Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 14, No. 1, 2025

Research on a SAC-Based Algorithm for UAV Autonomous Navigation and Obstacle Avoidance

Zewen Cui *

College of Intelligent Systems Science and Engineering, Harbin Engineering University, Harbin, Heilongjiang Province, 150001, China
*** Corresponding author Email:** cuizewan_123@163.com

Abstract: The traditional reinforcement learning algorithms face the problems of poor complex spatial adaptability and inefficient exploration in the autonomous UAV pathfinding and obstacle avoidance tasks in three-dimensional unknown environments. To this end, this paper combines the dynamically decayingε-greedy exploration strategy, the dynamic learning rate adjustment mechanism and the offline strategy SAC (Soft Actor-Critic) algorithm based on the maximum entropy reinforcement learning framework, and proposes an autonomous pathfinding and obstacle avoidance algorithm based on SAC, aiming to enhance the obstacle avoidance ability of UAVs in complex environments.ε-Greedy exploration strategy promotes exploration in the early stage of training and exploitation in the later stage, which helps the intelligent body to find the globally optimal strategy in complex environments and avoids falling into the local optimum in the early stage; the dynamic learning rate adjustment mechanism can refine the strategy and improve the training stability and final performance. The results in the test environment show that this algorithm makes the UAV collision rate decrease significantly, and the timeout rate is close to 0, which effectively enhances the algorithm's adaptability and stability to the environment.

Keywords: Unmanned Aircraft; Obstacle Avoidance; SAC; Autonomous Navigation.

1. Introduction

In recent years, the basic research as well as practical application of Unmanned Aerial Vehicles (UAV) has been gradually and widely emphasized by scholars at home and abroad[1] . With the in-depth research in the field of optimal control theory, new sensor technology and artificial intelligence, the autonomous flight technology of UAV is also developing vigorously. At present, UAVs have been widely used in military, civil and commercial fields, such as military reconnaissance, disaster monitoring, express delivery, aerial photography, etc.[2] . Therefore, how UAVs can efficiently and accurately avoid obstacles and reach the end point in these complex scenarios is still an important issue.

For UAV autonomous obstacle avoidance technology, researchers have proposed a variety of algorithms, which can be roughly categorized into traditional path planning algorithms and reinforcement learning algorithms. Traditional path planning methods are divided into graph search-based path planning algorithms, sampling-based path planning algorithms and artificial potential field methods. Although graph search-based path planning algorithms (e.g., A* algorithm, Dijkstra's algorithm, etc.) perform well in static environments, they tend to lack flexibility and adaptability in complex and uncertain environments, and are highly dependent on environmental information [3, 4] Sampling-based methods (e.g., RRT) are effective in realizing obstacle avoidance navigation through the asymptotic optimality property, but such algorithms rely on random sampling resulting in low efficiency [5]. Artificial potential field method by constructing an artificial potential field the target and obstacles are regarded as objects with gravitational and repulsive forces on the UAV respectively, and the UAV moves along the combined force of gravitational and repulsive forces, this algorithm is a feedback control strategy, which is robust to control and sensing errors, but it does not necessarily find the optimal solution [6].

Reinforcement Learning (RL), as a method in which an agent learns optimal strategies through interaction with the environment, is particularly suitable for tasks where the environment is dynamically changing, the model is unknown, or optimal solutions cannot be directly obtained. Therefore, it has gradually been applied in the field of drone path planning. Amala Sonny et al. proposed a Q-learning method based on shortest distance priority strategy to improve the accuracy and efficiency of drone path planning [7]. Qi et al. proposed an FD-PPO algorithm based on frequency decomposition, which efficiently guides drones to complete path planning tasks by decomposing rewards into multi-dimensional frequency rewards and then calculating frequency returns [8]. Omar Bouhamed et al. combined the continuous deep deterministic policy gradient (DDPG) algorithm with drone 3D obstacle avoidance to solve the problem of exploration efficiency in continuous control [9]. Luo et al. introduced the priority experience playback mechanism into the TD3 algorithm to transform its experience pool, so that the intelligent physical ability can distinguish the priority of different experience samples, thus improving the sampling efficiency and shortening the training time [10]. Xue et al. proposed a framework of deep RL and developed a value network that integrates crowd and static obstacle information, utilizing spatiotemporal reasoning and LiDAR data to help the agent effectively understand the surrounding environment [11].

In summary, this paper takes the autonomous UAV pathfinding and obstacle avoidance task in a three-dimensional unknown environment as the research object, and through the decision-making ability of RL, the UAV can autonomously avoid all kinds of static and dynamic obstacles and reach the end point. Specifically, this paper first establishes the Markov decision-making process of autonomous UAV navigation task, designs the non-sparse reward function, and realizes the autonomous pathfinding and obstacle avoidance of UAV based on SAC algorithm, based on which, this paper introduces the dynamically decayingε -

greedy exploring strategy, and adds the dynamic learning rate adjustment mechanism, so that the UAV can explore a wider action space and improve the stability of the training, and finally, compare and validate its performance with other algorithms in the testing environment.

2. Automatic Obstacle Avoidance Algorithm Based on Reinforcement Learning

2.1. Markov Decision Process

Reinforcement Learning (RL) is a method that emphasizes the ability of an intelligent body to maximize its expected benefits by acting based on its environment. The agent interacts with the environment through policy-driven actions, generates new data, and then uses the new data to modify its own action strategies, and so on iteratively optimizes its strategies until it learns the action strategies needed to complete the task. Since the current state generated by the intelligent body interacting with the environment contains useful information needed for future prediction, and the state generated in the past is irrelevant for prediction, which satisfies Markov property, reinforcement learning can be described as Markov Decision Process (MDP). Markov Decision Process is denoted as <S, A, P, R, γ>, where S is the set of state space; A is the set of action space; P is the state transfer probability, $P_{ss'}^{a} = \mathbb{E}(S_{t+1} = s' | S_t = s, A_t = a)$; R is the reward function, $R_{(s_t=s,a_t=a)} = E[R_t \mid s_t = s, a_t = a]$;γ is the discount factor, the ultimate goal of this process is to find the strategy with the greatest payoff among the various strategies π. The process is based on a set of strategies π, which is the most efficient one. The strategies satisfy the function $\pi(a \mid s) = p[A_t = a \mid S_t = s]$.

The task of this paper is to let the UAV interact with the environment during flight, avoid various static and dynamic obstacles without collision, and finally reach the target point successfully. Due to the local observability of the environment, the task can be modeled as a partially-observable Markov decision-making process, which can be incorporated into the framework of reinforcement learning by considering the use of environment observations instead of complete state quantities.

2.2. SAC algorithm Framework

SAC (Soft Actor-Critic) is an off-policy actor-critic deep reinforcement learning algorithm grounded in the maximum entropy framework, wherein the actor jointly maximizes expected reward and policy entropy, i.e., completing the task as randomly as possible [12] [13]. Maximizing entropy allows the strategy to be motivated to explore the space more extensively, capturing multiple near-optimal behavioral patterns while abandoning apparently unpromising pathways, making the model more robust in the face of disturbances.

The SAC algorithm framework is shown in Figure 1 and contains the intelligences, the environment, the experience buffer pool, one policy network, two evaluation networks and two goal evaluation networks.

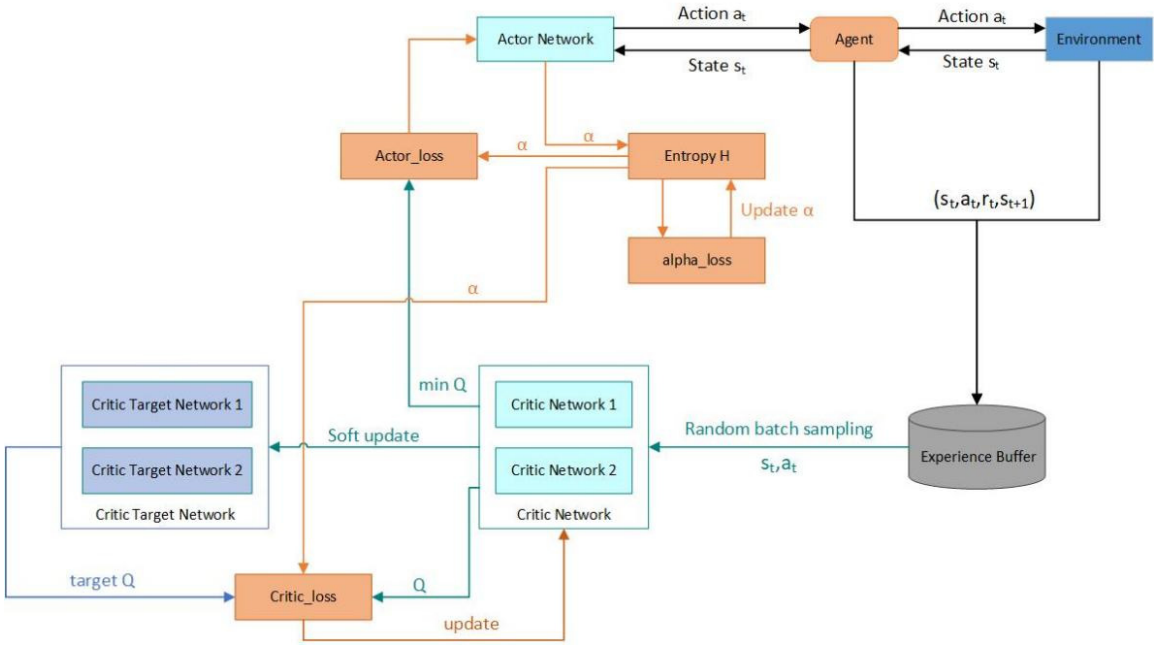

Figure 1. SAC algorithm framework

Actor network is used to output mean μ and standard deviation σ conditioned on current state s_t. The policy $\pi(\cdot \mid s_t)$ is usually modeled as a Gaussian distribution with mean μ and standard deviation σ. Actions are sampled by reparameterization, and the intelligent body receives a reward r_t after executing the action a_t, and the state changes to s_{t+1}. Since the SAC algorithm takes into account the maximum entropy of the output action, the policy optimization objective formula with entropy regularization is:

$$J(\pi) = \mathbb{E}_{(s_t,a_t)}\left[\sum_{t=0}^{\infty} \gamma^t \left(r(s_t, a_t) + \alpha H\left(\pi(\cdot \mid s_t)\right)\right)\right] \quad (1)$$

Where $H\left(\pi(\cdot \mid s_t)\right)$ is the entropy of the strategy in the state s_t, $H\left(\pi(\cdot \mid s_t)\right) = -log\pi(\cdot \mid s_t)$, γ is discount factor, and α is an automatically adjustable temperature coefficient that regulates the entropy-reward trade-off to modulate optimal policy stochasticity.

The Critic network is used to estimate the expected cumulative reward (including the entropy reward) for a given state and action.SAC uses two Critic networks and the

corresponding target network to reduce the overestimation bias. The network outputs two Q-values based on the s_t, a_t of the intelligent body, and the minimum of the two Q-values is taken as the target value when updating, which provides a more conservative estimation. The Q-function equation is:

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}}[V(s_{t+1})] \quad (2)$$

$V(s_t)$is the state-value function that represents the value at s_t. The state-value function formula is:

$$V(s_t) = \mathbb{E}_{a_t \sim \pi}\left[Q(s_t, a_t) - \alpha \log \pi_\varphi\left(a_t \mid s_t\right)\right] \quad (3)$$

In addition, SAC constructs a Critic target network with the same structure as the Critic network but with different parameters, which outputs the target $\widehat{Q}$ value, and then obtains the target value$\hat{y}$, which improves the stability of neural network training.The Critic target network also takes the smallest of the two $\widehat{Q}$ values as the target value at the time of updating, which mitigates overestimation bias at the time of training.The loss function of the Critic network adopts the Bellman equation's mean-square error:

$$L_Q(\theta) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \sim D}[(Q_\theta(s_t, a_t) - \hat{y})^2] \quad (4)$$

The target value $\hat{y}$ Calculation formula:

$$\hat{y} = r_t + \gamma\left(\min_{j=1,2} Q_{\widehat{\theta_j}}\left(s_{t+1}, a_{t+1}\right) - \alpha \log \pi_\varphi\left(a_{t+1} \mid s_{t+1}\right)\right) \quad (5)$$

The loss function of Actor network is:

$$L_\pi(\varphi) = \mathbb{E}_{s_t \sim D}\left[\mathbb{E}_{a_t \sim \pi}\left[\alpha \log \pi_\varphi\left(a_t \mid s_t\right) - Q_\theta(s_t, a_t)\right]\right] \quad (6)$$

According to the loss functions $L_Q(\theta)$ and $L_\pi(\varphi)$, the parameters of Actor network and Critic network θ, φ are updated inversely, and the parameter $\hat{\theta}$ of Critic target network is softly updated:$\hat{\theta} \leftarrow \tau\theta + (1-\tau)\hat{\theta}$, and the softly update coefficient τ is 0.005.

SAC automatically adjusts the temperature coefficientα so that the entropy of the strategy is close to the target entropy. The loss function is:

$$L(\alpha) = \mathbb{E}_{s_t \sim D}\left[-\alpha\left(\log \pi\left(a_t \mid s_t\right) + \widehat{H}\right)\right] \quad (7)$$

$\widehat{H}$ is the target entropy, which is usually set to the negative action dimension, and automatic adjustment is achieved by minimizing the entropy loss.

3. Experimental Design

3.1. Experimental Scene

Due to its small size and convenient features, UAV is often used to perform tasks in complex three-dimensional space. In order to better simulate the real environment, this paper designs a diversified flight scene containing static and dynamic obstacles, and the specific settings are as follows:

(1) Experimental environment: the spatial size of the environment in this paper is set to60m × 60m × 30m, while the flight altitude of the UAV is limited to no less than 0.5m, and the safety distance is set to 2m, and different obstacles are introduced into this environment to enhance the spatial complexity.

(2) Obstacle setting: in order to simulate the complex environment in reality, this paper sets up 5 static obstacles and 8 dynamic obstacles as shown in Figure 2, in which the dynamic obstacles include the motion of circular motion, spiral motion, compound periodic motion, asymmetric motion, straight line + fluctuation, and other kinds of complex trajectories (the motion trajectory is shown in Figure 3), and the radii and speeds of these obstacles vary, and randomly select 5 static obstacles and 8 dynamic obstacles from them when training. Five static obstacles and five dynamic obstacles are randomly selected for training, so as to further test the autonomous obstacle avoidance ability of the UAV.

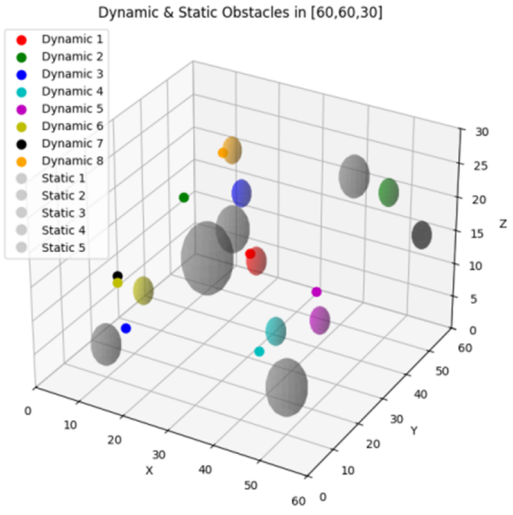

Figure 2. Various types of obstacle

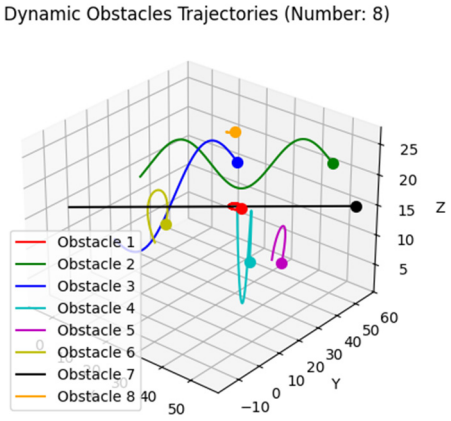

Figure 3. Dynamic obstacle trajectory

(3) UAV flight task: the initial coordinates of the UAV are (10,10,5) m, and the endpoint coordinates are (50,50,10)m. The UAV needs to start from the starting point, fly and avoid all kinds of obstacles, and arrive at the endpoint without collision.

3.2. Strategy Network Design

The input dimension of the strategy network is the same as the state dimension, and the hidden layer adopts two layers of full connectivity (256 neurons per layer), processing the input state vector through the ReLU activation function. The output layer adopts a two-branch output structure: one branch generates the action meanμ, and the other branch generates the bounded logarithmic standard deviation $\log\sigma \in [-20,2]$, which constructs the action probability distribution. A stochastic strategy is used in the action sampling phase: sampling from a Gaussian distribution is processed by tanh transformation and physical constraints, and the modified log probability is computed to compensate for the change in probability density caused by the tanh transformation. Meanwhile, the strategy network integrates a safety mechanism: when the distance to the obstacle is detected to be less than 80% of the safety threshold, it immediately

switches to a preset emergency obstacle avoidance action library. This action library contains five sets of evasive actions (e.g., left turn, right turn, dive, tilt, and climb) for obstacles in different directions, which achieve rapid extrication through thrust reinforcement. In addition, this experiment introduces an exploration strategy in the normal decision-making process: 10% probability of completely randomized actions (ε-greedy), and the remaining 90% adds attenuated Gaussian noise (noisy at the early stage of training and gradually reduced at the later stage) to the network output actions, thus helping the intelligent body to find the globally optimal strategy in a complex environment, and avoiding falling into the local optimum at an early stage. On this basis, this paper utilizes the LambdaL R scheduler to achieve dynamic adjustment of the learning rate: **by** fast learning in the early stage and fine tuning in the later stage, the strategy is refined to improve the training stability and final performance. This architecture enables UAVs to realize safe navigation in dynamic obstacle environments while obeying the laws of physics.

3.3. State Space Design

In this paper, a 15-dimensional state space is designed, which consists of three parts: the first part is the UAV's own normalized 3D position$(\mathrm{x}, \mathrm{y}, \mathrm{z})$, the second part is the UAV's own normalized 3D velocity$(\mathrm{v_x}, \mathrm{v_y}, \mathrm{v_z})$, and the third part is the relative positions of the three nearest obstacles, each contributing three dimensions, for a total of 9 dimensions. Based on the working principle of analog sensors, this paper adopts the design of prioritizing just-in obstacles, while reducing the computational complexity and ensuring the stability of the neural network structure.

3.4. Action Space Design

In order to ensure the maneuverability of the UAV in complex environments, this paper designs the continuous action space to meet the demand. As shown in Table 1, the three-axis continuous control volume is mapped to the three-dimensional continuous space, which simulates the maneuverability of the UAV in the real environment.

Table 1. Action space table

Maneuver	Range of maneuver	Description
v_x	[-1.5m/s,1.5m/s]	x velocity
v_y	[-1.5m/s,1.5m/s]	y velocity
v_z	[0.0m/s,1.5m/s]	z velocity

3.5. Reward Function Design

The reward function is a core element in defining environmental feedback in reinforcement learning, which directly affects the performance of learning algorithms. The objective of this study is to enable the UAV to avoid various static and dynamic obstacles and reach the end point safely in a complex environment. Based on the above objectives, this paper designs a non-sparse reward function to form a safe navigation strategy through progressive reward gradient and asymmetric punishment mechanism, and at the same time with stagnation detection to prevent the training from falling into a local optimum, to guide the UAV to explore the environment more efficiently and to improve the convergence. The specific design is as follows:

(1) Base navigation reward

To guide UAV navigation toward targets, the reward scales inversely with distance: positive incentives when approaching the goal and negative penalties when moving away, i.e., the UAV is incentivized to move towards the target point by successive distance differences, so the distance reward formula is:

$$R_1 = d_{\mathrm{prev}} - d_{\mathrm{curr}} \tag{8}$$

Where d_{prev} and d_{curr} denote the Euclidean distance to the target at the previous and current moment respectively.

(2) Target Area Enhancement Bonus

The UAV tends to rotate around the target point as it approaches it instead of reaching the end point directly, so it needs to be given an additional gradient reward to promote accurate navigation. Specifically:

$$R_2 = \begin{cases} 10 \times (3 - d_{\mathrm{curr}}) & d_{\mathrm{curr}} < 3 \\ 20 \times (2 - d_{\mathrm{curr}}) & d_{\mathrm{curr}} < 2 \\ 200\ (Terminate\ round) & d_{\mathrm{curr}} < 1 \end{cases} \tag{9}$$

At $d_{\mathrm{curr}} < 3$, a target attraction zone is established to attract the drone towards the target point; at $d_{\mathrm{curr}} < 2$, the reward strength is increased to reinforce end guidance so that the drone reaches the end point more quickly; at $d_{\mathrm{curr}} < 1$, the drone reaches the end point and the round ends.

(3) Obstacle constraint penalty

Inversely penalize the drone when it approaches an obstacle; when it collides with an obstacle, give a larger penalty and end the round. The details are as follows:

$$R_3 = \begin{cases} -20(Terminate\ round) & d_{\mathrm{obs}} < 2.0 \\ -\dfrac{2}{d_{\mathrm{obs}}} & 2.0 \le d_{\mathrm{obs}} < 3.5 \end{cases} \tag{10}$$

Where d_{obs}represents the distance between the drone and the nearest obstacle, set $2.0 \le d_{\mathrm{obs}} < 3.5$ area as the caution area, and set $d_{\mathrm{obs}} < 2.0$ as the collision area.

(4) Hover Penalty

The drone may accumulate navigation rewards by hovering, so it is set that when the maximum distance moved by the drone in 20 consecutive steps is less than 0.1, the drone is considered to be in a hovering state and a hovering penalty is given to it. The specifics are as follows:

$$\mathrm{R_4} = -10 \qquad \Delta d_{\mathrm{recent}} < 0.1 \tag{11}$$

Where $\Delta d_{\mathrm{recent}}$ is the maximum distance difference of 20 consecutive steps.

In summary, the final reward function is:

$$R_{\mathrm{total}} = R_1 + R_2 + R_3 + R_4 \tag{12}$$

4. Simulation and Analysis

4.1. Algorithm Parameter Setting

The algorithm parameters used in this paper are shown in Table 2, in which the exploration rate is linearly decaying, which is convenient for the UAV to explore the state space with a larger probability at the beginning of the training to avoid falling into the local optimum too early, and stabilize policies throughout later phases then improve the efficiency.

4.2. Experimental Analysis

To evaluate algorithmic performance, this paper trains and tests the SAC algorithm with DDPG and TD3 in the simulation environment [14, 15]. At the beginning of training, the initial reward is set to zero, and when the UAV flies to the end point or close to the obstacle, it will be rewarded and penalized accordingly. Since the reward value is the sum of the rewards obtained at each step, a larger reward value indicates a higher probability of the UAV avoiding obstacles and reaching the end point, i.e., a better learned strategy. Since

the obstacle avoidance task may end early at a later stage, or the penalty for collision is not enough to offset the accumulated rewards resulting in a reward value that does not reflect the real situation well, this paper chooses the average reward value calculated by the 50-round sliding average method as the evaluation index to eliminate the above interference.

Table 2. Algorithm parameters setting table

Parameter	Value
Discount factor γ	0.99
Soft update factor τ	0.005
Initial α	0.2
Experience playback buffer pool	100000
Training batch size	128
Maximum step size for a single training batch	500
Initial exploration rate	0.5
Minimum exploration rate	0.01
Exploration rate decay coefficient	0.995

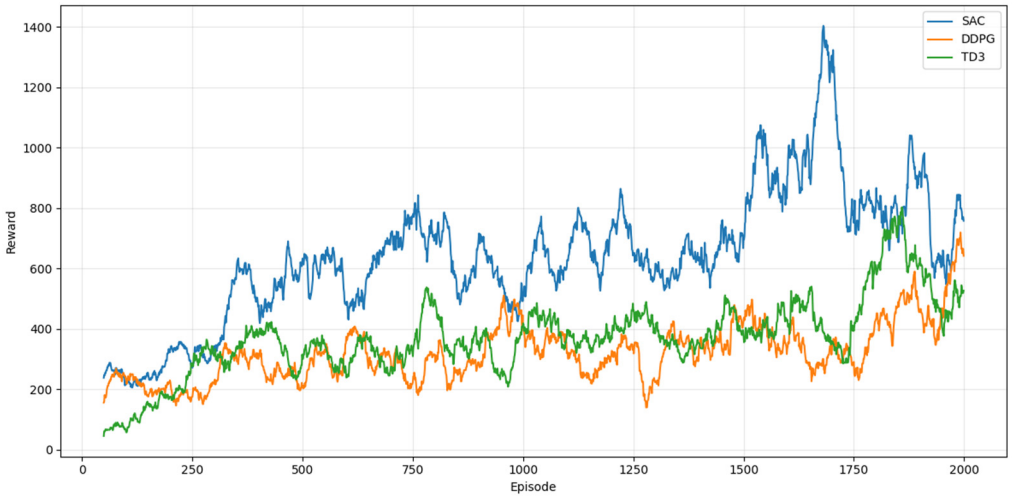

Figure 4. Change curve of average reward value (50-round sliding average)

Figure 4 shows the average reward value change curve of each algorithm. From the figure, it can be found that the DDPG algorithm reward growth is gentle and smooth, with no significant fluctuations throughout, but the reward value is the lowest, which indicates that the algorithm strategy is conservative, weak in exploration, and prone to falling into local optimization. Compared to the DDPG algorithm, the TD3 algorithm has a higher average reward value and is smooth in the later stages, reflecting the suppression of action noise by the double Q-learning mechanism, but the potential is limited compared to the SAC algorithm, which introduces maximum entropy, encourages the exploration of a greater variety of actions, and has a higher efficiency in the complex space, so the reward value rises rapidly from rounds 0 to 750 and has a high reward value after rounds 1400, and through the double Q-learning mechanism and soft updates, the performance fluctuation after 1500 rounds is made to be more effective, exhibiting greater stability compared to earlier rounds. This shows that SAC algorithm has high sample efficiency, high reward limit, strong adaptability to complex environment, better performance than DDPG algorithm and TD3 algorithm, and gives full play to the maneuverability of UAV.

Figure 5 shows the success rate curve of each algorithm per 50 rounds of mission. From the figure, it can be found that the success rate of DDPG fluctuates from 40% to 60%, which indicates that the algorithm is not ideal for this task.TD3 algorithm has a lower success rate than DDPG in the first 150 rounds, but the success rate gradually increases in the later rounds and is higher than that of DDPG in most of the cases, with a peak of 76%, which proves the effectiveness of the delayed updating strategy of this algorithm. The success rate of the SAC algorithm fluctuates within the range of 70% to 80%, generally outperforming both the DDPG and TD3 algorithms. This demonstrates SAC's strong capacity for rapid learning, effective adaptation to complex tasks, and superior overall performance.

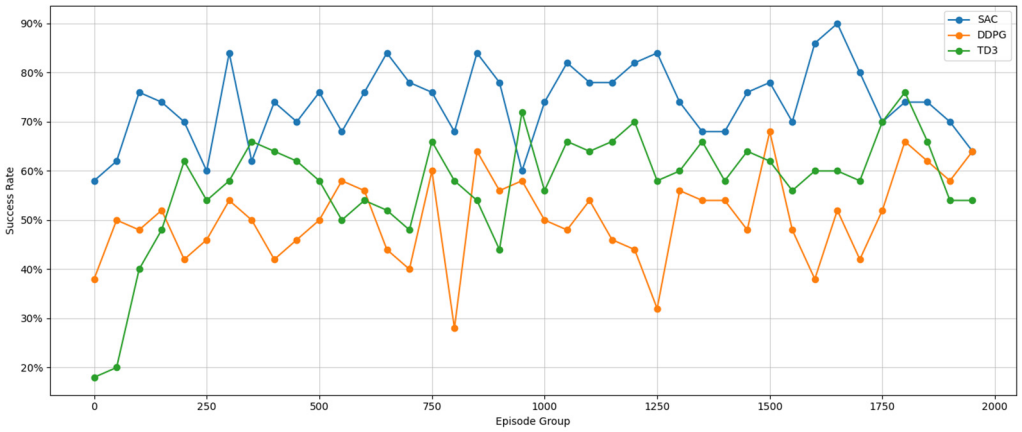

Figure 5. Success rate change curve

In order to further verify the performance of the algorithms, this paper tests each algorithm in the test environment. The starting coordinates of the UAV in the testing environment are (10,10,5) m, and the ending coordinates are (50,50,10) m. The static obstacles are set to five, and the dynamic obstacles are increased to seven. The task requires the UAV to first start from the starting point, automatically avoid obstacles when encountering obstacles, and finally arrive at the end point without collision.

Table 3 shows the results of collision rate and timeout rate of each algorithm for 2000 rounds in the test environment. The collision rate represents the ratio of the number of collisions rounds to the total number of rounds, and the timeout rate represents the ratio of the number of rounds that have not reached the end point after taking the whole step length to the total number of rounds.

Table 3. Algorithm results for each metric in the test environment

Algorithm	Collision rate	Timeout rate
SAC	28.50%	0.00%
TD3	33.05%	6.15%
DDPG	34.60%	9.25%

From Table 3, it can be found that the collision rate of SAC is lower than that of DDPG algorithm and TD3 algorithm, and the timeout rate is even 0, which indicates that SAC has achieved a good balance between environment exploration and obstacle avoidance decision-making, and further verifies that the algorithm's maximum entropy strategy, ε -greedy exploration, and double Q-learning network design can significantly improve its environment adaptability, so as to find the optimal strategy more quickly and achieve good training results.

5. Conclusion

In this paper, taking the autonomous UAV pathfinding and obstacle avoidance task in a three-dimensional unknown environment as the research background, the SAC algorithm is combined with the dynamically decaying ε -greedy exploration strategy and the dynamic learning rate adjustment mechanism, and a dynamic obstacle avoidance algorithm based on SAC for UAVs is designed. Simulation experiments show that this algorithm can effectively cope with the autonomous obstacle avoidance task of UAVs in complex environments, especially in multi-dynamic obstacle environments, and improves the training speed and stability, so that the maneuverability of UAVs can be effectively utilized.

Although the UAV can complete the obstacle avoidance task well by the SAC algorithm, the collision rate of 28.5% is still high relative to the actual application scenario, In addition, as the performance of the algorithm has not been validated in real environments, its generalization ability still needs further evaluation and improvement. In the future, it can be further tested in the real environment, and a variety of environment sensing methods can be combined with this algorithm to enhance its value for practical application.

References

[1] Tao, Y., & Li, P. (2014). An overview of unmanned aircraft system development and key technologies. Aeronautical Manufacturing Technology,57(20),34-39.

[2] Liu, M., & Shi, H. (2025). Autonomous obstacle avoidance strategy for UAV based on EFRE-SAC. Computer System Applications,34(06),53-61.

[3] Wang, Z., & Xiang, X. (2018). Improved astar algorithm for path planning of marine robot. In 2018 37th chinese control conference (CCC), IEEE, 5410-5414.

[4] Prasad, N. L., & Ramkumar, B. (2022). 3-D deployment and trajectory planning for relay based UAV assisted cooperative communication for emergency scenarios using Dijkstra's algorithm. IEEE Transactions on Vehicular Technology, 72(4), 5049-5063.

[5] Du, Z., & Liu, S. (2018). Asymptotical RRT-based path planning for mobile robots in dynamic environments. In 2018 37th Chinese control conference (CCC), IEEE, 5281-5286.

[6] Chen, H., Chen, H., & Qiang, L. (2020). Multi-UAV 3D formation path planning based on improved artificial potential field. Journal of System Simulation, 32(3), 414-420.

[7] Sonny, A., Yeduri, S. R., & Cenkeramaddi, L. R. (2023). Q-learning-based unmanned aerial vehicle path planning with dynamic obstacle avoidance. Applied Soft Computing, 147, 110773.

[8] Qi, C., Wu, C., Lei, L., Li, X., & Cong, P. (2022). UAV path planning based on the improved PPO algorithm. In 2022 Asia Conference on Advanced Robotics, Automation, and Control Engineering (ARACE), IEEE, 193-199.

[9] Bouhamed, O., Ghazzai, H., Besbes, H., & Massoud, Y. (2020). Autonomous UAV navigation: A DDPG-based deep reinforcement learning approach. In 2020 IEEE International Symposium on circuits and systems (ISCAS), IEEE, 1-5.

[10] Luo, X., Wang, Q., Gong, H., & Tang, C. (2024). UAV path planning based on the average TD3 algorithm with prioritized experience replay. IEEE Access, 12, 38017-38029.

[11] Xue, B., Zhou, F., Wang, C., Gao, M., & Yin, L. (2024). Robot Mapless Navigation in VUCA Environments via Deep Reinforcement Learning. IEEE Transactions on Industrial Electronics, 72(1), 639-649.

[12] Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., ... & Levine, S. (2018). Soft actor-critic algorithms and applications. arXiv preprint arXiv:1812.05905.

[13] Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In International conference on machine learning, Pmlr, 1861-1870.

[14] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., ... & Wierstra, D. (2015). Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

[15] Fujimoto, S., Hoof, H., & Meger, D. (2018). Addressing function approximation error in actor-critic methods. In International conference on machine learning, PMLR, 1587-1596.