Autonomous Obstacle Avoidance Algorithm for Unmanned Aerial Vehicles Based on Deep Reinforcement Learning

Yuan Gao, Ling Ren, Tianwei Shi, Teng Xu, Jianbang Ding

***Abstract*—To overcome the challenges of obstacle avoidance for Unmanned Aerial Vehicles (UAVs) in autonomous flights, this paper proposes the Dual Experience Attention Convolution Soft Actor-Critic (DAC-SAC) algorithm. This algorithm integrates a dual experience buffer pool, a self-attention mechanism, and the Soft-Actor-Critic algorithm with a convolutional network. The dual experience buffer pools are used to solve the problem of ineffective UAV training due to the scarcity of successful training data. To overcome the drawbacks of the original Soft Actor-Critic (SAC) algorithm in handling image data, a Convolutional Neural Network (CNN) is applied to reconstruct the actor and critic network, allowing for better image feature extraction and classification. Furthermore, a self-attention mechanism is employed by adding a convolutional self-attention layer to the network. This modification enables dynamic adjustments for the attention weights based on varying input image features, effectively addressing focus-related challenges. Two simulation experiments are performed and the DAC-SAC algorithm achieves a 99.5% success rate in a known environment and an 84.8% success rate when dealing with an unknown environment. These results confirm that the proposed algorithm enables autonomous obstacle avoidance for UAVs even when considering depth images as input.**

***Index Terms*—Deep Reinforcement Learning; DAC-SAC; UAV; Self-Attention; Obstacle Avoidance**

I. Introduction

A multi-rotor aircraft, a type of Unmanned Aerial Vehicles (UAVs), integrates automatic control, image processing, and other technologies. It has gathered increasing attention in various domains, including agricultural plant protection, security monitoring, and research and rescue activities [1]-[3]. In recent years, vision-based autonomous UAV flight obstacle avoidance has become a prominent area of research.

When flighting in unknown environments, accurately describing the environment through mathematical models becomes challenging. Therefore, Some researchers have taken advantage of Deep Learning's (DL) superior performance in visual tasks, combining it with drones to gain a sense of the environment. For instance, a model based on a convolutional neural network (CNN) was used as an indoor environment classifier to achieve indoor monocular navigation of drones [4]. Deploying images as inputs to the CNN allows UAV navigation in complex forest environments [5]. Moreover, there exist approaches aiming to train UAV controllers to autonomously predict the instance when a UAV may crash [6]. However, DL-based methods face challenges, such as excessive reliance on labeled data, high computational resource requirements, and limited model generalization capabilities. To address these challenges, researchers have combined deep reinforcement learning with UAVs.

Deep Reinforcement Learning (DRL) was realized by combining DL with Reinforcement Learning (RL). RL aims to learn optimal strategies through the interaction of an agent with the environment. Thus, Deep Neural networks (DNNS) were used in DRL to fit policies or value functions. It enables agents to directly perceive information in high-dimensional spaces and learn more complex strategies.

In the application of the DRL algorithm. The Proximal Policy Optimization (PPO) was used to improve and adjust stock trading strategies [7]. Therefore, a virtual network function service chain (VNF-SC) deployment algorithm based on DRL was proposed [8]. In addition, DRL has also been applied to autonomous obstacle avoidance and tracking of UAV flight paths [9]-[10].

However, the use of deep networks in deep reinforcement learning can require a lot of computational resources, resulting in excessively lengthy reasoning and training times. Such as YOLOv6 and Visual Transformer network [11]-[12].

In addition, shallow network structures may not efficiently extract the required features from the visual information. Therefore, neural networks in DRL algorithms must be crucially tailored to tackle these challenges. For instance, the Variational Auto-Encoder (VAE) was applied to process image information. Its generated features are input into the DRL algorithm to avoid obstacles in UAV flights [13].

Therefore, UAVs' autonomous flight obstacle avoidance can be achieved using DRL. In addition, UAVs acquire data through interactions with their environment, reducing the

Manuscript received July 19, 2023; Revised January 3, 2024.

This work was supported by the Natural Science Foundation project of Liaoning Province (2021-KF-12-06).

Yuan Gao is a postgraduate student of School of Computer Science and Software Engineering, University of Science and Technology Liaoning, Anshan, 114051, China. (e-mail: 1411251936@qq.com).

Ling Ren is a Lecturer of School of Innovation and entrepreneurship, University of Science and Technology Liaoning, Anshan, 114051, China (corresponding author, phone: 152-4220-3353; e-mail: 176878392 @qq.com)

Tianwei Shi is an associate Professor of School of Computer Science and Software Engineering, University of Science and Technology Liaoning, Anshan, 114051, China. (e-mail: tianweiabbcc @163.com).

Teng Xu is a postgraduate student of School of Computer Science and Software Engineering, University of Science and Technology Liaoning, Anshan, 114051, China. (e-mail: 1220175209@qq.com).

Jianbang Ding is a postgraduate student of School of Computer Science and Software Engineering, University of Science and Technology Liaoning, Anshan, 114051, China. (e-mail: jianbang0219@qq.com).

necessity, to some extent, for data annotation and enhancing the model's generalization capabilities.

Tackling the aforementioned challenges, the development of an adaptive and intelligent obstacle avoidance algorithm to complete autonomous flight obstacle avoidance for UAVs has become an urgent research priority to deal with. This algorithm considers input depth images and incorporates the Soft-Actor-Critic (SAC), the dual experience buffer pool, the convolutional actor-critic network, and the self-attention mechanism algorithms. This proposed model trains the UAV to complete the autonomous flight obstacle avoidance task in a simulation environment.

In this study, the Unreal Engine 4 (UE4) was deployed to build the simulation training and testing environment. Thus, depth images, collected from the UE4 using AirSim, were employed as inputs to facilitate the autonomous flight obstacle avoidance of UAVs in simulation environments. As a result, experimental findings demonstrate that, compared to other DRL algorithms, the proposed DAC-SAC algorithm exhibits faster convergence in UAV obstacle avoidance tasks, performs exceptionally well in obstacle avoidance in the training environment, and demonstrates a superior degree of adaptability in unfamiliar environments.

To sum up, the remainder of this paper was organized as follows: Section II provides an overview of the related work regarding UAV obstacle avoidance navigation methods. Section III outlines the methods developed in this study. In Section IV, a detailed presentation of the experimental results was generated, and Section V summarizes the findings and limitations of this paper and proposes future research directions.

II. Related Work

A. Traditional Methods

In conventional approaches, UAVs rely on sensors to identify obstacles, employ predefined strategies for avoidance, and devise flight paths using path planning algorithms. The optimal three-dimensional terrain obstacle avoidance path was computed using a combination of dynamic programming and tree search [14]. Moreover, to optimize obstacle avoidance, an algorithm centered on a single grid point was used [15]. The obstacle avoidance path-solving model, based on mathematical optimization methods, was intuitive and easy to grasp. However, when constraint conditions became more complex, the computational workload and difficulty escalated, posing a challenge in meeting real-time requirements.

Furthermore, the Visual Simultaneous Localization And Mapping (VSLAM) algorithm can attain UAV positioning, navigation, map construction, and obstacle avoidance in unknown environments through the perception and analysis of the surrounding environment [16]. Traditional algorithms have been extensively studied and applied, but they often involve multiple parameters and need to be optimized for specific environments.

B. Deep learning-based obstacle avoidance methods

The CNN presents a promising solution for the visual navigation challenges in UAV obstacle avoidance algorithms embedded in DL. Moreover, CNN has made significant advances in the field of image processing, delivering superior feature extraction results compared to the manual dimension reduction. It can also enhance the sensing and obstacle avoidance capabilities of UAVs. For instance, pre-training the YOLO network and integrating it into the Advantage Actor Critic (A2C) model enabled successful training of the underwater vehicles for gate navigation [17]. Furthermore, UAVs were trained using decision data, collected from manually navigating through forests, resulting in a successful forest navigation [5]. Similarly, the collection of urban road data for drones' train yields in successful avoidance of common city obstacles [18].

However, within the domain of UAV obstacle avoidance methods grounded in DL, certain challenges persist. These encompass the need to manually collect and label extensive datasets, coupled with the fact that DL models are typically large and require extended inference times. This, in turn, poses a potential compromise to the performance of UAV obstacle avoidance.

C. Reinforcement learning-based obstacle avoidance methods

Reinforcement learning embraces the "trial and error" mechanism, emphasizing interactive learning with the environment to derive optimal decisions guided by environmental feedback. This approach holds great promise in the realm of UAV obstacle avoidance. Reinforcement learning can be classified into value function-based and policy-based approaches. As an illustration of a value function-based reinforcement learning algorithm, the Q-learning algorithm has proven its utility in various applications, including mobile robot navigation [19].

Moreover, DeepMind, introduced the Deep Q-Network (DQN), an innovative DRL algorithm that combines DL with reinforcement learning, to provide a continuous action spaces or continuous state spaces [20]. Subsequently, various scholars have explored the application of reinforcement learning in the UAV domain [21]. Building upon the Double DQN algorithm, researchers have achieved autonomous obstacle avoidance for UAVs for indoor settings [22]. Some scholars have even attempted to apply strategy-based DRL algorithms to UAVs [23].

However, due to the high parameters' complexity and nonlinear characteristics of visual learning processes, extensive data and computational resources are needed during training. For instance, the CNN model requests a substantial number of images and an extended training duration to generate a suitable DRL strategy [24]. Furthermore, policy-based DRL algorithms display lower convergence compared to discrete space DRL algorithms, primarily because of the increased constraints in complex environments [25].

Therefore, this paper employs a DRL approach that uses depth images as input for UAV obstacle avoidance training. The application of a depth image reduces the channel dimension of the image, and the key depth feature information was preserved, therefore facilitating faster convergence in obstacle avoidance algorithms.

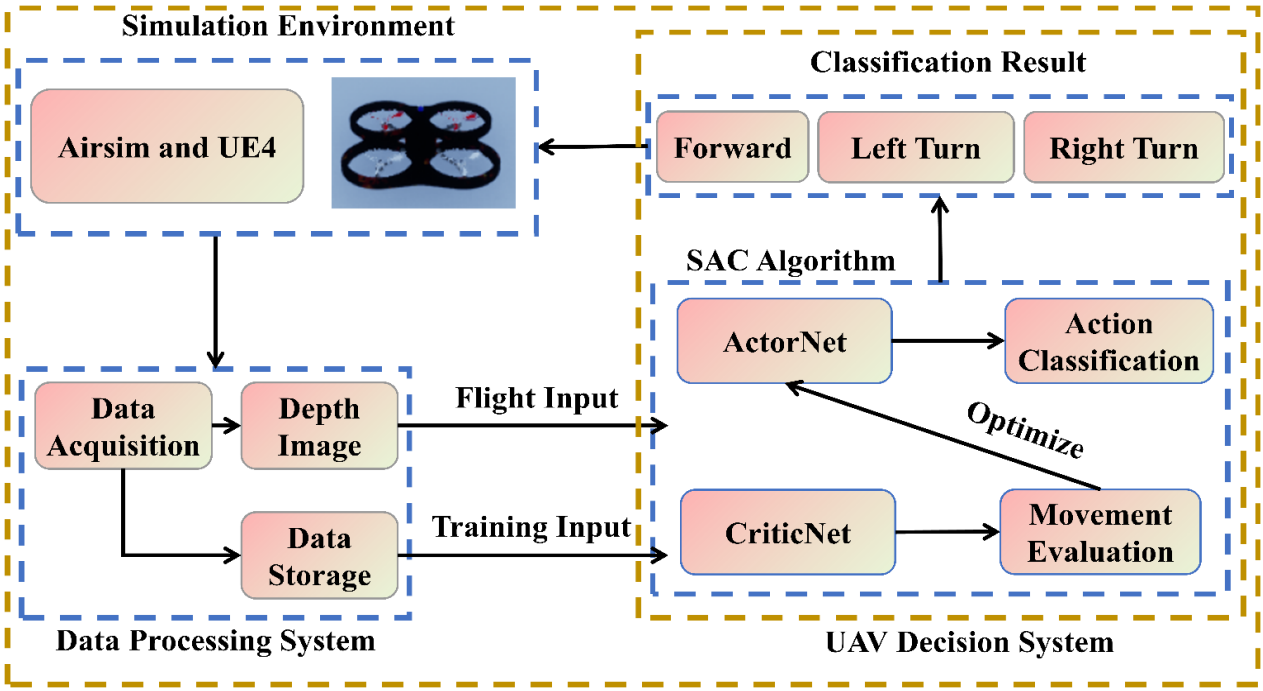

Fig. 1. Structure of the UAV obstacle avoidance decision system

III. Methods

Fig. 1 displays the structure of the UAV obstacle avoidance decision system. The simulation environment was designed using the UE4 game engine and AirSim, an open-source platform provided by Microsoft. This simulator offers a 3D flight and training environment for drones. Data processing system consists of generating depth images in real-time using AirSim. Moreover, the experience buffer pool was employed to store the data. The UAV decision-making system was built upon the SAC algorithm. Initially, when not trained, the UAV's flight actions (*e.g.*, forward, turn left, and turn right) are determined through the *actor* network based on the UAV's current state. Subsequently, during training, the decision-making system was mostly trained through the extraction of data from the experience buffer pool. The *actor* network optimizes the evaluation of its output actions using the *critic* network to achieve improved flight action classification. Finally, the UAV performs these actions in the simulation environment, resulting in acquiring new flight data.

Therefore, the remaining part of this section elaborates the methods used to implement the DAC-SAC algorithm, encompassing the simulation environment, the SAC algorithm framework, the improved Actor-Critic network structure, the self-attention mechanism, the dual experience buffer pool, the delayed learning time, and, finally, the reward function.

A. Simulation Environment

AirSim was an open-source drone and driverless car simulator, developed by Microsoft [26]. It supports Unity 3D and UE4 software, and integrates various sensors, such as vision sensors, to capture high-resolution real-time scenarios.

Through the proposed experiment, firstly, a rectangular closed corridor environment was constructed using the powerful physics engine and rendering technology in UE4.This environment served as the simulation setting for UAV obstacle avoidance training and testing. Additionally, in AirSim, depth images can be directly generated using integrated vision sensors and they serve as inputs for the DRL network.

B. Soft-Actor-Critic algorithm framework

In standard reinforcement learning algorithms, the main objective for the agent consists of attaining the maximum cumulative reward and acquiring an optimal strategy to explore the environment.

To start, at time t, the agent selects a corresponding action A_t based on its current state S_t and strategy Y. Upon execution, the agent involves with the environment, receiving a corresponding reward $R(S_t|A_t)$ based on the received feedback from the environment. Furthermore, the action value function Q_Y was employed to estimate the expected sum of rewards associated with considering a specific action with respect to a given state. Subsequently, the expectation function $E(S_t|A_t)$ was applied to compute the expectation of the next state, while considering the current action and state. Finally, by accumulating the maximum reward and strategy Y, the optimal strategy Y^* was computed. It can be expressed as follows:

$$Y^* = argmax_Y E(S_t|A_t) \sim Q_Y\left[\sum R(S_t|A_t)\right]. \quad (1)$$

To address the challenge of continuous space problems, three algorithms, namely Deep Deterministic Policy Gradient (DDPG), Proximal Policy Optimization (PPO), and SAC, were proposed [27]-[29]. Among them, the SAC algorithm combines the Actor-Critic approach with maximum entropy reinforcement learning, achieving an improved convergence and stability.

Through policy optimization, an entropy regularization term was incorporated into the SAC algorithm to maintain policy randomness and promote the UAV for space exploration. In addition, it maximizes the accumulation of

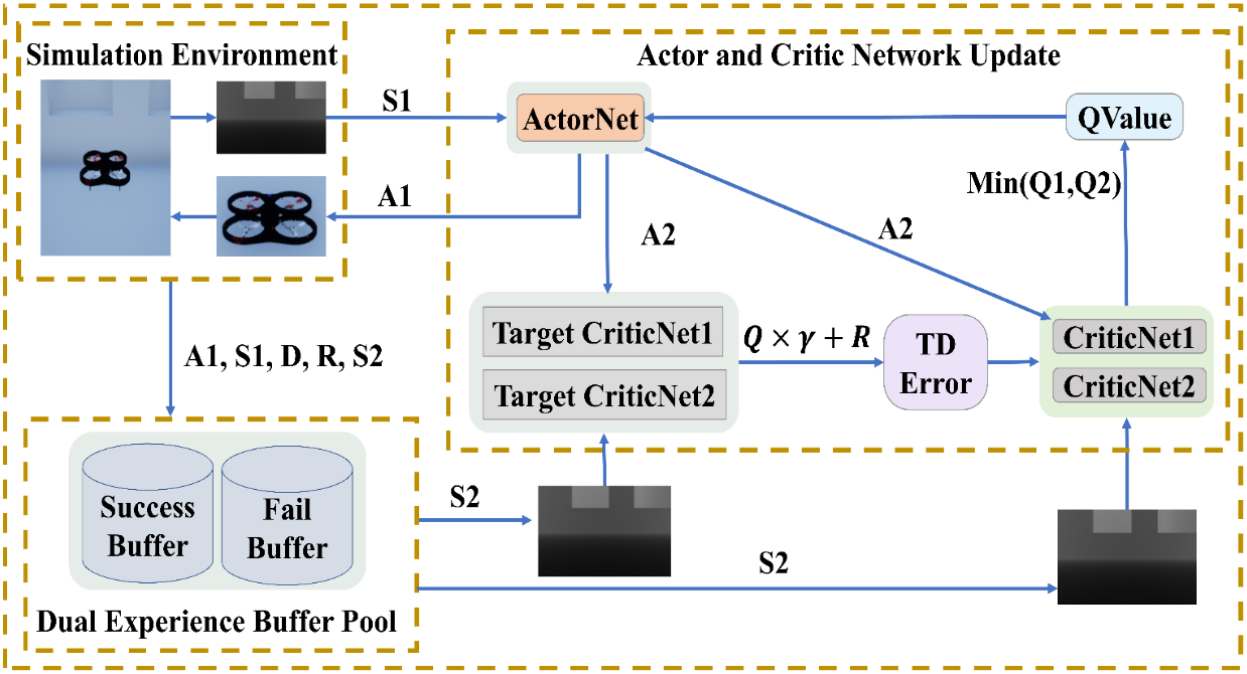

Fig. 2. DAC-SAC algorithm update process

reward expectations throughout the learning process, ultimately yielding to the optimal strategy Y^* described as follows:

$$Y^* = argmax_Y E_{(S_t|A_t)\sim D}\left[\sum_{t=0}^{\infty}\gamma^t\left(R_{(S_t|A_t)} + \alpha H\left(Y(\cdot|S_t)\right)\right)\right]. \quad (2)$$

where α represents the temperature factor and serves as the weight for the entropy term $H\left(Y(\cdot|S_t)\right)$. Considering the dynamic nature of the immediate reward $R_{(S_t|A_t)}$, the use of a fixed α value may introduce training instability. Therefore, automatic adjustment was required. Moreover, when α was small, the policy tends to consider the exploration of different actions, while a larger α value favorably disposes the policy toward performing known optimal actions. In addition, the entropy $H\left(Y(\cdot|S_t)\right)$ was defined as follows:

$$H\left(Y(\cdot|S_t)\right) = -\int_x Y(x|S_t)\log\pi(x|S_t)\,dx. \quad (3)$$

where $\log\pi(x|S_t)$ indicates the logarithm of the conditional probability that the random variable π considers for the x value under specific state S_t, and $Y(x|S_t)$ was the conditional probability distribution where the random variable Y considers the value x under the given state S_t .

The automatic adjustment of α was approximated using an iterative gradient descent method. This process optimizes α based on the logarithm of the reference entropy $\overline{H}$ and the conditional probability distribution $Y(x|S_t)$. The optimization formula was expressed as follows:

$$F(\alpha) = E_{A_t}\left[-\alpha\log Y(A_t|S_t) - \alpha\overline{H}\right]. \quad (4)$$

In more detail, the update process of the DAC-SAC algorithm was illustrated in Fig. 2 (A1 is action, A2 is next action, S1 is state, S2 is next state, R is reward, D is done) where the UAV stores the data including state *S*, action *A*, reward *R*, next time state *SS*, and end sign '*done'* generated using the interaction process in a buffer. The target-critic network gets UAV state and action feedback from the *actor* network, computes the *TDError* value for the current strategy, and optimizes the *critic* network. As for the *actor* network, the *critic* network evaluates the *Q* value of the action taken under the current policy, considering the current state as well as the action fed back by the actor network. This information was used to calculate losses, update *actor* networks, and optimize obstacle avoidance strategies.

C. *Improved Actor Critic network structure*

The DRL algorithm, with its actor-critic structure, can select any action residing inside the continuous action space. However, DRL algorithms typically employ lightweight network structures, making them unable to execute complex visual information. To enable UAVs autonomous obstacle avoidance using visual information as input, it was necessary to optimize the lightweight structure of the actor-critic network. The optimization approach involves a CNN to implement the Gaussian strategy for both the *actor* and *critic* networks. The formal network structure comprises four convolutional layers, two convolutional self-attention layers, two pooling layers, one fully connected layer, and, finally, two parallel fully connected layers. As for the latter network structure, it consists of four convolutional layers, two convolutional attention layers, two pooling layers, and two fully connected layers. The modified structure for both the *actor* and *critic* networks was displayed in Fig. 3

In the network, Firstly, depth images were fed into the actor network, and image features were extracted through convolution and pooling. Secondly, Flatten layer was used to compress the output tensor into one dimensional tensor, and the data was fed into the fully connected layer. After passing through this layer, the output data was sent to the parallel full connection layer. Then, to calculate the mean and logarithmic probability of the action. therefore, it uses reparameterization

to compute the action probability from the mean and logarithmic probabilities of the action. Finally, input the state and action outputs from the actor into the *critic* network. It processes data through the same feature extraction method and evaluates the actions output using the current actor policy.

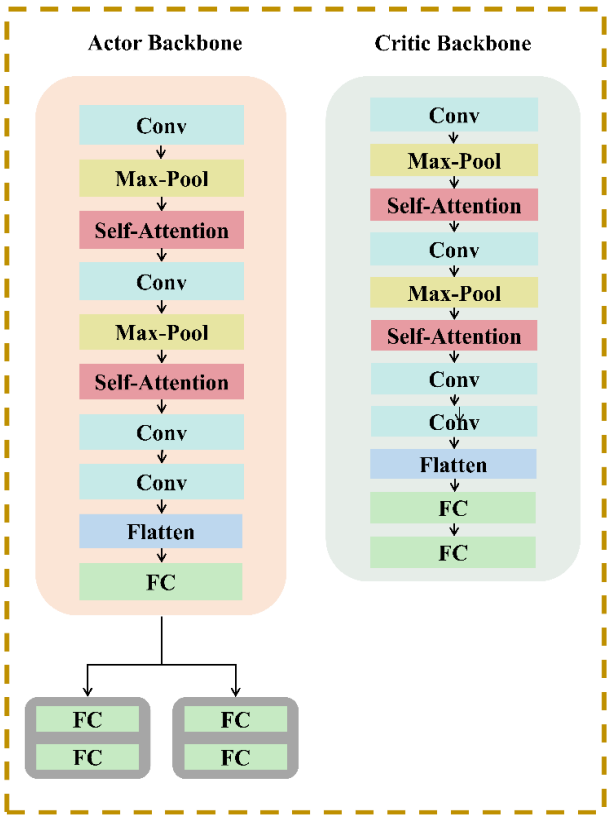

Fig. 3. Modified structure of Actor and Critic networks

D. *Self-Attention*

Self-attention mechanism was a technique used to increase the expressive capabilities of NNs. The CNN was used to implement the convolutional self-attention module. Initially, this module was used to extract significant information from the input tensor. Consequently, the extracted information was weighted and combined with the original input, thereby improving the expressive power of the initial input. The structure of the self-attention technique was proposed in Fig. 4.

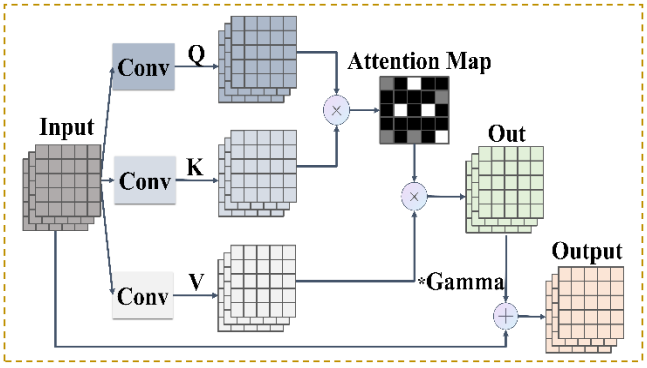

Fig. 4. Self-Attention structure

First, three convolutional layers convolve the input feature graph X (Input) to obtain the query vector Q, the key vector K, and the value vector V. The attention matrix was then derived based on the multiplication of the query vector Q and key vector K, followed by the generation of the normalized attention matrix A (Attention Map). Subsequently, the value vector V was multiplied by the attention matrix to yield the weighted attention matrix O (Out). It was then reshaped to match the format of the input feature graph X. Finally, the output feature map Y (Output) was acquired by adjusting the ratio between the weighted attention matrix and the input feature map using a learnable scaling factor gamma. as represented here below:

$$Y = \gamma * O + X. \quad (5)$$

Within the Actor-Critic network, the inclusion of a convolutional attention layer enables adaptive adjustment of attention weights in response to different input image features. This addresses the focus area challenge, enhancing the model's learning and generalization capabilities.

E. *Dual experience buffer pool*

In the process of training UAVs for autonomous obstacle avoidance, data collection becomes necessary for model updates. Consequently, experience buffer pools are formulated to store relevant data. Throughout the training process, especially in its early stages, the amount of collision data often outweighs instances of successful obstacle avoidance. Storing all this data in a single Experience Buffer Pool (EBP) can hinder the UAV's ability to learn successful obstacle avoidance strategies from failed experiences. In response, a Dual Experience Buffer Pool (DEBP) was introduced to manage the interaction between the UAV and its environment. This DEBP consist of two components: 1) an EBP for successful obstacle avoidance, and 2) an EBP of unsuccessful obstacle avoidance. Thus, the DEBP assists the model in obtaining a substantial amount of successful data during the updates, thereby enhancing training and empowering the UAV to learn optimal obstacle avoidance strategies.

F. *Delayed learning*

Delayed learning was an effective strategy to enhance the training efficiency of reinforcement learning algorithms. During the typical training process of the Actor-Critic network structure, real-time update strategies are frequently applied. However, when considering autonomous obstacle avoidance for UAVs, real-time updates may result in frequent alterations regarding the obstacle avoidance strategy. Therefore, the algorithm stability can be reduced, the learning efficiency may diminish, and the jitter of the obstacle avoidance strategy may increase.

Consequently, in the SAC algorithm's training process, a delayed learning strategy was introduced to postpone network updates until reaching the conclusion of each training round. This method was designed to minimize instability in the training process and amplify learning efficiency. The delayed learning strategy guarantees consistency in the UAV's obstacle avoidance strategy approach, promoting stability in its interactions with the environment. This, in turn, reduces the estimation error in Q values attributed toward uncertainty. Meanwhile, accumulated experiences are batch-updated to the model's parameters, enhancing training stability and convergence speed. This strategy not only alleviates the oscillations often

encountered in the non-stationary gradient descent, but also helps the model to process and utilize the long sequence information more effectively. This, thereby, enables better discovery of hidden structural features within the state space.

G. Reward Functions

The reward function comprises a conditional reward function and a flight distance reward function. In accordance with the requirements of the reinforcement learning algorithm, specific types and values for conditional reward are established, as detailed in Table I.

TABLE I.
CONDITIONAL REWARD TYPES AND CONDITIONAL REWARD VALUES

Award Type	Award Values
Collision	-30
Arrival at Destination	+200
Flight Height Overrun	-20
Maximum Number of Steps	-20

The conditional reward function R_X can be expressed as follows:

$$R_X = \begin{cases} R_C, & \text{Collision} \\ R_T, & \text{Arrival at destination} \\ R_S, & \text{Maximum number of steps} \\ R_B, & \text{Flight height overrun} \end{cases} \quad (6)$$

When calculating the reward function for flight distance, the initial determination of the flight direction was based on the UAV's yaw. Subsequently, the distance reward, denoted as D_r, was obtained through the Euclidean distance between the UAV and the target point. The flight distance reward, denoted as R_D, was then derived by calculating the sum of weighted D_r and the *cos(yaw)* as follows:

$$R_D = \sum(-D_r * 0.01 + \cos(yaw)). \quad (7)$$

During the training process, the conclusion of each round was determined by assessing the collisions and the number of taken steps. The final reward function R was defined as follows:

$$R = R_X + R_D. \quad (8)$$

IV. Experiment

A. Environment Setup

The simulation environment information and the experimental equipment setup are displayed in Table II.

TABLE II.
EXPERIMENTAL EQUIPMENT SETUP AND SIMULATION ENVIRONMENT INFORMATION

Hardware/Frameworks	Parameters and Versions
CPU	Intel Core i7-11700
GPU	NVIDIA RTX 3060TI
RAM	80 GB
Operating System	Windows 10
Program Language	Python 3.9
ML Library	Pytorch 1.12.1
Simulator	Airsim 1.8.1
Game Engine	Unreal Engine 4.27.2

In the experiments, UE4 was used to design two simulation environments, identified as Environment 1 and Environment 2. These environments are identical as they possess dimensions of 55 meters in length, 10 meters in width, and 5 meters in height. Moreover, they are specifically designed for UAV obstacle avoidance training.

In Environment 1, cylindrical obstacles, numbered from 1 to 4, are placed at respective intervals of 8 meters and 2 meters along the length and width, whereas cylindrical obstacles, numbered from 5 to 7 and from 12 to 21, are spaced at intervals of 6 meters and 2 meters regarding the length and width, respectively. moreover, obstacles numbered from 8 to 11 are placed at intervals of 6 meters along the length 1.5 meters along the width. As for the diameter of the cylindrical obstacle, it was set to 1 meter. Fig. 5 presents a top view of Environment 1, and Environment 1 was shown in Fig. 6.

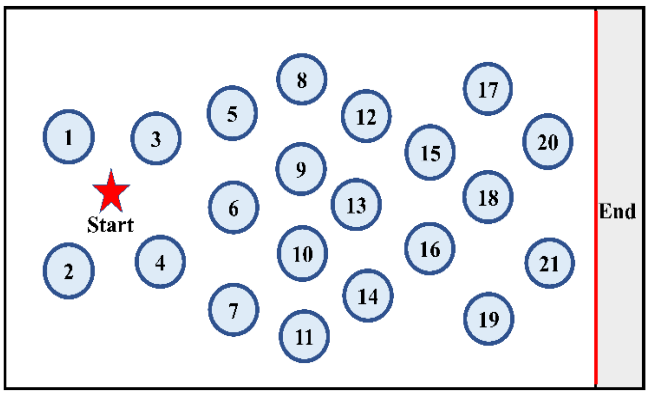

Fig. 5. Top view of Environment 1

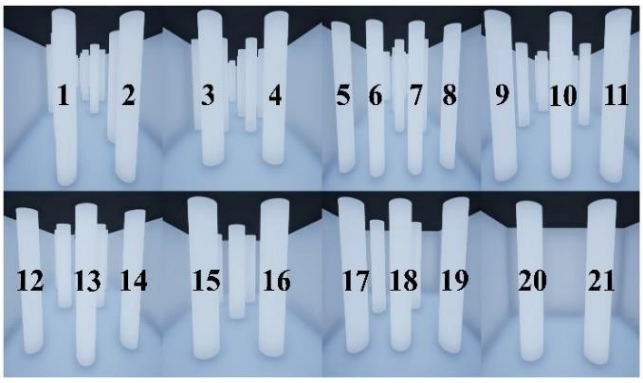

Fig. 6. Environment 1

Environment 2, on the other hand, features wall-type cylindrical obstacles, numbered from 1 to 5, with each obstacle positioned 9 meters apart. As for the wall, it has a length of 10 meters and a height of 5 meters. Moreover, the irregular area enclosed within the walls represents the UAV's traversable area. A top view of Environment 2 was highlighted in Fig. 7, while the complete details of this environment are featured in Fig. 8.

The "Start" mark represents the take-off position of the UAV in the environment, and the "End" area represents the safe landing area of the UAV.

The security zone's dimensions are set to a length of 10 meters and a width of 1 meter. A successful obstacle avoidance was completed when the UAV safely reaches the safety zone.

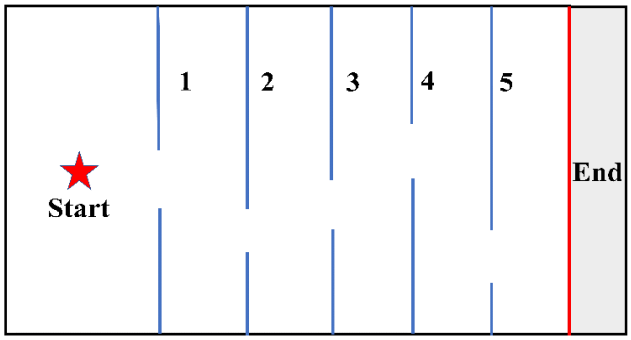

Fig. 7. Top view of Environment 2

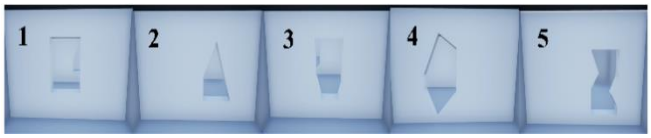

Fig. 8. Environment 2

B. Dual Experience Buffer Pool and Self-Attention Setup

To validate the effectiveness of the dual experience buffer pool and the self-attention network in enhancing the UAV obstacle avoidance performance, the proposed experimental setup was divided into two parts: the evaluation of the effectiveness of the DEBP and the assessment of the effectiveness of the self-attention network.

In more detail, in the first part, Environment 1 was used as the validation environment for this method. Consequently, the cumulative reward size of 1000 training rounds, during the training process, was compared between the use of DEBP and EBP.

As for the second part, the DEBP was initially used for training, and the network was trained for 1000 rounds with and without the application of the self-attention layer. The cumulative reward size of the 1000 training rounds was then evaluated.

C. Delayed learning Setup

To verify whether delay learning strategy was effective for the stability of UAV obstacle avoidance strategy, environment 1 was firstly adopted as the verification environment of the method. Secondly, the delayed learning strategy was applied to train the UAV for 500 rounds using the DEBP and the self-attention network; moreover, the network was updated after each round. Finally, the UAV was trained for 500 rounds using DEBP and self-attention network instead of the delayed learning strategy, and the network will be updated after each round. The effectiveness of the delayed learning strategy was judged based on the cumulative reward size through the UAV training.

D. Training Setup

The training process consists of two stages: the first stage consists of data acquisition, where the UAV takes random actions to gather sufficient state and action data. This data was then stored in a buffer pool. The second stage consists of training where the UAV initially refrains from taking random actions; it instead inputs the state into the policy network to obtain the action output. Subsequently, the UAV's state and the action data, produced by the policy network, are saved in the buffer pool. After each training round, data within the buffer pool was extracted to update the policy network, and the updated strategy network was employed for the next round of training. Based on the reward setup, the drone will receive a greater reward for collision-free and longer flights. These rewards will be calculated after each turn.

E. Testing Setup

During testing, the UAV was launched from the starting position (start). A successful obstacle avoidance was considered if the UAV reaches the safe area; therefore, the count of successes was tallied.

To evaluate the obstacle avoidance ability of the trained UAV, obstacle avoidance tests are conducted in both Environment 1 and Environment 2 (with obstacle sequence 1, 2, 3, 4, and 5 – 12345 for simplicity) for 100, 200, and 500 rounds. Subsequently, the obstacles' order in Environment 2 was adjusted to 45231, 23145, 32541, and 53412, while keeping the same number of rounds for obstacle avoidance tests. The top view of obstacle sequences 45231, 23145, 32541, and 53412 was shown in Figs. 9-12, where the top view of the transposed obstacle sequence was displayed in Fig. 13. Lastly, a new environment, referred to Environment 3, was created by combining different obstacles from Environment 1 and Environment 2. In addition, obstacle avoidance rate tests are conducted for 100, 200, and 500 rounds, as illustrated in Fig. 14.

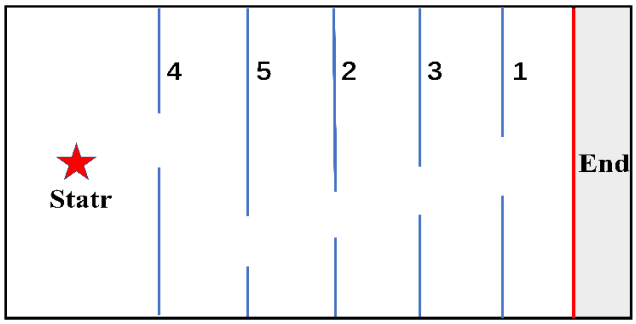

Fig. 9. Top view with obstacle sequence 45231

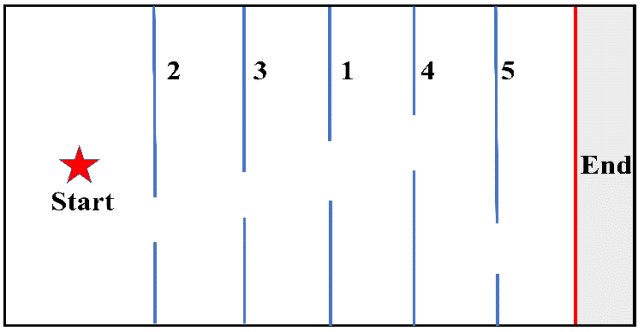

Fig. 10. Top view with obstacle sequence 23145

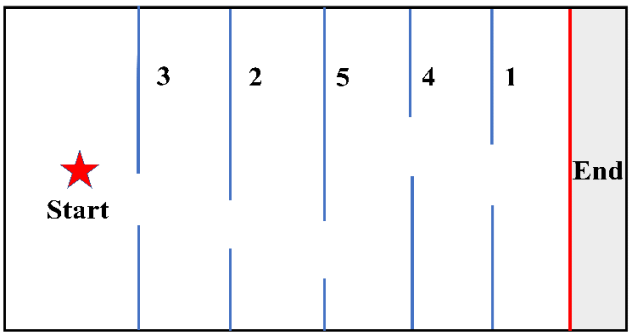

Fig. 11. Top view with obstacle sequence 32541

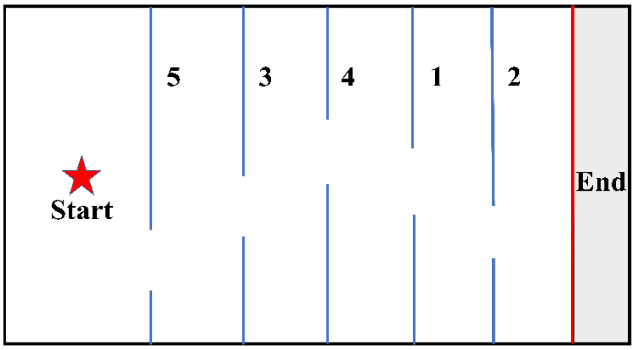

Fig. 12. Top view with obstacle sequence 53412

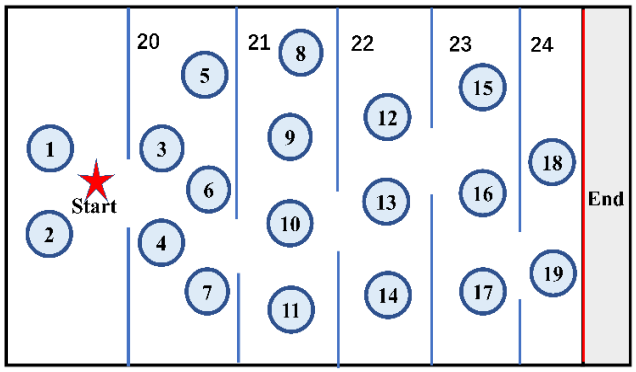

Fig. 13. Environment 3 Top view

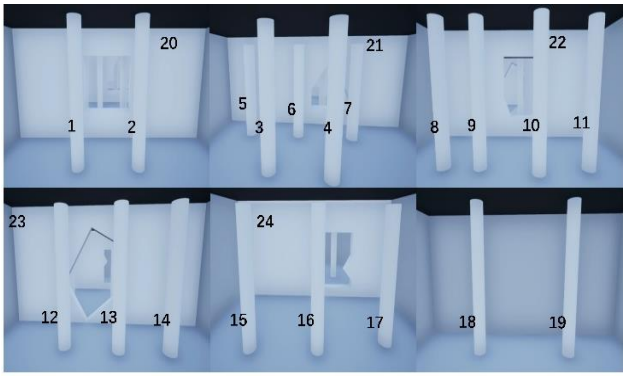

Fig. 14. Environment 3

F. Dual Experience Buffer Pool and Self-Attention Results

The cumulative reward for 1000 rounds of training, using the EBP and DEBP, was shown in Fig. 15. Additionally, the cumulative reward for 1000 rounds of training with and without the self-attention layer was illustrated in Fig. 16.

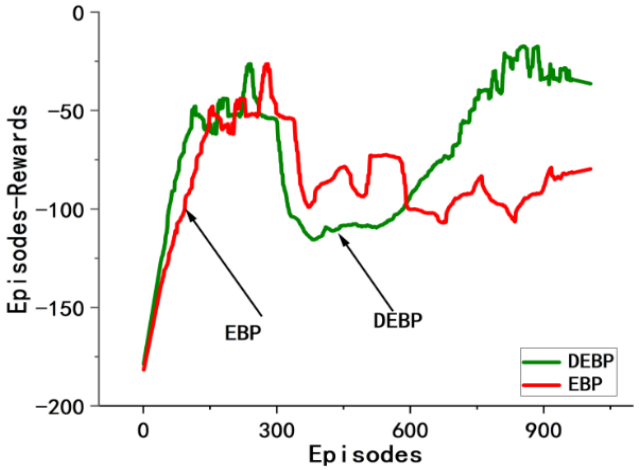

Fig. 15. Cumulative rewards for DEBP and EBP

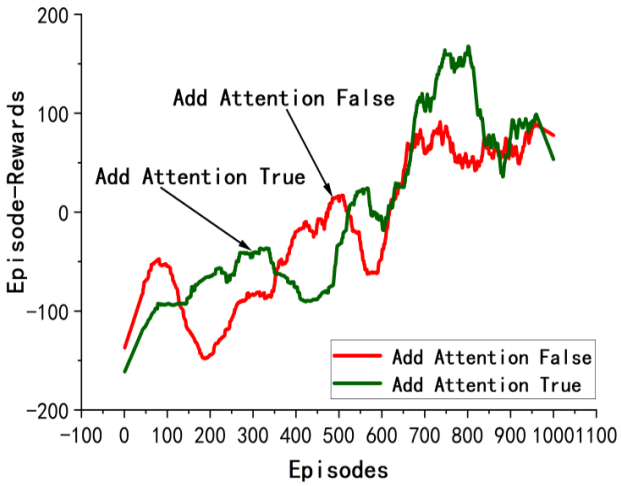

Fig. 16. Rewards with and without added attention

By comparing the size of the cumulative reward values, the cumulative reward in DEBP was higher than in EBP for roughly 60% of the cases. Furthermore, the cumulative reward attained with the self-attention layer network surpasses that achieved without the use of the self-attention layer in approximately 60% of the cases. These findings validate the efficiency of DEBP and the self-attention network in improving UAV obstacle avoidance performance.

G. Delay learning Result

The cumulative reward for 500 rounds of training, with and without delayed learning strategies, was displayed in Fig. 17. (DL is Delay learning)

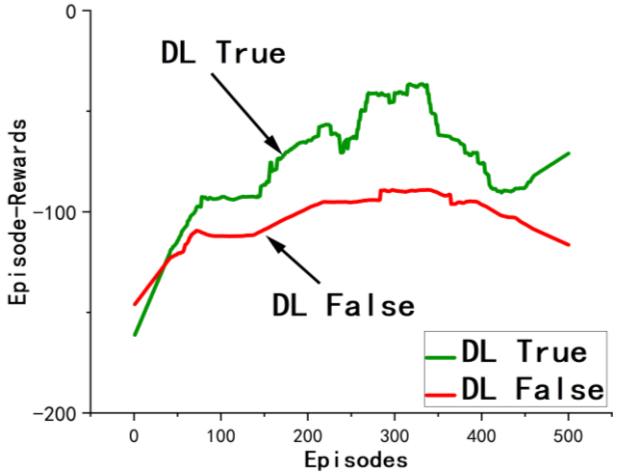

Fig. 17. Training accumulation reward

By comparing the size of the cumulative reward value, the implementation of the delayed learning strategy yields in better results. These results demonstrate the effectiveness of delayed learning strategy regarding the stability of UAV obstacle avoidance strategy, and enhance the obstacle avoidance ability of UAV.

H. Training Results

The UAV was trained in both Environment 1 and Environment 2 (using the obstacle sequence 12345), where the network was updated after each episode. Following the findings of each episode, the cumulative reward (refer to Fig. 18) was recorded. In Fig. 18, the horizontal axis represents the number of training rounds whereas the vertical axis denotes the cumulative reward value for each round (it was

worth noting that the cumulative reward shown in the figure was the smoothed data).

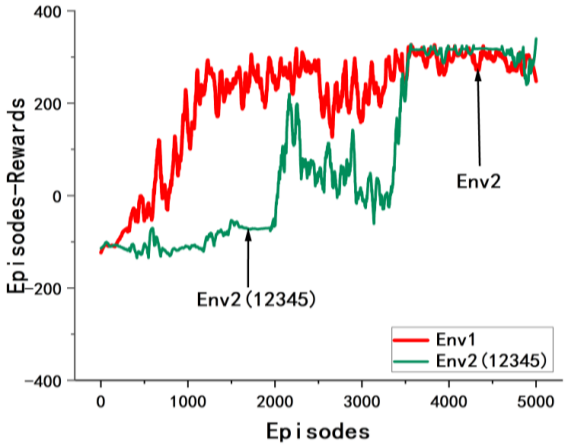

Fig. 18. Training accumulation reward

The comparison of the cumulative reward values during the training phase reveals a growing trend as the number of training rounds progresses. Data growth stabilizes after approximately 3500 training rounds, suggesting that the UAV's policy selection has achieved stability and the DAC-SAC algorithm has reached a state of convergence. Referring to the cumulative reward values, it was obvious that the trained DAC-SAC algorithm empowers the UAV to autonomously navigate diverse known environments while avoiding obstacles.

I. Testing Results

The success rate of obstacle avoidance, denoted as δ_i, was defined as the ratio of the number of successes (S) to the total number of test rounds (T) for a given test iteration i:

$$\delta_i = S \div T. \tag{9}$$

After the training phase, the model exhibiting the most effective obstacle avoidance performance was selected for testing. Therefore, the best-performing obstacle avoidance model in Environment 1 was chosen, and obstacle avoidance tests, consisting of 100, 200, and 500 rounds, are conducted within that environment. The model having the best obstacle avoidance performance in Environment 2 (12345) was also selected, and tests are conducted for the same number of rounds. The results of these tests are presented in Table III.

TABLE III.
OBSTACLE AVOIDANCE TEST RESULTS FOR 100, 200, AND 500 ROUNDS IN ENVIRONMENT 1 AND ENVIRONMENT 2 (12345)

Test rounds	Environment 1	Environment 2 (12345)
500	0.992	0.992
200	0.995	0.995

Subsequently, the model with the best obstacle avoidance performance in Environment 2 (12345) was selected. The obstacle order in this environment then changes to 53412, 45231, 23145, and 34125; however, the UAV's obstacle avoidance was still tested for 100, 200, and 500 rounds. The test results are presented in Table IV. The average obstacle avoidance success rate was calculated using the below equation:

$$\bar{\delta} = (\delta_{100} + \delta_{200} + \delta_{500}) \div 3. \tag{10}$$

Finally, the model demonstrating the best obstacle avoidance capabilities after training in Environments 1 and 2 (12345) was deployed. Obstacle avoidance tests are then conducted in Environment 3 for 100, 200, and 500 rounds. The results of these tests are displayed in Table V.

According to the experimental results in Table III, the DAC-SAC algorithm in enabling autonomous UAV obstacle avoidance in known environments, the success rate reached more than 99.5%. This underscores the algorithm's high stability and reliability in handling obstacle avoidance tasks within familiar settings.

According to the experimental results in Table IV, in a similar new environment without retraining, the highest average obstacle avoidance success rate reached 97.43%. This shows that the algorithm was adept at autonomously avoiding obstacles even without retraining.

Notably, according to the experimental results in Table V, the highest obstacle avoidance success rate of 84.8% was

TABLE IV.
AVERAGE OBSTACLE AVOIDANCE SUCCESS RATE IN ENVIRONMENT 2 WITH CHANGED OBSTACLE ORDER

Order of obstacles in environment 2	53412	45231	23145	32541
Success count for 100 rounds	96	84	90	96
Success count for 200 rounds	197	184	186	195
Success count for 500 rounds	489	467	476	484
Average successful obstacle avoidance rate	0.9743	0.898	0.9273	0.9676

TABLE V
RESULT OF OBSTACLE AVOIDANCE SUCCESS RATE FOR 100, 200, AND 500 ROUNDS IN ENVIRONMENT 3

Test Rounds	Environment 3 (Env1-Best-Model)	Environment 3 (Env2-Best-Model)
500	0.770	0.848
200	0.705	0.810
100	0.690	0.790

achieved without the need for re-training in a new environment. This suggests the algorithm's proficiency in autonomous obstacle avoidance, even in unfamiliar environments. Furthermore, the experimental results demonstrate that the proposed algorithm was well-suited for autonomous flight and obstacle avoidance for UAVs, characterized by its fast learning ability across various environments, and exhibiting robust generalization and adaptability to new surroundings.

Compared to SAC algorithm using the VAE model [31], the DAC-SAC algorithm just requires 3500 rounds to achieve relatively optimal results. This significantly alleviates the slow convergence issue commonly observed in DRL algorithms.

For a direct comparison of superiority, the DAC-SAC and the TD3 algorithms are trained for 3000 rounds, and the cumulative rewards are compared, as shown in Fig. 19. The results clearly show that the DAC-SAC algorithm surpasses the TD3 algorithm in both the growth rate and the peak value regading the reward value.

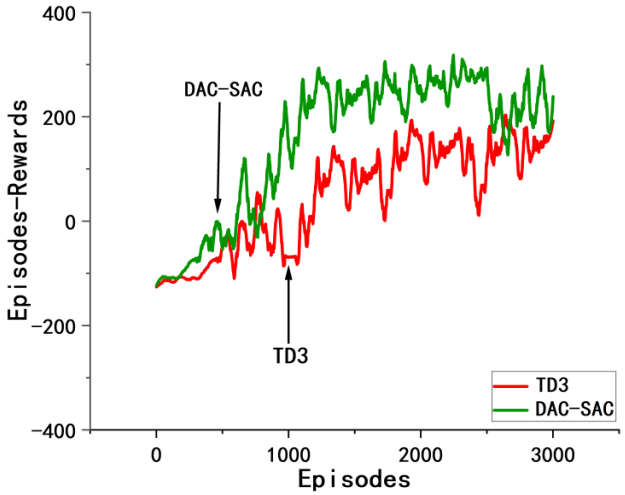

Fig.19. Accumulated reward values for TD3 and DAC-SAC

V. Conclusion

Concerning the end-to-end autonomous UAV obstacle avoidance problem, a deep reinforcement learning-based autonomous flight obstacle avoidance method, called DAC-SAC, was proposed. This method consists of an actor and critic network with convolutional self-attention layers and a DEBP. The lightweight network, implemented using this method, can directly process the depth image data. In opposite to other obstacle avoidance algorithms, this approach just uses visual sensors as input to enable autonomous flight obstacle avoidance of UAVs in continuous motion space. Moreover, it was applicable to multiple obstacle types, and has excellent obstacle avoidance ability in unknown environment.

In the realized experiments, the UAV flight actions include forward, turning left, and turning right movements. While testing known environments, the trained DAC-SAC algorithm achieved 99.5% success rate when performing obstacle avoidance. To simulate UAV flight in an unknown environment, the test environment was reconfigured without retraining the UAV; however, the DAC-SAC algorithm still competed the highest obstacle avoidance success rate of 84.8%. Since the UAV was not aware of the obstacles in the new environment, this resulted in a lower obstacle avoidance success rate. Compared to the TD3 algorithm, having a cumulative reward peak value of 247.6, the DAC-SAC algorithm reached a peak value of approximately 319.1. Moreover, the slope of the cumulative reward value for the DAC-SAC algorithm between 0 and 3000 rounds was equal to 0.13182, while the slope for the TD3 algorithm was about 0.10998. Therefore, the DAC-SAC algorithm significantly outperforms the TD3 algorithm in terms of the growth rate of the reward value and the peak value of the cumulative reward. When compared to the results of an 8000-round VAE-SAC model in similar research [31], the DAC-SAC algorithm achieved relatively ideal results in just 3500 rounds. This helped, to some extent, addressing the issue of slow convergence speed of DRL algorithms for UAV obstacle avoidance tasks.

The future works will investigate three objectives: first, enabling the autonomous flight of the UAV in all directions in a 3D space to avoid obstacles, thus improving their avoidance ability in complex environments; second, enhancing the UAV's obstacle avoidance capability in unknown new environments; and, finally, continuing the implement efforts to optimize the actor-critic network structure to enhance the memory and processing capability of the UAV for continuous state information.

References

[1] Shubo, Wang, Chen Jian, and Peng Bingzhong. "Analysis on The Industrial Chain of Agricultural Unmanned Aerial Vehicles in China." Journal of China Agricultural University 23.32 (2018): 131-139.

[2] Outay, Fatma, Hanan Abdullah Mengash, and Muhammad Adnan. "Applications of Unmanned Aerial Vehicle (UAV) in Road Safety, Traffic and Highway Infrastructure Management: Recent Advances and Challenges." Transportation Research Part A: policy and practice 141 (2020): 116-129.

[3] M. Atif, R. Ahmad, W. Ahmad, L. Zhao and J. J. P. C. Rodrigues, "UAV-Assisted Wireless Localization for Search and Rescue," in IEEE Systems Journal, vol. 15, no. 3, pp. 3261-3272, Sept. 2021,

[4] Padhy, R. P., Verma, S., Ahmad, S., Choudhury, S. K., & Sa, P. K. (2018). Deep Neural Network for Autonomous UAV Navigation in Indoor Corridor Environments. Procedia Computer Science, 133, 643-650.

[5] A. Giusti et al., "A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots," in IEEE Robotics and Automation Letters, vol. 1, no. 2, pp. 661-667, July 2016

[6] D. Gandhi, L. Pinto and A. Gupta, "Learning to Fly by Crashing," 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 2017, pp. 3948-3955

[7] Yaoming Li, and Yun Chen, "Enhancing A Stock Timing Strategy by Reinforcement Learning," IAENG International Journal of Computer Science, vol. 48, no.4, pp930-939, 2021

[8] Hejun Xuan, Xuelin Zhao, Jianwei Fan, Yahui Xue, Fangfang Zhu, and Yanling Li, "VNF Service Chain Deployment Algorithm in 5G Communication Based on Reinforcement Learning," IAENG International Journal of Computer Science, vol. 48, no.1, pp1-7, 2021

[9] Azar, A. T., Koubaa, A., Ali Mohamed, N., Ibrahim, H. A., Ibrahim, Z. F., Kazim, M., ... & Hameed, I. A. (2021). Drone Deep Reinforcement Learning: A Review. Electronics 2021, 10, 999.

[10] B.Rubí, B.Morcego and R. Pérez, "A Deep Reinforcement Learning Approach for Path Following on A Quadrotor," 2020 European Control Conference (ECC), St. Petersburg, Russia, 2020, pp. 1092-1098

[11] Li, C., Li, L., Geng, Y., Jiang, H., Cheng, M., Zhang, B., ... & Chu, X. (2023). Yolov6 v3. 0: A Full-scale Reloading. arXiv preprint arXiv:2301.05586.

[12] Dosovitskiy, A, Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X, Unterthiner, T., ... & Houlsby, N. (2020). An Image is Worth 16x16

Words: Transformers for Image Recognition at scale. arXiv preprint arXiv:2010.11929.
[13] Zhou, Y., & Ho, H. W. Online Robot Guidance and Navigation in Non-stationary Environment with Hybrid Hierarchical Reinforcement Learning. Engineering Applications of Artificial Intelligence, 2022, 114, 105152.
[14] Denton, R. V. (1985). Demonstration of an Innovative Technique for Terrain Following/Terain Avoidance, -The Dynapath Algorithm. In NAECON (pp. 522-529).
[15] Bousson, K. (2005, August). Single gridpoint Dynamic Programming for Trajectory Optimization. In AIAA Atmospheric Flight Mechanics Conference and Exhibit (p. 5902).
[16] Y. Park and S. Bae, "Keeping Less is More: Point Sparsification for Visual SLAM," 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 2022, pp. 7936-7943
[17] Zieliński, P., & Markowska-Kaczmar, U. (2021). 3D Robotic Navigation using a Vision-based Deep Reinforcement Learning Model. Applied Soft Computing, 110, 107602.
[18] A. Loquercio, A. I. Maqueda, C. R. del-Blanco and D. Scaramuzza, "DroNet: Learning to Fly by Driving," in IEEE Robotics and Automation Letters, vol. 3, no. 2, pp. 1088-1095, April 2018
[19] Low, E. S., Ong, P., & Cheah, K. C. (2019). Solving the Optimal Path Planning of a Mobile Robot using Improved Q-learning. Robotics and Autonomous Systems, 115, 143-161.
[20] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level Control through Deep Reinforcement Learning. nature, 518(7540), 529-533.
[21] Vamvoudakis, K. G., Vrabie, D., & Lewis, F. L. (2014). Online Adaptive Algorithm for Optimal Control with Integral Reinforcement Learning. International Journal of Robust and Nonlinear Control, 24(17), 2686-2710.
[22] Xie, L., Wang, S., Markham, A., & Trigoni, N. (2017). Towards Monocular Vision based Obstacle Avoidance through Deep Reinforcement Learning. arXiv preprint arXiv:1706.09829.
[23] H. Qie, D. Shi, T. Shen, X. Xu, Y. Li and L. Wang, "Joint Optimization of Multi-UAV Target Assignment and Path Planning Based on Multi-Agent Reinforcement Learning," in IEEE Access, vol. 7, pp. 146264-146272, 2019
[24] M. Savva et al., "Habitat: A Platform for Embodied AI Research," 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea (South), 2019, pp. 9338-9346
[25] Wijmans, E., Kadian, A., Morcos, A., Lee, S., Essa, I., Parikh, D., & Batra, D. (2019). Dd-ppo: Learning Near-perfect Pointgoal Navigators from 2.5 Billion Frames. arXiv preprint arXiv:1911.00357.
[26] Shah, S., Dey, D., Lovett, C., & Kapoor, A. (2018). Airsim: High-fidelity Visual and Physical Simulation for Autonomous Vehicles. In Field and Service Robotics: Results of the 11th International Conference (pp. 621-635). Springer International Publishing.
[27] 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.
[28] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. arXiv preprint arXiv:1707.06347.
[29] Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J. (2018). Soft Actor-Critic Algorithms and Applications. arXiv preprint arXiv:1812.05905.
[30] Fujimoto, S., Hoof, H., & Meger, D. (2018, July). Addressing Function Approximation Error in Actor-Critic Methods. In International Conference on Machine Learning (pp. 1587-1596). PMLR.
[31] Xue, Z., & Gonsalves, T. (2021). Vision Based Drone Obstacle Avoidance by Deep Reinforcement Learning. AI, 2(3), 366-380.