangyi Yao, Xi Li and Xiongwei Li

Equipment Simulation Training Center, Shijiazhuang Campus, Army Engineering University, Shijiazhuang 050003, China
* Correspondence: youngzhxm@aeu.edu.cn

Abstract: This paper investigates the path planning problem of an unmanned aerial vehicle (UAV) for completing a raid mission through ultra-low altitude flight in complex environments. The UAV needs to avoid radar detection areas, low-altitude static obstacles, and low-altitude dynamic obstacles during the flight process. Due to the uncertainty of low-altitude dynamic obstacle movement, this can slow down the convergence of existing algorithm models and also reduce the mission success rate of UAVs. In order to solve this problem, this paper designs a state detection method to encode the environmental state of the UAV's direction of travel and compress the environmental state space. In considering the continuity of the state space and action space, the SD-TD3 algorithm is proposed in combination with the double-delayed deep deterministic policy gradient algorithm (TD3), which can accelerate the training convergence speed and improve the obstacle avoidance capability of the algorithm model. Further, to address the sparse reward problem of traditional reinforcement learning, a heuristic dynamic reward function is designed to give real-time rewards and guide the UAV to complete the task. The simulation results show that the training results of the SD-TD3 algorithm converge faster than the TD3 algorithm, and the actual results of the converged model are better.

Keywords: unmanned aerial vehicle; deep reinforcement learning; TD3; dynamic reward function; state detection

Citation: Zhang, D.; Xuan, Z.; Zhang, Y.; Yao, J.; Li, X.; Li, X. Path Planning of Unmanned Aerial Vehicle in Complex Environments Based on State-Detection Twin Delayed Deep Deterministic Policy Gradient. *Machines* **2023**, *11*, 108. https://doi.org/10.3390/machines11010108

Academic Editors: Wojciech Giernacki, Andrzej Łukaszewicz, Zbigniew Kulesza, Jaroslaw Pytka and Andriy Holovatyy

Received: 12 December 2022
Revised: 6 January 2023
Accepted: 10 January 2023
Published: 13 January 2023

1. Introduction

In recent years, UAVs have been widely used in the military by virtue of their stealth and high maneuverability. The small and medium-sized UAVs are widely used on the battlefield to attack important enemy targets because of their small size and the ability to evade radar detection by flying at low or ultra-low altitudes [1–4]. In addition, under the original technology, UAVs were controlled by rear operators for all operations and did not achieve unmanned operation in the true sense. Further, with the advancement of artificial intelligence technology, UAV intelligent pilot technology has also been rapidly developed, and UAV autonomous control can be realized in many functions. However, in order to further enhance the UAV's autonomous control capability, research on UAV path planning, real-time communication, and information processing needs to be strengthened. Among them, UAV autonomous path planning is a hot issue attracting current researchers' attention [5–8].

The path planning problem can be described as finding an optimal path from the current point to the target point under certain constraints, and many algorithms have been used so far to solve UAV path planning problems in complex unknown environments. Nowadays, the common path planning algorithms are the A*algorithm, the artificial potential field algorithm, the genetic algorithm, and the reinforcement learning method [9,10]. In recent years, deep learning (DL) and reinforcement learning (RL) have achieved a lot of results in many fields. Deep learning (DL) has strong data fitting ability, and reinforcement

learning (RL) can model the process reasonably and can be trained without labels [11–13] combined the advantages of DL and RL to obtain deep reinforcement learning (DRL), which provides a solution to the problem of perception and decision-making for UAVs in complex environments [14].

The DRL can effectively solve problems with both continuous and discrete spaces. Therefore, many researchers proposed using DRL to solve the path planning problem. Mnih, V., et al. proposed the deep Q-network (DQN) algorithm [15] by combining Q-learning and deep learning to solve the problem of dimensional explosion triggered by high-dimensional inputs. The DQN algorithm has achieved greater results in discrete action and state spaces but cannot effectively solve the continuous state and action spaces. In addition, when the changes of states and actions are infinitely partitioned, the amount of data for states and actions will show exponential growth with the increase of degrees of freedom, which can significantly impede the training and ultimately result in the algorithm failing [16]. Moreover, the discretized state and action space actually removes a large amount of important information, which will eventually lead to poor control accuracy, which will not meet the requirements for UAV control accuracy in air warfare. The actor-critic (AC) algorithm has the ability to handle the continuous action problem and is therefore widely used to solve problems in the continuous action space [17].The network structure of the AC algorithm includes an actor network and a critic network, actor network is responsible for outputting the action, and the critic network evaluates the value of the action and uses a loss function to continuously update the network parameters to get the optimal action strategy [18]. However, the effect of the AC algorithm relies heavily on the judgment of the value of the critic network, and the critic network converges slowly, which leads to the actor network. Lillicrap, T.P. et al. [19] proposed the deep deterministic policy gradient (DDPG) algorithm. The DDPG builds on the DQN algorithm's principles and combines the Actor-Critic algorithm's framework with several enhancements over the original AC algorithm to more effectively tackle the path planning problem in static environments. However, when applying the DDPG algorithm to solve path planning problems in dynamic environments, there is the problem of overvaluation of the value network, which leads to slow model convergence and a low training success rate. Scott Fujimoto [20] et al. improved on the deep deterministic policy gradient (DDPG) algorithm to obtain the twin delayed deep deterministic policy gradient (TD3) algorithm, which is the TD3 algorithm that incorporates the idea of the double DQN algorithm [21] into the DDPG algorithm, which effectively solves the problem of difficult algorithm convergence in a dynamic environment.

In this paper, a state detection method is proposed and combined with a dual-delay deep deterministic policy gradient (TD3) to form the SD-TD3 algorithm, which can solve the global path planning problem of a UAV in a dynamic battlefield environment and also identify and avoid obstacles. The main contributions of this paper are as follows:

(1) After combining the battlefield environment information, the information interaction mode between UAV and the battlefield environment is analyzed, a simulation environment close to the real battlefield environment is built, and the motion model of UAV autonomous path planning is constructed.

(2) The network structure and parameters most suitable for the SD-TD3 algorithm model are determined through multiple experiments. A heuristic dynamic reward function and a noise discount factor are also designed to improve the reward sparsity problem and effectively improve the learning efficiency of the algorithm.

(3) A state detection method is proposed that divides and compresses the environment state space in the direction of the UAV and encodes the space state by a binary number, so as to solve the problem of data explosion in the continuous state space of the reinforcement learning algorithm.

(4) The simulation experiments are carried out to verify the performance of the SD-TD3 algorithm, and the simulation experiments are based on the model of a UAV performing a low airspace raid mission. The results show that the SD-TD3 algorithm can help the UAV

avoid radar detection areas, mountains, and random dynamic obstacles in low-altitude environments so that it can safely and quickly complete the low-altitude raid mission.

(5) By analyzing the experimental results, it can be concluded that the SD-TD3 algorithm has a faster training convergence speed and a better ability to avoid dynamic obstacles than the TD3 algorithm, and it is verified that the SD-TD3 algorithm can further refine the environmental state information to improve the reliability of the algorithm model, so that the trained algorithm model has a higher success rate in practical applications.

The rest of the paper is structured as follows, with related work presented in Part II. The third part models and analyzes the battlefield environment. Part IV describes the state detection scheme, the specific structure of the SD-TD3 algorithm, and the setting of the heuristic reward function, and Part V verifies the performance of the SD-TD3 algorithm through a simulation environment and analyzes the experimental results. The conclusion is given in the sixth part.

2. Related Work

In recent years, a lot of research on autonomous UAV path planning has been carried out at home and abroad, which can be divided into four categories of algorithms according to their nature: graph search algorithms, linear programming algorithms, intelligent optimization algorithms, and reinforcement learning algorithms.

The graph search algorithm mainly contains the Dijkstra algorithm, the RRT algorithm, the A* algorithm, the D* algorithm, etc. The most classical Dijkstra algorithm shows higher search efficiency than depth-first search or breadth-first search in the problem of finding the shortest path. However, the execution efficiency of the Dijkstra algorithm gradually decreases as the map increases. Ferguson, D. et al. [22] optimized Dijkstra and proposed the A* and D* algorithms. Zhan et al. [23] proposed an UAV path planning based on the improved A* algorithm for the path planning problem of low altitude UAVs in a 3D battlefield environment that satisfies UAV performance constraints such as safe lift and turn radii. Saranya, C. et al. [24] proposed an improved D* algorithm for the path planning problem in complex environments, which introduced slope into the reward function. The simulations and experiments proved the effectiveness of the method, which can be used to guarantee the flight safety of UAVs in complex environments. Li, Z. et al. [25] applied the RRT algorithm to the unmanned ship path planning problem, and an improved fast extended random tree algorithm (Bi-RRT) is proposed. The simulation results show that the optimized RRT algorithm can shorten the planning time and reduce the number of iterations, which has better feasibility and effectiveness.

The linear programming algorithm is a mathematical theory and method to study the extremum of a linear objective function under linear constraints that is widely used in the military, engineering technology, and computer fields. Yan, J. et al. [26] proposed a mixed-integer linear programming-based UAV conflict resolution algorithm that establishes a safety separation constraint for pairs of conflicting UAVs by mapping the nonlinear safety separation constraint to sinusoidal value-space separation linear constraints, then constructs a mixed-integer linear programming (MILP) model, mainly to minimize the global cost, and finally conducts simulation experiments to verify the effectiveness of the algorithm. Yang, J. et al. [27] proposed a cooperative mission assignment model based on mixed integer linear programming for multiple UAV formations against enemy air defense fire suppression. The algorithm represents the relationship between UAVs and corresponding missions by decision variables, introduces continuous time decision variables to represent the execution time of missions, and establishes the synergistic relationship among UAVs and between UAVs performing missions by mathematical descriptions of linear equations and inequalities between decision variables. The simulation experiments show the rationality of the algorithm.

The intelligent optimization algorithms are developed by simulating or revealing certain phenomena and processes in nature or the intelligent behaviors of biological groups, and they generally have the advantages of simplicity, generality, and ease of parallel

processing. In UAV path planning, genetic algorithms, particle swarm algorithms, ant colony algorithms, and hybrid algorithms have been applied more often. Hao, Z. et al. [28] proposed an UAV path planning method based on an improved genetic algorithm and an A* algorithm for system positioning accuracy in the UAV path planning process, considering the UAV obstacle constraints and performance constraints, and taking the shortest planned trajectory length as the objective function, which achieved the goal of accurate positioning with the goal of the least number of corrected trajectories. Lin, C.E. [29] established an UAV system distance matrix to solve the multi-target UAV path planning problem and ensure the safety and feasibility of path planning, used genetic algorithms for path planning, and used dynamic planning algorithms to adjust the flight sequence of multiple UAVs. Milad Nazarahari et al. [30] proposed an innovative artificial potential field (APF) algorithm to find all feasible paths between a starting point and a destination location in a discrete grid environment. In addition, an enhanced genetic algorithm (EGA) is developed to improve the initial path in continuous space. The proposed algorithm not only determines the collision-free path but also provides near-optimal solutions for all robot path planning problems.

Reinforcement learning is an important branch of machine learning that can optimize decisions without a priori knowledge and by continuously trying to iterate to obtain feedback information based on the environment. Currently, many researchers combine reinforcement learning and deep learning to form deep reinforcement learning (DRL), which can effectively solve path planning problems in dynamic environments. Typical DRL algorithms include the deep Q-network (DQN) algorithm, the actor-critic (AC) algorithm, the deep deterministic policy gradient (DDPG) algorithm, and the twin delayed deep deterministic policy gradient (TD3) algorithm. Cheng, Y. et al. [31] proposed a deep reinforcement learning obstacle avoidance algorithm under unknown environmental disturbances that uses a deep Q-network architecture and sets up a comprehensive reward function for obstacle avoidance, target approximation, velocity correction, and attitude correction in dynamic environments, overcoming the usability problems associated with the complexity of control laws in traditional parsing methods. Zhang Bin [32] et al. applied the DDPG algorithm. The improved algorithm has significantly improved efficiency compared with the DDPG algorithm. Hong, D. [33] et al. proposed an improved double-delay deep deterministic policy gradient (TD3) algorithm to control multiple UAV actions and also utilized the frame superposition technique for continuous action space to improve the efficiency of model training. Finally, simulation experiments showed the reasonableness of the algorithm. Li, B. [34] et al. combined meta-learning with the dual-delay depth-deterministic policy gradient (TD3) algorithm to solve the problem of rapid path planning and tracking of UAVs in an environment with uncertain target motion, which improved the convergence value and speed. Christos Papachristos et al. [35] proposed an off-line path planning algorithm for the optimal detection problem of an a priori known environment model. The actions that the robot should take if no previous map is available are iteratively derived to optimally explore its environment.

In summary, many approaches for autonomous path planning have been proposed in the field of UAVs, but relatively little work has been done to apply these approaches to battlefield environments. In the previous experiment, we selected the DQN algorithm, the DDPG algorithm, and the TD3 algorithm. The experimental results show that the DQN algorithm and the DDPG algorithm are difficult to converge, and the training results are not ideal. In this paper, the double-delay deep deterministic policy gradient (TD3) algorithm is selected for UAV path planning because TD3 not only has powerful deep neural network function fitting capability and better generalized learning capability but also can effectively solve the problem of overestimation of Q-value during the training process for algorithmic models with actor-critic structure. The TD3 also has the advantage of fast convergence speed and is suitable for acting in continuous space. However, the original TD3 algorithm usually only takes the current position information of the UAV as the basis for the next behavior judgment, and the training effect is not ideal in a dynamic environment. In

this paper, we provide a state detection method to detect the environmental space in the direction of UAV flight so that the algorithm model can have stronger environmental awareness and make better decisions during the flight process.

3. Environmental Model

3.1. Description of the Environmental Model

Figure 1 illustrates the battlefield environment of an UAV on a low-level raid mission, where the UAV is assigned to attack a radar position 50 km away. In order to avoid flying too high and being detected by radar, the UAV must take a low-altitude flight below 1 km. During low-altitude flight, the UAV needs to autonomously avoid static ground obstacles such as mountains and buildings. At the same time, because the low-altitude environment is susceptible to dynamic obstacles such as flying birds and civilian low-altitude vehicles, the UAV also needs to make accurate and timely responses to random dynamic obstacles.

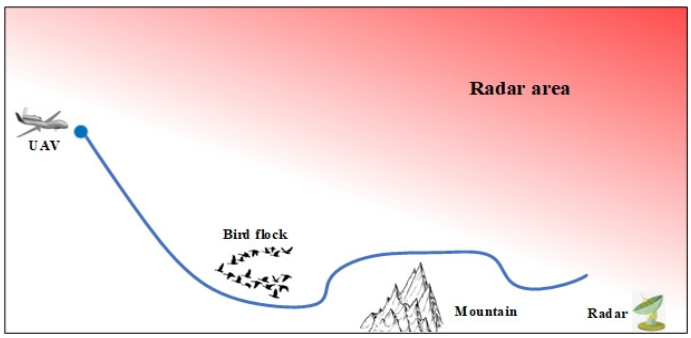

Figure 1. Schematic of battlefield environment.

3.2. Environment Parameters Setting

The simulation experimental environment is set as a low-altitude area of 50 km long and 1 km high, and the radar position $Radar_{(x,y)}$ and the UAV initial position $UAV_{(x_0,y_0)}$ can be expressed as:

$$Radar_{(x,y)} = [50\text{ km}, 0.2\text{ km}] \tag{1}$$

$$UAV_{(x_0,y_0)} = [0\text{ km}, 1\text{ km}] \tag{2}$$

The velocity v of the UAV can be divided into horizontal velocity $v_{xi} \in (0\text{ m/s},\ 100\text{ m/s})$ and vertical velocity $v_{yi} \in (-3\text{ m/s}, 3\text{ m/s})$, and the UAV real-time position $uav(t_i)$ can be expressed as

$$uav(t_i) = [x_{t_i}, y_{t_i}] = [\sum_i (v_{xi} * \Delta t), \sum_i (v_{yi} * \Delta t)] \tag{3}$$

The UAV should avoid collision with static ground obstacles such as mountains and buildings during low-altitude flight. In addition, the assumption is that the height of the ground obstacle is 100 m and the coordinates of its lowest center point $Static\_obstacle_{(x,y)}$ can be expressed as:

$$Static\_obstacle_{(x,y)} = [20\text{ km}, 0\text{ km}] \tag{4}$$

In the process of low-altitude flight, the UAV should also avoid dynamic obstacles such as flying birds, low-altitude civil vehicles, etc. Assuming that dynamic obstacles are randomly generated in the area below 300 m in height and the initial position is expressed as (x_0, y_0), its dynamic real-time position $Dynamic\_obstacle_{(x,y)}$ can be expressed as follows:

$$Dynamic\_obstacle_{(x,y)} = [x_d, y_d] = \left[x_0 + v_{dx} * \Delta t, y_0 + v_{dy} * \Delta t\right] \tag{5}$$

The initial position (x_0, y_0) in Equation (5), $x_0 \in (0\text{ km}, 50\text{ km})$, $y_0 \in (0\text{ km}, 0.3\text{ km})$, dynamic obstacle moving speed $v_{dx} \in (-10\text{ m/s}, 10\text{ m/s})$, $v_{dy} \in (-1\text{ m/s}, 1\text{ m/s})$.

A safe distance of more than 50 m should be maintained between the UAV and the dynamic obstacle.

$$d_{safe} = \left| uav(t_i) - Dynamic_{obstacle(x,y)} \right| = \sqrt{(x_{t_i} - x_d)^2 + (y_{t_i} - y_d)^2} \geq 0.05 \text{ km} \quad (6)$$

The maximum firing range of the air-to-ground missile on the UAV is 10 km. Assuming that the missile has a 100% hit rate within the firing range, the UAV is considered to have completed its mission when it safely reaches a position 10 km from the radar. The UAV is equipped with a radar warning device to determine if it is locked by the radar, and the maximum radius of the guided radar is 40 km. However, the probability P of an UAV being detected by radar in the airspace below 1 km in altitude is related to the distance d between radars and the current flight altitude h due to factors such as the curvature of the earth, detection angle, ground obstructions, and ground clutter. If the UAV is flying at too low an altitude, there is a low altitude blind zone that is completely undetectable by radar. Assuming that the radar blind zone is an airspace with an altitude less than 300 m, the radar detection probability model can be expressed as:

$$P = \begin{cases} 0 & d > 50 \text{ km} \\ 0 & h < 0.3 \text{ km} \\ 1 & d \leq 50 \text{ km},\ h \geq 1 \text{ km} \\ \frac{1}{0.01 + 2e^{-5h+5}} - \frac{d^2}{3200} + \frac{1}{2} & d \leq 50 \text{ km},\ 0.3 \text{ km} \leq h \leq 1 \text{ km} \end{cases} \quad (7)$$

Equation (7) can be obtained from the radar detection probability model as shown in Figure 2.

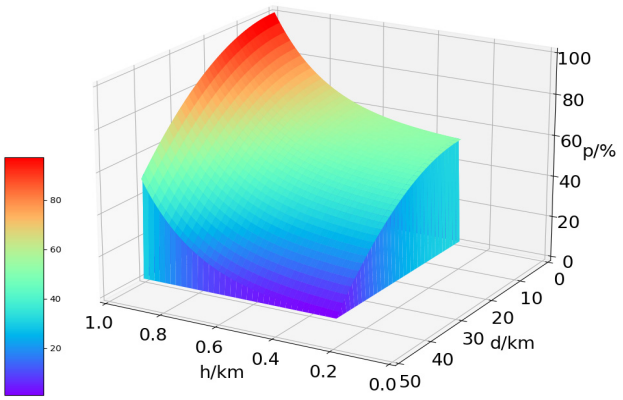

Figure 2. Probability model of radar detection.

From Figure 2, we can see that the d-axis is the distance between the UAV and radar, the h-axis is the flight height of the UAV, and the p-axis is the probability of the UAV being detected by radar. In addition, through the analysis, it can be seen that the UAV can effectively avoid the detection of radar when it flies below 300 m but the probability of being detected by radar is not 100% in the range of flight altitude $h \in (0.3 \text{ km},\ 1 \text{ km})$, which makes it difficult for the UAV to accurately identify the radar-covered airspace during the training process. In summary, this path planning experiment is challenging.

4. TD3-Based UAV Path Planning Model

In this section, we will describe the origin and development of the TD3 algorithm and improve it to better solve this path planning problem. The improvements are twofold. First, a dynamic reward function is set up to solve the problem of sparse rewards in the traditional deep reinforcement learning algorithm model, which can provide real-time feedback to the corresponding rewards according to the state of the UAV and speed up the convergence of the algorithm model in the training process. Secondly, the SD-TD3 algorithm is proposed, which mainly sets the segmentation of the region in the flight direction of the UAV, detects

and encodes the states at different regional locations with binary numbers, and adds the detected environmental state values to the input of the algorithm model to improve the UAV's obstacle avoidance capability.

4.1. Deep Reinforcement Learning Model

Deep reinforcement learning (DRL) is a learning method combining DL and RL that combines the data processing capability of DL with the decision-control capability of RL. In recent years, DRL has achieved great results in continuous space motion control and can effectively solve the UAV path planning problem. The deep deterministic policy gradient (DDPG) algorithm is a representative algorithm in DRL for solving continuous motion space problems, which can lead to deterministic actions based on state decisions. The idea of the DDPG algorithm is derived from the Deep Q Network (DQN) algorithm, and the update function of DQN can be expressed as:

$$Q(s,a) = Q(s,a) + \alpha\left(r + \gamma \underset{a'}{\max} Q(s',a') - Q(s,a)\right) \tag{8}$$

where $\alpha \in (0,1]$ is the learning rate, which is used to control the proportion of future rewards during learning. In addition, $\gamma \in (0,1)$ is the decay factor, which indicates the decay of future rewards. r denotes the reward after performing action a. From Equation (8), it can be seen that DQN is updated using the action currently considered to be of the highest value at each learning, which results in an overestimation of the Q-value, and thus DDPG also suffers from this problem. In addition to this, DDPG is also very sensitive to the adjustment of hyperparameters [36].

The double-delay deep deterministic policy gradient (TD3) algorithm solves these problems. The TD3 makes three improvements over the DDPG: first, it uses two independent critic networks to estimate Q values and selects smaller Q values for calculation when calculating the target Q values, which can effectively alleviate the problem of overestimation of Q values; second, the actor network uses delayed updates. The critic network is updated more frequently compared with the actor network, which can minimize the error; third, smoothing noise is introduced in the action value output from the actor target network to make the valuation more accurate, but no noise is introduced in the action value output from the actor network.

The pseudo-algorithm of TD3 can be expressed as Algorithm 1:

Algorithm 1: The Pseudo-Algorithm of TD3

1. Initialize critic networks Q_{θ_1}, Q_{θ_2}, and actor network with random parameters θ_1,θ_2,$\emptyset$
2. Initialize target networks $\theta_1' \leftarrow \theta_1$, $\theta_2' \leftarrow \theta_2$, $\emptyset' \leftarrow \emptyset$
3. Initialize replay buffer B
4. **for** t = 1 **to** T **do**
5. Select action with exploration noise $a \sim \pi_{\emptyset}(s) + \epsilon, \epsilon \sim N(0,\sigma)$ and observe reward r and new state s'
6. Store transition tuple (s,a,r,s') in B
7. Sample mini-batch of N transitions (s,a,r,s') from B
8. Compute target action $\widetilde{a} \leftarrow \pi_{\emptyset'}(s') + \epsilon, \epsilon \sim \mathrm{clip}(N(0,\widetilde{\sigma}), -\mathrm{c}, \mathrm{c})$
9. Compute target Q value $y \leftarrow \mathrm{r} + \gamma \min_{i=1,2} Q_{\theta_i'}(s',\widetilde{a})$
10. Update critics $\theta_i \leftarrow argmin_{\theta_i} N^{-1} \sum \left(y - Q_{\theta_i}(s,a)\right)^2$
11. **if** t mod d **Then**
12. Update $\emptyset$ by the deterministic policy gradient:
13. $\nabla_{\emptyset} J(\emptyset) = N^{-1} \sum \nabla_a Q_{\theta_1}(s,a)\Big|_{a=\pi_{\emptyset}(s)} \nabla_{\emptyset} \pi_{\emptyset}(s)$
14. Update target networks:
15. $\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$
16. $\emptyset' \leftarrow \tau\,\emptyset + (1-\tau)\emptyset'$
17. **end if**
18. **end for**

4.2. State Detection Method

The conventional TD3 algorithm takes the current position information (x,y) of the UAV as input, outputs the action to the environment, and continuously learns from the environment with rewards for interaction. When trained enough times, the UAV is able to take the corresponding correct action at any position to reach the destination. In this method, it is difficult to effectively achieve the purpose of avoiding obstacles through the current position information of the UAV when dynamic obstacles appear in the environment. In order to be able to effectively complete the path planning task in a complex dynamic environment, the UAV must be able to identify the environmental state of the forward region, and when an obstacle appears in the forward region, the UAV can immediately identify the location of the obstacle and make a decision to avoid the obstacle. Therefore, in this paper, a state detection coding method is designed to encode the environmental state of the UAV's forward area.

The UAV is usually equipped with various sensors for detecting the surrounding environment. Suppose the UAV is equipped with sensors that can detect the state of the area ahead, and the working maximum distance of the sensors is 100 m. In addition, through these sensors, the UAV can detect whether there is an obstacle in the area ahead. We use the binary number 1 or 0 to indicate the presence or absence of obstacles. Further, through the state detection code, we can get a set of current environment state information arrays; this array will be added to the input of the algorithm model. In this way, the UAV can make the correct decision based on the environmental state information of the forward area.

By the state detection coding method, the environment space of the UAV advance region needs to be divided. Since the current environment is a continuous space, it can theoretically be divided an infinite number of times, but it will cause an increase in the training computation. Therefore, our scheme takes a limited number of divisions, which can also be regarded as the compression of the environmental state space in front of the UAV. Taking Figure 3 as an example, it can be seen that the region in front of the UAV is divided into six parts on average, and there are seven location points for encoding. The state input information of the UAV, S_{UAV}, can be expressed as follows:

$$S_{UAV} = [s_0, s_1, s_2, s_3, s_4, s_5, s_6, x, y], s_i \in [0, 1] \tag{9}$$

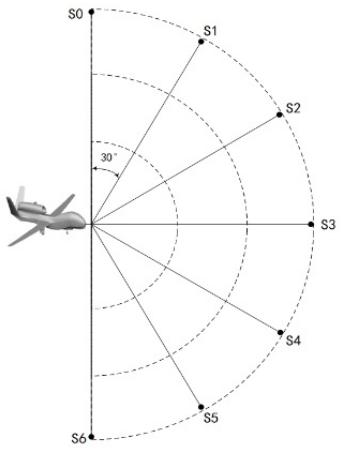

Figure 3. Schematic diagram of status detection code.

In Equation (9) s_i represents the environmental state information about each direction; $s_i = 0$ means there is no obstacle in the corresponding direction, and $s_i = 1$ means there is an obstacle in the corresponding direction. In addition, by inputting S_{UAV} to the algorithm model, after training the UAV can make correct decisions based on environmental state information in the forward direction, and thus accomplish the task of avoiding various

obstacles. In order to verify the effectiveness of the state detection coding method, this experiment will also further refine the state and divide the area in front of the UAV into 12 parts on average; the location points for coding will then be 13, and the state input information S_{UAV} can be expressed as:

$$S_{UAV} = [s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10}, s_{11}, s_{12}, x, y], s_i \in [0, 1] \tag{10}$$

4.3. Heuristic Dynamic Reward Function

The reward functions, also known as immediate rewards, are an important component of deep reinforcement learning algorithm models. When an UAV performs an action, the environment generates feedback information for that action and evaluates the effect of that action. In traditional reinforcement learning algorithms, intelligence is rewarded when it completes a task and is not rewarded in other states. However, such rewards are prone to the reward sparsity problem in the face of complex environments [37]. The effective rewards are not available in a timely manner, and the algorithm model will be difficult to converge, which can be solved by setting up a heuristic reward function with guidance. The heuristic reward function designed in this paper can be expressed as follows:

$$R = \begin{cases} \frac{\beta(D-d_{t+1})}{D}, & d_t > d_{t+1} \\ -\frac{\beta(D-d_{t+1})}{D}, & d_{t+1} \geq d_t \end{cases} \tag{11}$$

In Equation (11) $\beta \in (0, +\infty)$ is the reward coefficient, D is the initial distance of 50 km between the UAV and radar position, d_t is the distance between the UAV and radar position at the current moment, and d_{t+1} is the distance between the UAV and radar position at the next moment. In analyzing Equation (11), we can see that when the UAV performs each action, if it is closer to the target at the next moment, it gets a positive reward, and the closer it is to the radar position, the greater the positive reward value it gets; if it is further away from the radar position at the next moment, it gets a negative reward, and the further it is from the radar position, the greater the negative reward value it gets.

In addition to the heuristic reward, when the distance between UAV and a dynamic obstacle is less than the safe distance d_{safe} or when there is a collision with a static obstacle, the reward of 300 dollars will be obtained, the environment will be reset, and UAV will start training from the initial position again. In addition, when the distance between the UAV and the radar is less than 10 km, the mission is completed, the reward of 300 dollars will be given, the environment is reset, and the UAV starts training again from its initial position.

In general, the reward function in this study is a dynamic reward function generated by combining the current state of the UAV. The dynamic nature of the reward function has two main points. One is that the reward generated by the environment interaction is generated in real time when the UAV is trained in the environment, which solves the problem of sparse reward compared to traditional reinforcement learning. Second, the reward value obtained during the training process of UAV and environment interaction will change with the current location information. According to the change in reward value, the UAV can be guided to move in the appropriate direction, which can promote the convergence of the algorithm model. The reward function has the role of heuristic guidance for UAVs, so it can be called the heuristic reward function.

4.4. State Detection Double-Delay Depth Deterministic Policy Gradient Algorithm Model

In combining the TD3 algorithm model with the above-mentioned heuristic reward function and the state detection method, it is possible to design the state detection double-delay depth deterministic policy gradient (SD-TD3) algorithm model. The UAV detects the state information s of the environment in the forward region, encodes it as the input of the algorithm model, and outputs the action after calculating it with the TD3 model. Figure 4 shows the specific algorithm. During the process, the UAV executes a and the environment's state changes to the next state, s'; while executing a, the environment will

feedback a reward r, according to the reward function, and the quadratic information set (s, a, r, s') is obtained. The quaternion information reaches the experience pool, and when the experience pool is stored to a certain number, random samples are drawn for training, and by inputting these sample data, they are used to update the network of actor-critic. The existence of the experience pool helps UAV learn from previous experiences and improve the efficiency of sample utilization. The random sampling can break the correlation between samples and make the learning process of the UAV more stable [38].

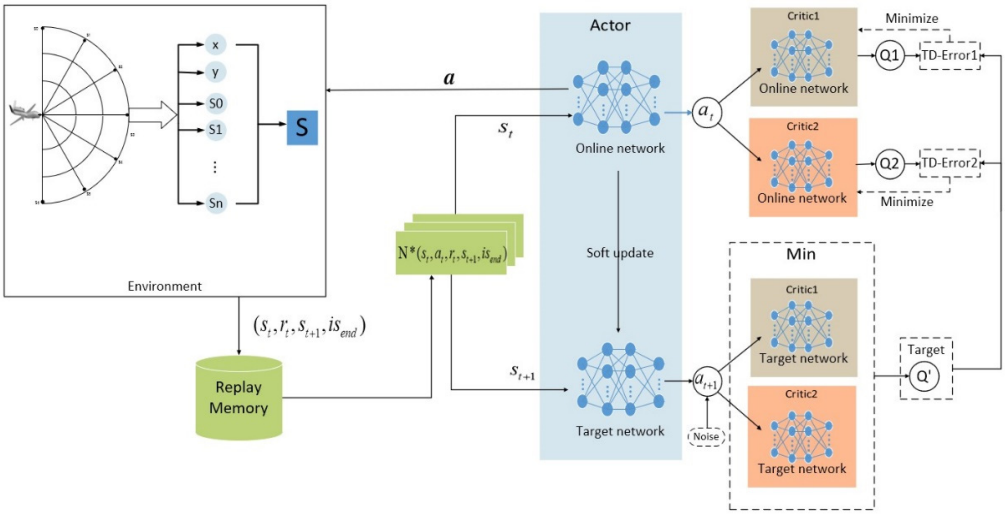

Figure 4. The combination of the state probing method and the TD3 model.

The TD3 algorithm model sets up a total of six neural networks based on the actor-critic structure, which are actor network $\pi_{\emptyset}$, actor target network $\pi_{\emptyset'}$, critic network Q_{θ_1}, critic network Q_{θ_2}, critic target network $Q_{\theta_1'}$, critic target network $Q_{\theta_2'}$. The roles and updates of these networks can be expressed as:

1. Actor network $\pi_{\emptyset}$: input the current state s of the UAV, output the current action a and then interact with the environment to reach the next state s' and the obtained reward r. The actor network parameters $\emptyset$ are updated iteratively in this process.
2. Actor target network $\pi_{\emptyset'}$: s' in the quaternion is used as input after random sampling from the experience pool, and the next action a' is generated after adding noise ϵ to the output result. The actor target network parameter $\emptyset'$ is based on the actor network parameter $\emptyset$ for delayed soft update, $\theta_i' = \tau\theta_i + (1-\tau)\theta_i'$.
3. Critic network Q_{θ_i}: input current state s and current action a, output current Q value $Q_{\theta_i}(s, a)$, and iteratively update critic network parameter θ_i in this process. in calculating the target Q value, take the smallest of $Q_{\theta_1'}(s', a')$ and $Q_{\theta_2'}(s', a')$ is calculated, $y = \mathrm{r} + \gamma \min_{i=1,2} Q_{\theta_i'}(s', a')$.
4. Critic target network $Q_{\theta_i'}$: After random sampling from the experience pool, s' in the quaternion and the next action a' generated by the Actor target network are used as input, and $Q_{\theta_1'}(s', a')$ and $Q_{\theta_2'}(s', a')$ are output. The critic target network parameter θ_i' is delayed soft update based on critic network parameter θ_i, $\emptyset' = \tau\emptyset + (1-\tau)\emptyset'$.

For the Critic network, the loss function is expressed as:

$$J(\theta_i) = N^{-1} \sum \left(y - Q_{\theta_i}(s, a)\right)^2 \tag{12}$$

For actor networks, a deterministic strategy is used to optimize the parameters, and the loss function is expressed as

$$\nabla_{\emptyset} J(\emptyset) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)\Big|_{a=\pi_{\emptyset}(s)} \nabla_{\emptyset} \pi_{\emptyset}(s) \tag{13}$$

5. Simulation Experiment Setup and Result Analysis

In order to test the effectiveness of SD-TD3 algorithm performance, this experiment set up two simulation experimental environments for training Environment 1 is a static environment, set up with a radar detection area and a mountain range as static obstacles. Environment 2 is a dynamic environment, in which a random low-altitude dynamic obstacle is added to Environment 1, and the relevant environmental parameters are described in Section 3.2. The TD3 algorithm and SD-TD3 algorithm are verified in these two environments. All training experiments were conducted on a computer with an Intel(R) Core(TM) i7-10700 CPU and an NVIDIA GeForce RTX3060Ti GPU, using Python 3.9 as the project interpreter. The deep learning framework Pytorch-1.12.1 is used for neural network training under Windows.

5.1. Algorithm Hyperparameter Settings

The hyperparameters of TD3 and SD-TD3 algorithms are: neural network structure parameters, learning rate α, discount factor γ, experience pool size R, number of samples B, target network soft update factor τ, noise attenuation factor k. These parameters have different effects on the performance of the algorithms. If the number of hidden layers and hidden layer neurons in the neural network is too small, the neural network cannot fit the data well, and if the number of hidden layers and hidden layer neurons in the neural network is too large, the increase in the calculation amount of the algorithm cannot effectively learn. The larger the value of learning rate α, the faster the training speed of the algorithm, but prone to oscillation; the smaller the value, indicating that the slower the training speed model is difficult to converge. The larger the discount factor γ, the more the algorithmic model focuses on past experience; the smaller the value, the more it focuses on current experience. In addition, both the size of the experience pool R and the number of samples sampled B affect the learning efficiency of the algorithm; if the value is too small, the learning efficiency will be low, and if the value is too large, the algorithm tends to converge to a local optimum. The smaller the soft update coefficient τ of the target network, the more stable the algorithm is, and the smaller the change of the target network parameters, the slower the algorithm will converge. Since the action values of the actor target network output in the algorithm model is added with smoothing noise ϵ

$$\epsilon \sim \mathrm{clip}(N(0, \widetilde{\sigma}), -\mathrm{c}, \mathrm{c}) \tag{14}$$

In Equation (14), $\widetilde{\sigma}$ is the standard deviation of the normal distribution, and the larger the value, the larger the value of the added noise. However, as the model gradually converges with the training process, it is not easy to converge if the noise value is too large to produce oscillations. Therefore, the noise attenuation factor $k \in (0, 1)$ is set in this experiment, and the standard deviation $\widetilde{\sigma}$ is multiplied by the noise attenuation factor k to reduce the noise whenever the UAV completes a task during training. The specific hyperparameter settings are shown in Table 1.

Due to the fact that both TD3 algorithm and SD-TD3 algorithm use the actor-critic framework, the learning rates α_a and α_c of the Actor module network and Critic module network are important hyperparameters. Where α_a corresponds to actor network $\pi_\emptyset$ and actor target network $\pi_{\emptyset'}$, α_c corresponds to critic network Q_{θ_1}, critic network Q_{θ_2}, critic target network $Q_{\theta_1'}$ and critic target network $Q_{\theta_2'}$. so, α_c has a large impact on the convergence effect of the model. In order to select a suitable α_c, we set five different α_c for training in environment 1 and environment 2 through several experiments, and compare the convergence of the TD3 algorithm and SD-TD3 algorithm in different environments by analyzing the results.

Table 1. Hyperparameter settings.

Hyperparameters	Symbol	Value
Hidden layers	-	2
Hidden layer units	-	256
Max episodes	-	3000
Max steps per episode	-	400
Actor network learning rate	α_a	0.0001
Critic network learning rate	α_c	0.001
Discount factor	γ	0.99
Replay buffer size	R	6400
Batch size	B	128
Soft update rate	τ	0.005
Noise attenuation factor	k	0.999

5.2. Analysis of Experimental Results

5.2.1. TD3 Algorithm Model

Figures 5 and 6 show the training effects achieved by the TD3 algorithm after setting five different critic module network learning rates α_c in environments 1 and 2, respectively. From Figure 5, it can be seen that the most suitable α_c for the TD3 algorithm model in Environment 1 is 0.0009, and the model starts to converge at a round number of 660 No oscillation occurs after the model converges. From Figure 6, it can be seen that in environment 2, the most suitable α_c for the TD3 algorithm model is 0.0001, and the model starts to converge when the number of rounds is 710. Due to the dynamic obstacles of random appearance and random motion in environment 2, the model can converge, but there are still more obvious oscillations.

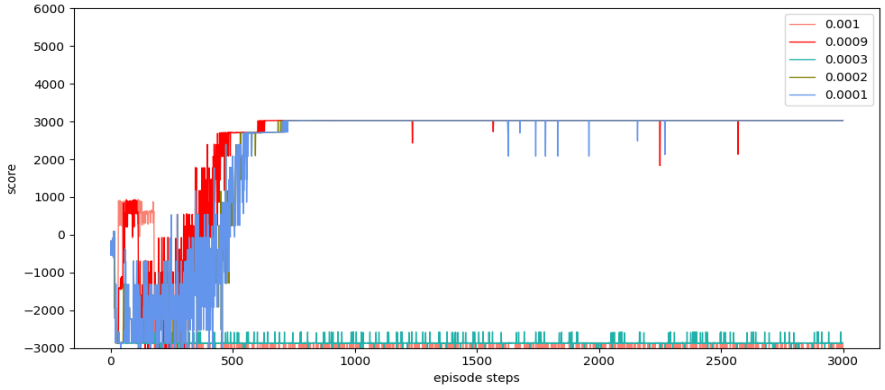

Figure 5. Training result of TD3 algorithm model in environment 1.

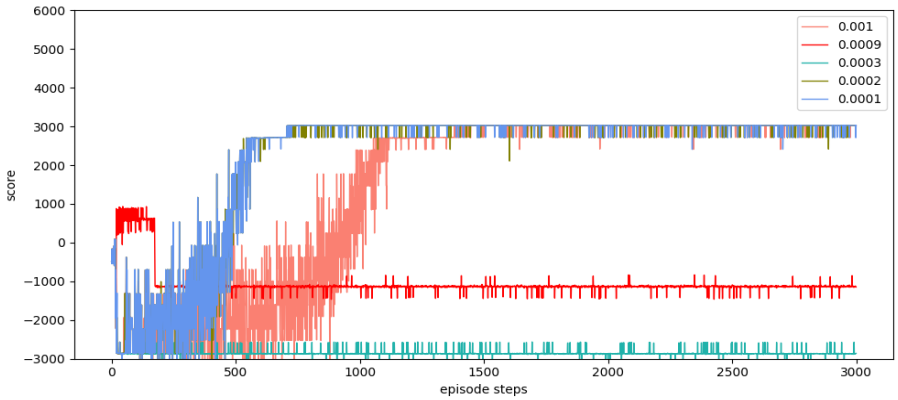

Figure 6. Training result of TD3 algorithm model in environment 2.

5.2.2. SD-TD3 Algorithm Model

To verify the performance of the SD-TD3 algorithm model, the area in front of the UAV is divided equally into six parts. Figures 7 and 8 show the training results achieved by the SD-TD3 (6) algorithm with five different critic module network learning rates α_c set in environment 1 and environment 2, respectively. From Figure 7, it can be seen that the most appropriate α_c for the SD-TD3 (6) algorithm model is 0.0003 in Environment 1, and the model starts to converge at a round number of 410. From Figure 8, it can be seen that under environment 2, the most suitable α_c for the SD-TD3 (6) algorithm model is 0.001, and the model starts to converge at a number of rounds of 635. At this point, it can be seen that the SD-TD3 (6) algorithm model converges faster compared to the TD3 algorithm model.

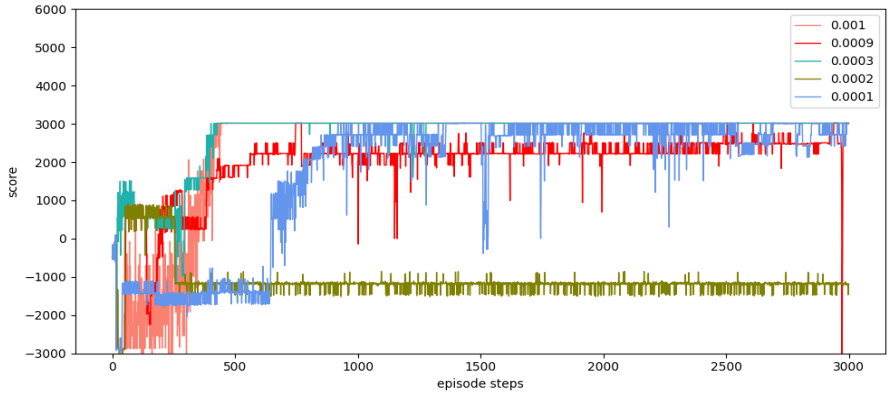

Figure 7. Training result of SD-TD3 (6) algorithm model in environment 1.

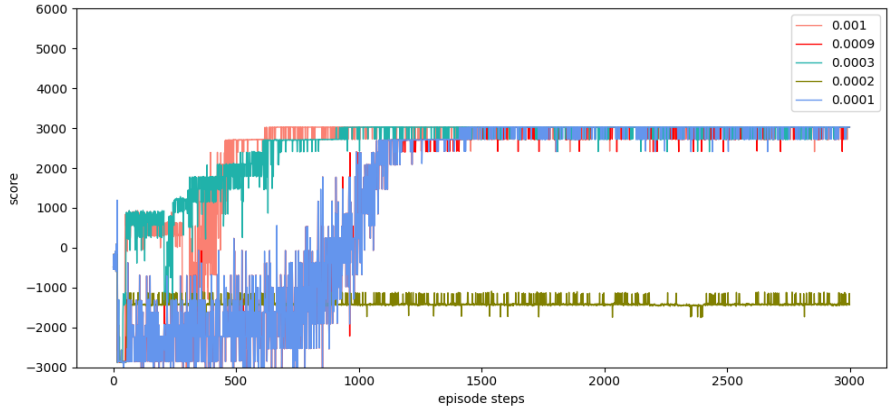

Figure 8. Training results of SD-TD3 (6) algorithm model in Environment 2.

In order to further validate the performance of the SD-TD3 algorithm model, the area in front of the UAV's travel was divided equally into 12 sections. Figures 9 and 10 show the training results achieved by the SD-TD3 (12) algorithm in Environment 1 and Environment 2 after setting five different Critic module network learning rates α_c, respectively. From Figure 9, it can be seen that the most appropriate α_c for the SD-TD3 (12) algorithm model is 0.0009 in Environment 1, and the model starts to converge at a round number of 90. From Figure 10, it can be seen that under environment 2, the most suitable α_c for the SD-TD3 (12) algorithm model is 0.0009, and the model starts to converge at the number of rounds of 210. At this point, it can be seen that the SD-TD3 (12) algorithm model converges faster compared to the SD-TD3 (12) algorithm model.

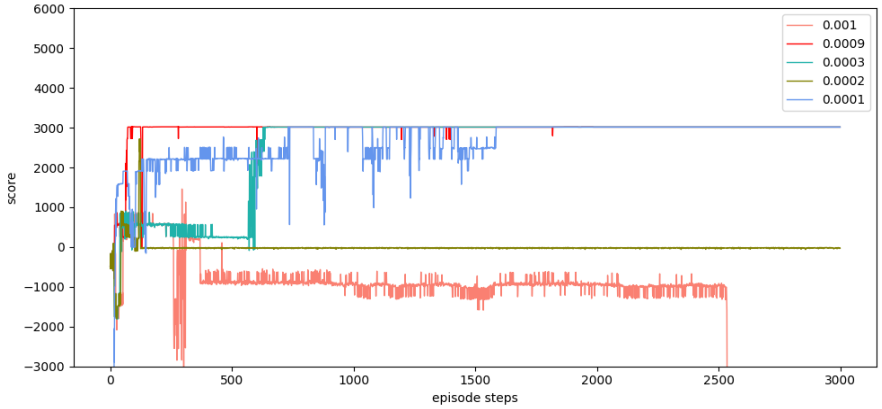

Figure 9. Training result of SD-TD3 (12) algorithm model in environment 1.

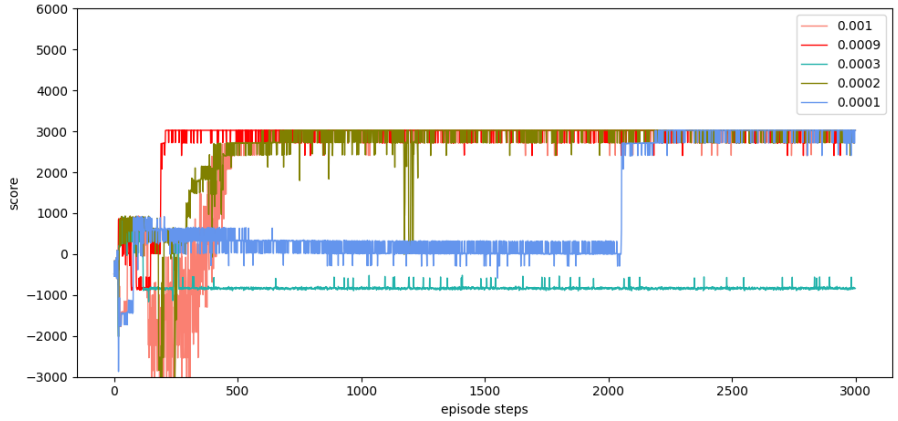

Figure 10. Training result of SD-TD3 (12) algorithm model in environment 2.

5.2.3. Comprehensive Analysis

Figure 11 compares the best training results of the three algorithmic models in the two environments.

In analyzing all the training results, it is clear from them that the TD3 algorithm model and the SD-TD3 algorithm model eventually obtain the same reward value in both static and dynamic environments, and both can reach convergence, but the convergence speed in the dynamic environment is slower than that in the static environment. In comparison with the convergence speed of the three models, the SD-TD3(12) algorithm model converges faster than the SD-TD3(6) algorithm, and the SD-TD3(6) algorithm model converges faster than the TD3 algorithm. Therefore, the convergence speed of the SD-TD3 algorithm model is higher than that of the TD3 algorithm model in both static and dynamic environments.

Further observation of the training results shows that the oscillation amplitude of the three models in the dynamic environment is larger than that in the static environment. In order to verify the robustness of the model, the model that had been trained and reached convergence was run for 30,000 rounds in two environments, and the analysis was carried out by comparing the probability of successfully completing the task.

In Figure 12, the blue color indicates the probability in environment 1, and the orange color indicates the probability in environment 2. The actual success rates of the three algorithm models in the static environment are similar and close to 1. The actual success rates of the three models in the dynamic environment are lower compared to those in the static environment, but the SD-TD3 algorithm's performance is higher compared to the TD3 algorithm, and the performance of the SD-TD3 algorithm model can be improved with further refinement of the spatial states, thus improving the actual success rates.

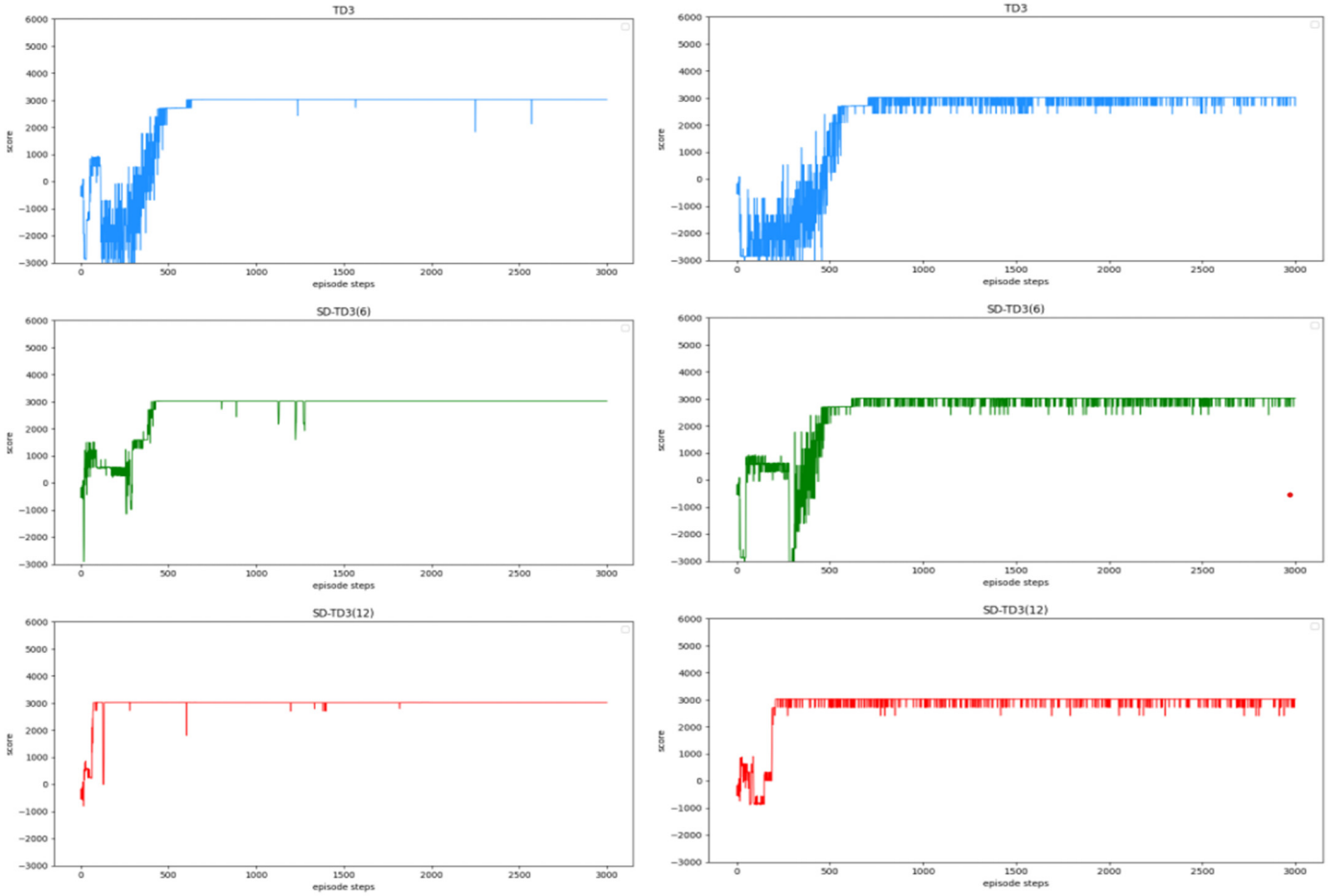

Figure 11. The best training results of the three algorithmic models. (**a**) Best training results of the model in environment 1; (**b**) Best training results of the model in environment 2.

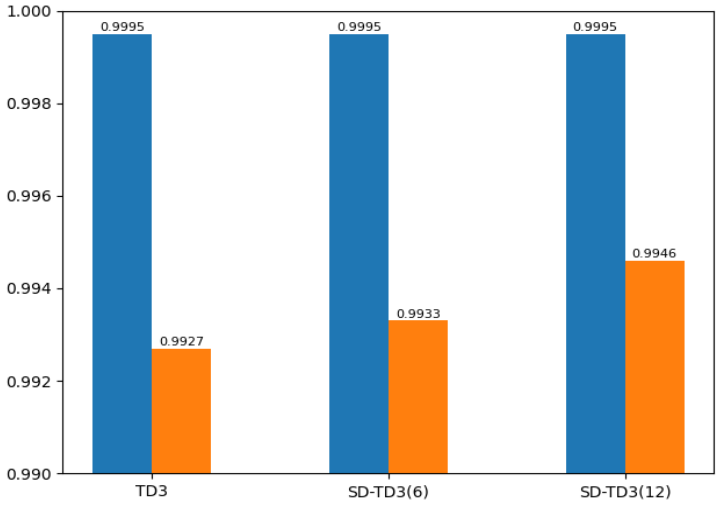

Figure 12. Success rate of all algorithmic models in both environments.

6. Conclusions

In this paper, we propose a state detection method based on the TD3 algorithm to solve the autonomous path planning problem of UAVs in low-altitude conditions. Firstly, the process of a UAV raid mission in a complex low-altitude environment was modeled, as were

the static environment and dynamic environment of low-altitude flight. Similarly, in order to solve the problem of sparse reward in traditional reinforcement learning, a dynamic reward function with heuristic guidance is set up, which can make the algorithm model converge faster. On the basis of these works, combined with the state detection method, the SD-TD3 algorithm is proposed. The simulation results show that the convergence speed of the SD-TD3 algorithm model is faster than that of the TD3 algorithm model in both a static and a dynamic environment. In the static environment, the actual task completion rate of the SD-TD3 algorithm is similar to that of the TD3 algorithm, but in the dynamic environment, the success rate of the SD-TD3 algorithm model to complete the raid task is higher than that of the TD3 algorithm, and with the detailed division of the space state information in the direction of UAV travel, the success rate of the SD-TD3 algorithm model will also improve. In general, the SD-TD3 algorithm has a faster training convergence speed and a better ability to avoid dynamic obstacles than the TD3 algorithm. The SD-TD3 algorithm needs to accurately extract environmental information to determine the position of obstacles, but in practical applications, many sensors are needed to extract and process environmental information. This paper does not study the collaborative processing method of these sensors, so it will be challenging in practical applications. In future work, it can be further studied to change the input mode of the algorithm model and input more effective environmental information to promote the algorithm model's ability to make correct decisions. At the same time, the SD-TD3 algorithm is combined and compared with other DRL algorithms, such as the PPO (proximal policy optimization) algorithm and SAC (soft actor critic) algorithm.

Author Contributions: Conceptualization, D.Z.; methodology, D.Z.; software, D.Z.; validation, D.Z., Z.X. and X.L. (Xiongwei Li).; formal analysis, D.Z.; investigation, Z.X.; resources, X.L. (Xiongwei Li), Y.Z., J.Y. and X.L. (Xi Li).; data curation, X.L. (Xi Li); writing original draft preparation, D.Z.; writing—review and editing, D.Z., J.Y., Z.X., X.L. (Xiongwei Li), Y.Z. and X.L. (Xi Li).; visualization, D.Z. and J.Y.; supervision, Z.X., X.L. (Xiongwei Li) and Y.Z.; project administration, Z.X.; funding acquisition, Y.Z. and X.L. (Xiongwei Li). All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, grant number 62071483, and the National Natural Science Foundation of China, grant number 61602505.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest: The authors declare that they have no conflict of interest or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Aggarwal, S.; Kumar, N. Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges. *Comput. Commun.* **2020**, *149*, 270–299. [CrossRef]
2. Tsourdos, A.; White, B.; Shanmugavel, M. *Cooperative Path Planning of Unmanned Aerial Vehicles*; John Wiley & Sons: Hoboken, NJ, USA, 2010.
3. Chen, Y.; Mei, Y.; Yu, J.; Su, X.; Xu, N. Three-dimensional unmanned aerial vehicle path planning using modified wolf pack search algorithm. *Neurocomputing* **2017**, *266*, 445–457.
4. Zhang, H.; Xin, B.; Dou, L.-H.; Chen, J.; Hirota, K. A review of cooperative path planning of an unmanned aerial vehicle group. *Front. Inf. Technol. Electron. Eng.* **2020**, *21*, 1671–1694. [CrossRef]
5. Cabreira, T.M.; Brisolara, L.B.; Paulo, R.F.J. Survey on coverage path planning with unmanned aerial vehicles. *Drones* **2019**, *3*, 4. [CrossRef]
6. Yao, P.; Wang, H.; Su, Z. Real-time path planning of unmanned aerial vehicle for target tracking and obstacle avoidance in complex dynamic environment. *Aerosp. Sci. Technol.* **2015**, *47*, 269–279. [CrossRef]
7. Liang, X.; Meng, G.; Xu, Y.; Luo, H. A geometrical path planning method for unmanned aerial vehicle in 2D/3D complex environment. *Intell. Serv. Robot.* **2018**, *11*, 301–312. [CrossRef]

8. Dolicanin, E.; Fetahovic, I.; Tuba, E.; Hrosik, R.C.; Tuba, M. Unmanned combat aerial vehicle path planning by brain storm optimization algorithm. *Stud. Inform. Control* **2018**, *27*, 15–24. [CrossRef]
9. Sun, P.; Guo, Z.; Lan, J.; Hu, Y.; Baker, T. ScaleDRL: A Scalable Deep Reinforcement Learning Approach for Traffic Engineering in SDN with Pinning Control. *Comput. Netw.* **2021**, *190*, 107891. [CrossRef]
10. Roberge, V.; Tarbouchi, M.; Labonté, G. Comparison of parallel genetic algorithm and particle swarm optimization for real-time 865 UAV path planning. *IEEE Trans. Ind. Inform.* **2012**, *9*, 132–141. [CrossRef]
11. Liu, Z.; Zhang, Y.; Yu, X.; Yuan, C. Unmanned surface vehicles: An overview of developments and challenges. *Annu. Rev. Control* **2016**, *41*, 71–93. [CrossRef]
12. Zhang, S.; Zhou, Y.; Li, Z.; Pan, W. Grey wolf optimizer for unmanned combat aerial vehicle path planning. *Adv. Eng. Softw.* **2016**, *99*, 121–136. [CrossRef]
13. Zhang, X.; Duan, H. An improved constrained differential evolution algorithm for unmanned aerial vehicle global route planning. *Appl. Soft Comput.* **2015**, *26*, 270–284. [CrossRef]
14. Guo, S.; Zhang, X.; Zheng, Y.; Du, Y. An autonomous path planning model for unmanned ships based on deep reinforcement learning. *Sensors* **2020**, *20*, 426. [CrossRef] [PubMed]
15. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
16. Wang, Y.; Gao, Z.; Zhang, J.; Cao, X.; Zheng, D.; Gao, Y.; Ng, D.W.K.; Di Renzo, M. Trajectory design for UAV-based internet of things data collection: A deep reinforcement learning approach. *IEEE Internet Things J.* **2021**, *9*, 3899–3912. [CrossRef]
17. Han, M.; Zhang, L.; Wang, J.; Pan, W. Actor-critic reinforcement learning for control with stability guarantee. *IEEE Robot. Autom. Lett.* **2020**, *5*, 6217–6224. [CrossRef]
18. Li, B.; Wu, Y. Path planning for UAV ground target tracking via deep reinforcement learning. *IEEE Access* **2020**, *8*, 29064–29074. [CrossRef]
19. LLillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
20. Fujimoto, S.; Hoof, H.; Meger, D. Addressing function approximation error in actor-critic methods. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 1587–1596.
21. Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; Freitas, N. Dueling network architectures for deep reinforcement learning. In Proceedings of the International Conference on Machine Learning PMLR, New York, NY, USA, 19–24 June 2016; pp. 1995–2003.
22. Ferguson, D.; Stentz, A. Using interpolation to improve path planning: The Field D* algorithm. *J. Field Robot.* **2006**, *23*, 79–101. [CrossRef]
23. Zeiwei, Z.; Wei, W.; Nengcheng, C.; Chao, W. An improved A* algorithm using unmanned aerial vehicle (uav) flight path planning. *Geomat. Inf. Sci. Wuhan Univ.* **2015**, *40*, 315–320. [CrossRef]
24. Saranya, C.; Unnikrishnan, M.; Ali, S.A.; Sheela, D.; Lalithambika, V. Ter-rain based D algorithm for path planning. *IFAC Pap.* **2016**, *49*, 178–182. [CrossRef]
25. Li, Z.; Li, L.; Zhang, W.; Wu, W.; Zhu, Z. Research on Unmanned Ship Path Planning based on RRT Algorithm. *J. Phys. Conf. Ser.* **2022**, *2281*, 12004. [CrossRef]
26. Yan, J.; Li, X.; Liu, B.; Liu, L. Multi UAV suppression of on enemy air defense based on MILP. *J. Nav. Aeronaut. Astronaut. Univ.* **2014**, *29*, 6.
27. Yang, J.; Xu, X.; Yin, D.; Ma, Z.; Shen, L. A space mapping based 01 linear model for onboard conflict resolution of heterogeneous unmanned aerial vehicles. *IEEE Trans. Veh. Technol.* **2019**, *68*, 7455–7465. [CrossRef]
28. Zhou, H.; Xiong, H.-L.; Liu, Y.; Tan, N.-D.; Chen, L. Trajectory Planning Algorithm of UAV Based on System Positioning Accuracy Constraints. *Electronics* **2020**, *9*, 250. [CrossRef]
29. Lin, C.E.; Syu, Y.M. GA/DP Hybrid Solution for UAV Multi-Target Path Planning. *J. Aeronaut. Astronaut. Aviat.* **2016**, *48*, 203–220.
30. Nazarahari, M.; Khanmirza, E.; Doostie, S. Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm. *Expert Syst. Appl.* **2019**, *115*, 106–120. [CrossRef]
31. Cheng, Y.; Zhang, W. Concise deep reinforcement learning obstacle avoidance for underactuated unmanned marine vessels. *Neurocomputing* **2018**, *272*, 63–73. [CrossRef]
32. Zhang, B.; He, M.; Chen, X.; Wu, C.; Liu, B.; Zhou, B. Application of Improved DDPG Algorithm in Automatic Driving. *Comput. Eng. Appl.* **2019**, *55*, 264–270.
33. Hong, D.; Lee, S.; Cho, Y.H.; Baek, D.; Kim, J.; Chang, N. Energy-Efficient Online Path Planning of Multiple Drones Using Reinforcement Learning. *IEEE Trans. Veh. Technol.* **2021**, *70*, 9725–9740. [CrossRef]
34. Li, B.; Gan, Z.; Chen, D.; Aleksandrovich, D.S. UAV maneuvering target tracking in uncertain environments based on deep reinforcement learning and meta-learning. *Remote Sens.* **2020**, *12*, 3789. [CrossRef]
35. Papachristos, C.; Kamel, M.; Popović, M.; Khattak, S.; Bircher, A.; Oleynikova, H.; Dang, T.; Mascarich, F.; Alexis, K.; Siegwart, R. Autonomous exploration and inspection path planning for aerial robots using the robot operating system. In *Robot Operating System (ROS)*; Springer: Cham, Switzerland, 2019; pp. 67–111.
36. Zhang, F.; Li, J.; Li, Z. A TD3-based multi-agent deep reinforcement learning method in mixed cooperation-competition environment. *Neurocomputing* **2020**, *411*, 206–215. [CrossRef]

37. Memarian, F.; Goo, W.; Lioutikov, R.; Niekum, S.; Topcu, U. Self-supervised online reward shaping in sparse-reward environments. In Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 27 September–1 October 2021; pp. 2369–2375.
38. Wu, Y.H.; Yu, Z.C.; Li, C.-Y.; He, M.J.; Hua, B.; Chen, Z.M. Reinforcement learning in dual-arm trajectory planning for a free-floating space robot. *Aerosp. Sci. Technol.* **2020**, *98*, 105657. [CrossRef]