A Hierarchical Reinforcement Learning Algorithm Based on Attention Mechanism for UAV Autonomous Navigation

Zun Liu, Yuanqiang Cao, Jianyong Chen, and Jianqiang Li

***Abstract*— Unmanned Aerial Vehicles (UAVs) are increasingly being used in many challenging and diversified applications. Meanwhile, UAV's ability of autonomous navigation and obstacle avoidance becomes more and more critical. This paper focuses on filling up the gap between deep reinforcement learning (DRL) theory and practical application by involving attention mechanism and hierarchical mechanism to solve some severe problems encountered in the practical application of DRL. More specifically, in order to improve the robustness of DRL, we use averaged estimation function instead of the normal value estimation function. Then, we design a recurrent network and a temporal attention mechanism to improve the performance of the algorithm. Third, we propose a hierarchical framework to improve its performance on long-term tasks. Some realistic simulation environments, as well as the real-world, are used to evaluate the proposed UAV autonomous navigation method. The results demonstrate that our DRL-based navigation method performs well in different environments and outperforms the original DrQ algorithm.**

***Index Terms*— Unmanned aerial vehicles, autonomous navigation, deep reinforcement learning, hierarchical reinforcement learning.**

Manuscript received 18 July 2022; revised 20 October 2022; accepted 21 November 2022. Date of publication 16 December 2022; date of current version 1 November 2023. This work was supported in part by the National Natural Science Foundation of China under Grant U2013201, Grant 62006157, Grant 61836005, Grant 61806130, Grant 62073225, and Grant 62072315; in part by the National Key Research and Development Program of China under Grant 2020YFA0908700; in part by the Natural Science Foundation of Guangdong Province Outstanding Youth Program under Grant 2019B151502018; in part by the Shenzhen Science and Technology Innovation Commission under Grant R2020A045; in part by the Guangdong "Pearl River Talent Recruitment Program" under Grant 2019ZT08X603; in part by the Guangdong "Pearl River Talent Plan" under Grant 2019JC01X235; in part by the Shenzhen Talents Special Project-Guangdong Provincial Innovation and Entrepreneurship Team Supporting Project 2021344612; in part by the Foundation of Key Laboratory of Autonomous Systems and Networked Control (South China University of Technology), Ministry of Education of China, under Grant 2021A03; and in part by the Unmanned Aerial Vehicle Systems Engineering Technology Research Center of Guangdong (South China University of Technology). The Associate Editor for this article was H. Lv. *(Corresponding author: Jianqiang Li.)*

Zun Liu and Jianqiang Li are with the National Engineering Laboratory for Big Data System Computing Technology of China, Shenzhen 518060, China, and also with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China (e-mail: zunliu@szu.edu.cn; lijq@szu.edu.cn).

Yuanqiang Cao and Jianyong Chen are with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China (e-mail: caoyuanqiang2019@email.szu.edu.cn; jychen@szu.edu.cn).
(e-mail: zunliu@szu.edu.cn; caoyuanqiang2019@email.szu.edu.cn; jychen@szu.edu.cn; lijq@szu.edu.cn).

Digital Object Identifier 10.1109/TITS.2022.3225721

I. Introduction

In RECENT years, unmanned aerial vehicles (UAVs) have been extensively used in both civilian and military applications due to their advantages, such as good maneuverability and low cost. Among them, UAVs combined with computer vision have been successively applied in video surveillance [1], intelligent transportation [2], [3], post-disaster search and rescue [4], intelligence collection [5], [6], and other scenes. However, the scenarios mentioned above impose several limitations on regular operational tasks such as takeoff, navigation, target detection, and environmental interaction, especially for obstacle avoidance. It is very essential to improve the ability of autonomous navigation by combining artificial intelligence technology and using information gathered by the limited sensors of the UAV.

Traditional techniques once played important roles in autonomous navigation of UAV, ranging from non-learning-based to learning-based approaches. Perception and avoidance is one of the most prevalent non-learning-based techniques. These techniques avoid collisions and perform navigation by turning the vehicle in the opposite direction. Odelga et al. [7] designed sensors integrating inertial and optical flow distance measurement in which Kalman filter was used to estimate UAV linear velocity. To avoid collisions, they also employed RGB-D cameras to provide operators with visual input, as well as data to construct a probabilistic robot-centric obstacle model. For UAV navigation, Wang et al. [8] proposed a nonlinear signal-correction observer (NSCO) method to estimate the location and flying velocity of the UAV. In order to allow drones to fly independently in locations where wireless localization is not available, Tiemann et al. [9] developed a technique to enhance and fuse ultra-wideband localization with monocular SLAM. A unique all-source navigation filter, dubbed compressed pseudo-SLAM, was proposed by Kim et al. [10] for UAV navigation. It can smoothly combine all available information in a computationally efficient way.

In order to reduce the dependence of the obstacle avoidance algorithm on the environment and strengthen the applicability, many researchers have started to combine reinforcement learning (RL) to deal with the autonomous navigation problem. Some works have been done and have achieved good results. Imanberdiyev et al. [11] proposed TEXPLORE, a model-based DRL algorithm for drone navigation in unobstructed grid maps, as an enhanced control approach. Wang et al. [12]

developed a method called DRL with nonexpert helper (LwH). This method used a markov decision process (MDP) with sparse rewards to frame the challenge of autonomous UAV navigation in large-scale complex environments. He et al. [13] proposed an RL-based approach to solve the UAV autonomous navigation problem. This method was combined with a bio-inspired monocular vision perception method. It was more computationally efficient than both SLAM based and optical flow based navigation methods. Tong et al. [14] proposed a distributed DRL architecture. This architecture divided the UAV navigation problem into two sub-tasks. Each sub-task interacted with data via a DRL network which was based on long short term memory (LSTM) technology, and a loss function was developed to integrate the two sub-solutions. To perform successful navigation in dynamic and multi-obstacle environments, Zhang et al. [15] developed a DRL-based method and got good performance. Furthermore, Xin et al. [16] presented an empirical playback-based DRL method to perform autonomous navigation. [3] proposed a novel air-assisted vehicular caching scheme based on deep Q learning to respond to the driving safety-related requests from vehicle users. [17] designed a navigation policy for UAV to improve the data freshness and connectivity with the Internet of Things (IoT) devices. [2] used recurrent neural networks with temporal attention to solve UAVs' navigation problem. It provided better results comparing to prior works, in terms of distance covered without collisions. In order to achieve UAV autonomy, Chansuparp et al. [18] used the TD3 algorithm with simplified point cloud data and an augmentative backward reward function.

Hierarchical reinforcement learning (HRL) is an important computational method aiming at large-scale problems. It can perform at various degrees of temporal abstraction and greatly reduce the dimensionality and the difficulty of training. Rafati et al. [19] presented an efficient and general approach to sub-target discovery. This approach is based on a model-free HRL framework. With an unsupervised learning approach, it can automatically learn sub-goals to solve large-scale RL problems. Florensa et al. [20] used stochastic neural networks (SNNs), a type of neural network composed of stochastic units in the computation graph, to provide a broad framework. The framework can train policies for a series of tasks with sparse rewards.

For reinforcement learning in real environment, one of the main challenges for reinforcement learning's applications in real environments is the "dimension curse" caused by too many action-states pairs, which makes it difficult to converge or learn effective strategies. For example, in the autonomous obstacle avoidance of UAV, direct input of photos and sensor information will lead the algorithm to the lack of generality and robustness, which obstructs application of the algorithm in practice.

Previous UAV autonomous navigation research frequently encountered issues such as unstable training, slow convergence, and an excessively large state space. To address these issues, we introduce attention and hierarchical mechanisms, and propose a DRL framework for UAV autonomous navigation and obstacle avoidance. The navigation and obstacle avoidance issues are described as an MDP and are addressed by a novel online DRL algorithm. The focus of this work is to fill the gap between virtual and real environment in reinforcement learning. The design idea of our algorithm and the randomness of the training process ensure that the input of the model is as rich as possible. It can cover the depth maps acquired by the UAV in the real flight process. It can also make the selection of actions and sub-strategies, so as to ensure the effectiveness of the UAV in the real environment. The proposed algorithm is based on DrQ [21] and includes an estimation function of average value, temporal attention mechanism, hierarchical framework and recurrence. The main contributions of the paper are:

- We propose Hierarchical Temporal Attention Recurrent Averaged DrQ (HTARADrQ). HTARADrQ can learn to operate over different levels of temporal abstraction. It can greatly reduce the dimension of state-action pairs and the difficulty of training.
- The proposed Temporal Attention Recurrent Averaged DrQ (TARADrQ) includes the estimate function, temporal attention mechanism and recurrence. It demonstrates more stable in training. It also demonstrates more capable in handling longer input sequences and in exploring temporal dependencies.
- The feasibility and efficiency of the algorithm are observed by both simulations and real-world testing.

The remainder of this paper is organized as follows: Section II introduces the related work of learning based UAV navigation. Our proposed algorithm is introduced in Section III. Benchmark and analysis are carried out in Section IV. Section V shows the result of our proposed HTARADrQ algorithm tested in the real-world. Finally, the conclusion is presented in Section VI.

II. Related Work

A. Learning Based UAV Navigation

Obstacle detection and avoidance tasks benefit greatly from autonomous navigation. In the literature, monocular obstacle detection approaches are either based on scene detection utilizing conventional machine learning or computer vision utilizing deep learning. Smolyansky et al. [22] developed a Micro Air Vehicle (MAV) system to automatically follow pathways in unstructured outdoor situations like woods. The system employs a deep neural network (DNN) called TrailNet. It can estimate the direction of the vision and the MAV's lateral offset relative to the trajectory's center. Korris et al. [23] provided a CNN-based technique for self-supervised indoor drone navigation. This approach tackles real-time obstacle avoidance via the use of a regression CNN. The data come from an onboard monocular camera. Loquercio et al. [24] developed a CNN-based network called DroNet that can safely fly drones via city streets. DroNet is an eight-layer residual network and has two outputs. One output is a steering angle that allows the drone to retain navigation while avoiding obstacles. Another output is a collision possibility that allows the drone to recognize dangerous circumstances and respond swiftly. Kaufmann et al. [25] investigated the issues of autonomous drone racing in some dynamic situations. They also presented a method that blends control systems and CNNs with cutting-edge path planning. This approach runs totally on

board and does not require any explicit environment mapping. Lee et al. [26] provided a new approach to autonomously navigating tiny drones in planted woods by a single camera. Because monocular vision lacks depth information, they proposed a deep learning model called Faster Region-based Convolutional Neural Network (Faster R-CNN) to recognize tree trunks.

Recently, researchers tried to use deep learning algorithms to find a stable control method for UAVs. Hii et al. [27] proposed a DRL-based system for drone delivery optimization. According to the study, drone delivery is the path that a drone can reach a location while avoiding various impediments, also it employs multiple DRL algorithms to help UAVs achieve their goals. Shin et al. [28] presented experimental investigations on drones using various reinforcement learning algorithms, such as unsupervised learning, supervised learning, and reinforcement learning. Hodge et al. [29] developed a generic navigation algorithm that guides the drone to the issue site by data coming from the drone's on-board sensors. To construct a universal and adaptive navigation system, this study employs a proximal policy-optimized DRL algorithm in conjunction with incremental course learning and LSTM. Li et al. [30] presented a unique DRL framework to aid autonomous navigation in complex situations. This framework takes temporal abstractions and policy efficiency into account. It chooses the frequency of action decisions dynamically with efficiency regularization. Chikhaoui et al. [31] provided an autonomous framework for DRL-based UAV navigation. This framework is built on the PPO algorithm and takes the UAV's energy limit into consideration.

B. Hierarchical Reinforcement Learning

HRL is a method for exploring high-level object spaces in order to solve sparse reward or long-term challenges. The modular structure of HRL techniques facilitates transfer and multi-task learning in general because the concepts of sub-goals, alternatives, skills, and macro-actions are interrelated. Hierarchical planning is a well-known AI topic. Kulkarni et al. [32] developed hierarchical DQN (h-DQN) which operates at various time scales. It can hierarchically group goal-driven and intrinsically motivate DRL modules. h-DQN combines action-value functions at both the top and the bottom levels. The former acquires techniques for intrinsic subgoals or choices and the latter learns the original action's policy in order to achieve the aim of each sub-goal. Vezhnevets et al. [33] developed Feudal Networks (FuNs) for HRL. It contains a manager module and a worker module. The manager module sets abstract operations and sub-goals for a long period, and the worker module picks atomic actions at each time step to realize the manager's sub-goals. Bacon et al. [34] developed an option-criticism architecture and established the policy gradient theorem for options. This method gradually learns within-option policies and termination conditions using policy-to-options. It also integrates option discovery with option learning. Through the prism of options criticism, Harutyunyan et al. [35] examined the dilemma that exists between the flexibility of short options and the efficiency of long options. Similar to the off-policy learning algorithms, the authors decoupled behavior from termination state, and converted option learning into multi-step off-policy learning.

Intelligent species have the ability to explore their environs and gain important abilities without human intervention. Eysenbach et al. [36] developed an algorithm called DIAYN. It is a strategy to acquire valuable skills without a reward function. DIAYN optimizes an information theoretic goal to gain knowledge when it enforces the principle of least-entropy in its decision-making process. A novel hierarchical policy gradient with an unbiased latent-dependent baseline has been developed by Li et al. [37], which is called Hierarchical Proximal Policy Optimization (HiPPO). HiPPO is a mechanism to effectively train all levels of the hierarchy at the same time. They also devised a way to train time-abstractions that increases the resilience of the acquired abilities to changes of the environment. To facilitate exploration and hierarchical skill acquisition, an approach known as Hypothesis Proposal and Evaluation (HyPE) was created by Chuck et al. [38] and implemented in the software. The sample efficiency of HyPE is derived from implicit presumptions of behavior in both the actual world and simulation environments. Zhang et al. [39] introduced a hierarchical reinforcement learning approach called HIDIO. This technique is used to learn task-agnostic options in a self-supervised manner. These options can be used to solve sparse-reward problems at the same time. In this work, we combine the hierarchical framework with the DRL algorithm to improve the algorithm's performance on long-term tasks.

C. DrQ Algorithm

In order to optimize maximum entropy policies and energy-based policies, Haarnoja et al. [40] designed a soft Q-learning algorithm. An optimal policy is specified as a Boltzmann distribution in soft Q-learning, and a variational approach is used to build a sampling network. It can approximate samples from the distribution described by the ideal policy. In order to increase transfer ability, it is possible to enhance exploration by soft Q-learning, and to assist stochastic energy-based strategies in achieving compositionality. In the following year, Haarnoja et al. [41] developed an algorithm called soft actor-critic (SAC), which is based on the maximum energy reinforcement learning framework in [40]. The actor tries to maximize both predicted entropy and reward. SAC is an off-policy approach that serves as a link between deterministic policy gradient and stochastic policy optimization. With the clipped double-Q technique and an entropy regularization in its objective function, SAC trains the policy to optimize a trade-off between entropy and anticipated return while still maintaining a reasonable level of entropy. Essentially, entropy is a measure of the degree to which a strategy has been randomized. This process is analogous to the trade-off between exploitation and exploration. Furthermore, it has the potential to prevent the learning strategy from converging to a sub-optimal local optimal. DrQ [21] use data augmentation on the picture inputs to provide a more reliable output. There are two approaches to regularize the value function in DrQ. It can provide a natural way to use MDP structure in

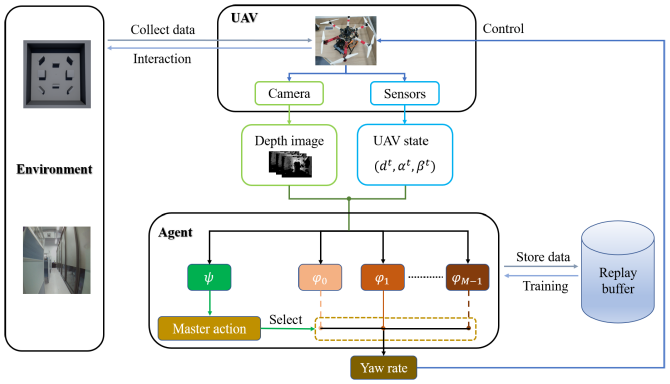

Fig. 1. DRL-based UAV autonomous navigation system.

a straightforward manner for model-free DRL learning. In our work, we propose the UAV autonomous navigation method based on DrQ. Simulations show evident improvements comparing to other state-of-the-art methods.

III. DRL-Based UAV Autonomous Navigation

In this section, a reactive framework based on DRL is presented to perform UAV autonomous navigation in an unknown environment. Instead of relying on SLAM, the proposed framework navigates the UAV with the sensor data that is presently available. Moreover, our framework doesn't need extensive optimization on-board, which is advantageous for tiny UAVs with limited computing resources. The system framework is shown in Fig.1.

A. Problem Formulation

UAVs' autonomous navigation in an unfamiliar environment will encounter a sequential decision-making challenge. According to the definition of a suitable reward function $R_a(s, s')$, this issue can be described as an MDP. Here s' denotes the next state, and the current state is s. The challenge of UAV autonomous navigation is formulated by MDP in this work. An MDP can be defined by a tuple $< S, A, R, P, \gamma >$. This tuple is composed of a set of states S, a set of actions A, a reward function $R(s, a)$, a transition function $P(s'|s, a)$, and a discount factor $\gamma \in (0, 1)$. A number of approaches can be used to solve the MDPs with finite states and action spaces, such as dynamic programming. The transition probabilities and reward functions, on the other hand, are not accessible in the majority of MDPs. The aim of the RL algorithm is to identify a best policy π that maps states to actions. Suppose that the UAV launches from a 3D point indicated as (x_0, y_0, z) in the Earth-centered coordinate system and flies to a target point indicated as (x_d, y_d, z). The state at time t comprises of some raw depth pictures as well as various UAV state characteristics, which are as follows: $o_t = [o^t_{depth}, o^t_{state}]$. The UAV state features can be defined as: $o^t_{state} = [d^t, \alpha^t, \beta^t]$, where d^t represents the euclidean distance between the UAV's current location and the target location, α^t represents the yaw angle of the UAV, and β^t represents the angle between the UAV's forward direction and the target position. The yaw angular velocity of the UAV is represented by the action a which is created by the policy network π.

B. HTARADrQ

There are some problems for the original DrQ method such as unstable training process, lack of processing power for time series data, and poor performance on long-term tasks. These problems lead DrQ to a bad performance on UAV autonomous navigation task. To improve the algorithm's performance on the UAV autonomous navigation task, we propose Hierarchical Temporal Attention Recurrent Averaged DrQ (HTARADrQ). HTARADrQ combines DrQ with an averaged estimate function, temporal attention and hierarchical framework. The averaged estimate function makes the training process more stable. The algorithm with temporal attention mechanism can handle time series data better. Hierarchical framework can divide long-term tasks into a hierarchy of sub-tasks. The high-level policy can identify the best sub-task and take it as the high-level action through learning. The sub-task itself is potentially easier to learn, thereby further improving the performance of the algorithm. The details of the improvements are described below.

1) Averaged Estimate Function: DrQ employs soft policy iteration to optimize the reward earned from the agent's interaction with the environment. The purpose of soft policy iteration is to alternate between policy review and policy improvement while it can only function within the maximum entropy paradigm. The DrQ agent's network is composed of three parts: the actor network ϕ, the critic network θ, and the target critic network $\overline{\theta}$. The actor network can predict agent's action according to agent's current state. The value of state-action pair is estimated by θ. $\overline{\theta}$ is the same as θ and can be used to estimate the target value of state-action pair. Soft Q-learning, like Q-learning, has a Q value overestimation issue. In our work, we alleviate the overestimation of soft Q-learning by averaging the estimation policy. This can make the training process more stable and improve the performance. The averaged estimate function is illustrated in the following equations:

$$J_Q(\theta) = \mathbb{E}_{(s_t, a_t, s_{t+1}) \sim D}\Big[\frac{1}{2}\big(Q_\theta(s_t, a_t) - Q_{target}\big)\Big]^2 \quad (1)$$

$$Q_{target} = r(s_t, a_t) + \gamma \frac{1}{K} \sum_{k=1}^{K} V(s_{t+1}) \quad (2)$$

$$V(s_t) = \mathbb{E}_{a_t \sim \pi}\big[Q_{\overline{\theta}}(s_t, a_t) - \alpha \log(\pi_\phi(a_t|s_t))\big] \quad (3)$$

where $J_Q(\theta)$ is the loss function, $\mathbb{E}$ is the expected value operator, D is the replay buffer to store past experience, s_t is the state of UAV at time t, a_t is the action of UAV at time t, $Q_\theta(s_t, a_t)$ is the Q value function, γ is the discount factor, $r(s_t, a_t)$ is reward function, $V(s_t)$ is value function, α is the temperature factor, and K is the number of action state estimates that are learned previously. In addition, as shown in Fig.2, we replace the last fully connected layer of the CNN module with global average pooling (GAP). Compared with a fully connected layer, the advantage

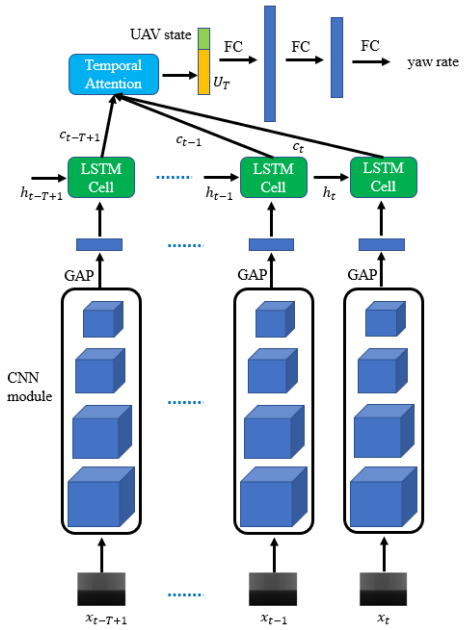

Fig. 2. Actor architecture of TARADrQ.

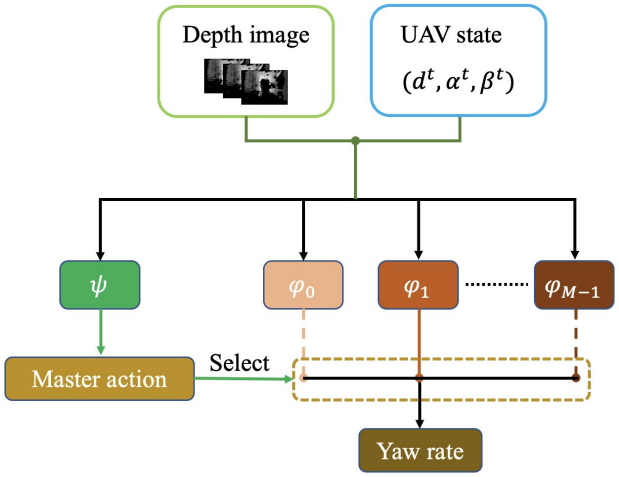

Fig. 3. Hierarchical framework.

of GAP is that overfitting in this layer can be avoided because there is no parameter to be optimized. Furthermore, GAP summarizes spatial information and becomes more robust to spatial transformations of the input.

2) Temporal Attention: In the autonomous navigation task of unmanned Aerial Vehicle (UAV), incomplete perception information and incidental noise are very common, caused by the partial observability of sensors. As a consequence, the UAV is unable to gather enough environmental information to develop the appropriate navigation behavior. In order to deal with this issue, the concept of recurrence is introduced to better assess the underlying state of the environment. We propose a recurrence mechanism in DrQ to improve the UAV autonomous navigation and an extra recurrent neural network (RNN) is added to the output of the CNN module. Instead of a single set of historical data as input, the RNN module analyses the temporal information contained inside the network. Additionally, because of the connectedness across time provided by RNN, a lengthier succession of historical data can be integrated and studied, making the generated policy more trustworthy. More specifically, we use the LSTM cell as the foundation of the RNN architecture and combine it with the DrQ algorithm, which is named as the Averaged DrQ (ADrQ).

Furthermore, in order to identify the most important frames in prior states, we propose Temporal Attention Recurrent Averaged DrQ (TARADrQ). TARADrQ can incorporate temporal attention over output of the LSTM cell layer, as illustrated in Fig.2. The temporal attention mechanism provides scalar weights for output of the LSTM cell. These weights are learned at distinct time steps. As shown in Eq.4, W_i is the weight of each LSTM cell output, h_i is LSTM cell hidden vector, w_i and b_i are learnable parameters, and the activation function is $ReLU$, followed by a $Softmax$ function. According to this concept, each learned weight relies on information both from the previous time step and the current state information, as indicated in Eq.4. In the next step, we calculate the combined context vector U_T which is shown in Eq.5, the context vector U_T is a weighted sum of LSTM cell outputs c_t across T time steps. Before computing the actions, U_T is concatenated with UAV state data and sent through three fully connected layers of the actor network. The learned weights W_i are the relevance of the LSTM output. As a result, the goal of the optimizing process is to maximize the likelihood of navigation success rate. To learn the proper behaviors, the optimizing process can learn to pick which states are comparatively more essential. This is because temporal attention can explicitly consider the LSTM output features from the previous T frames when the attention calculates the action output. Comparing with the original DrQ and ADrQ, TARADrQ has various improvements on capacity to process time series data and temporal relationships.

$$W_{t-i} = Softmax\big(ReLU(w_{t-i}h_{t-i} + b_{t-i})\big) \quad i = 0, \ldots, T-1 \tag{4}$$

$$U_T = \sum_{i=0}^{T-1} (W_{t-i}c_{t-i}) \tag{5}$$

3) Hierarchical Framework: The autonomous navigation task is a long-term task, and HRL can perform a difficult task by breaking it down into easier sub-tasks. It uses hierarchy of rules which have been learnt through RL. In a hierarchical structure, the highest-level policy often selects the sub-tasks of the primary task as the current actions. This policy is trained to complete the sub-tasks in order, then the incentive reward is produced and sent to the policy. By the internal reward associated with that sub-task, a lower-level policy can learn to accomplish the sub-task at the same level. Policy at the lowest level responds to identifying the most fundamental activities, which are referred to as primitive actions. Now we propose HTARADrQ, as shown in Fig.3, which consists of one master policy ψ and M sub-policy φ. At each time step, the master policy will predict a master action a_m according to current state s_t, as shown in Eq.6. a_m is the index of sub-policy

from 0 to $M-1$. Then, the algorithm will select a sub-policy according to master action a_m. The selected sub-policy will predict the UAV control action a_t according to current state s_t in N steps. For the current state, the agent can learn to select one from the M sub-policy and apply this sub-policy to predict action.

$$\begin{aligned} a_m &= Masterpolicy(s_t) = 0, 1, \ldots, M-1 \\ a_t &= Subpolicy_{a_m}(s_t) \end{aligned} \tag{6}$$

C. Reward Function

Eq.7 and Eq.8 are reward functions of sub-policy and mater policy, respectively. d_{pre} is the euclidean distance between the previous location of UAV and its target location. d_{cur} is the euclidean distance between the current location and the target location. If the UAV crashes or times out, the sub-policy will get a negative reward of -1 to reduce the probability that the model will select the action in that state. If the UAV arrives at the target point, the sub-policy will get a positive reward of 1 to increase the probability that the model will select the action in that state. In other conditions, the UAV will get reward $d_{pre}-d_{cur}$. The reward is positive when the drone is approaching the end point. Otherwise, the reward is negative. Eq.8 is the reward function of master policy. r_m is the reward of master policy. ${r_s}^i$ is the reward of sub-policy at the i-th step.

$$r_s = \begin{cases} -1, & crash \\ d_{pre} - d_{cur}, & others \\ 1, & arrived \end{cases} \tag{7}$$

$$r_m = \sum_{i=1}^{N} r_s^i \tag{8}$$

D. Hierarchical Temporal Attention Recurrent Averaged DrQ

In this section, we will describe the training process of HTARADrQ algorithm. First, we need to initialize the required parameters, including master policy ψ, sub-policy φ, master buffer D^m and sub buffer D^s. The master policy network consists of the network ξ and the target network $\overline{\xi}$. The sub-policy network consists of the actor network ϕ, the critic network θ and the target critic network $\overline{\theta}$. Then, the environment will be reset before a task is performed. During task period, if the current step is less than the seed steps, the master policy and sub-policy will randomly select an action. Otherwise, they will predict the action according to current state. The sub-policy is selected by master action and the sub-policy will be execute in N steps. In each sub-step, the agent will perform the sub action and the sub-policy will be updated. When the task is completed or terminated, the master policy will be updated. The complete algorithm is organized as Algorithm 1.

Algorithm 1 Hierarchical Temporal Attention Recurrent Averaged DrQ (HTARADrQ)

Require: Environment steps Z, seed steps n, batch size B, number of sub-policy M, number of sub step N, optimizer type, master policy learning rate λ_m, sub-policy learning rate λ_s, discount factor γ, temperature factor α

initialization: Randomly initialize master policy ψ, sub-policies φ, master buffer D^m, sub buffer D^s
for $i = 1$ to Z **do**
  Reset environment
  while the termination condition NOT satisfied **do**
    Get s_{t_m} from environment
    if $i < n$ **then**
      Randomly select an action as a_{t_m}
    else
      Feed s_{t_m} into master policy ψ to predict action a_{t_m}
    end if
    Select sub-policy according to action a_{t_m}
    for $j = 1$ to N **do**
      $i \leftarrow i + 1$
      Get s_{t_s} from environment
      if $i < n$ **then**
        Randomly select an action as a_{t_s}
      else
        Feed s_{t_s} into sub-policy φ to predict action a_{t_s}
      end if
      The agent performs the action a_{t_s} and gets reward r_{t_s} and state s'_{t_s} from environment
      $D^s \leftarrow D^s \cup (s_{t_s}, a_{t_s}, r_{t_s}, s'_{t_s})$
      Sample a batch size transition from D^s to update sub-policy according to Eq.1 and equation below:

$$J_\pi(\phi) = \mathbb{E}_{s_t \sim D^s}\big[\log \pi_\phi(\cdot|s_t) - Q_\theta(s_t, \pi_\phi(s_t))\big] \tag{9}$$

    end for
    Compute reward r_{t_m} according to Eq.8
    Get s'_{t_m} from environment
    $D^m \leftarrow D^m \cup (s_{t_m}, a_{t_m}, r_{t_m}, s'_{t_m})$
  end while
  Sample a batch size transition from D^m to update master policy by equation below:

$$J_Q(\xi) = \mathbb{E}_{(s_t, a_t, s_{t+1}) \sim D^m}\Big[\frac{1}{2}\big(Q_\xi(s_t, a_t) - (r_t + \gamma\, Q_{\overline{\xi}}(s_{t+1}))\big)\Big]^2 \tag{10}$$

end for

IV. Simulation Results and Discussions

In this part, we test the HTARADrQ algorithm in our simulation environment to evaluate its capabilities of autonomous navigation and obstacle avoidance. Numerous parameters are allocated based on the actual state of the environment, and the agent is trained to learn the policy in 100,000 environment steps. In order to provide a point of comparison, the SAC [41], DrQ [21], ADrQ, and TARADrQ algorithms are all trained for the UAV autonomous navigation task. The proposed method is tested in the UAV simulation environment Airsim.

A. Training Environment and Setting

To validate the performance of the HTARADrQ for UAV navigation tasks, firstly we conduct the experiment in simulation. We build three environments, which are shown in Fig.4.

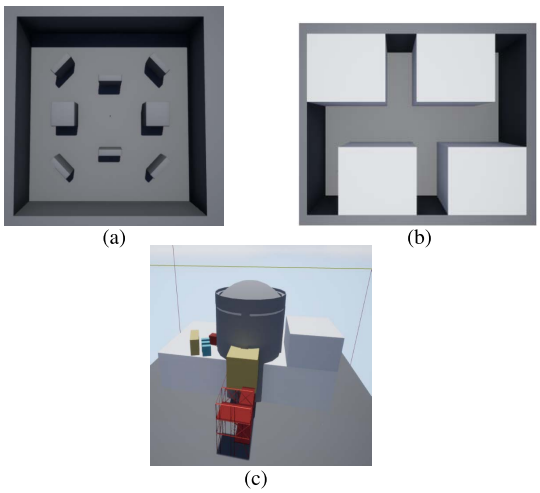

Fig. 4. Experiment environment. (a) is environment A. (b) is environment B. (c) is environment C.

TABLE I
PARAMETER SETTING

Parameter	Setting
Total environment step	100000
Master policy replay buffer capacity	20000
Sub-policy replay buffer capacity	10000
Sub-policy number	3
Seed steps	2000
Batch size	128
Discount factor	0.99
Optimizer	Adam
Master policy learning rate	2.5×10^{-4}
Sub-policy learning rate	2.5×10^{-4}
Temperature factor	0.1

Environment A is a carefully designed scene with difficulty and general obstacles. Environment B is an indoor corridor environment where the white building and surrounding walls are obstacles. Environment C is a simulation environment of a nuclear power plant that is considered as the next real-world application. A normalization range of [0, 1] is applied to all sensory inputs. The weights of both master policy and sub-policy networks are initialized with a uniform distribution.

During training, the flying speed of the drone is kept at 1.0 m/s. In order to increase the randomness, the starting point and ending point of each episode are randomly initialized during training. The input of the algorithm contains depth image and states of UAV. The size of depth image is 144 x 256. All of the input data are normalized to [0, 1]. The output of the algorithm is the yaw angular velocity in the range of [-60, 60]. During training, the action selection in the first 1,000 steps is made by a randomization algorithm. Each sub-policy model of HTARADrQ will be trained once it interacts with the environment. When each episode is terminated, master policy model will be trained 30 times. For every 5,000 interactions, the trained model will be tested with 40 episodes. The start and the end points of each test episode are also randomly initialized. When the test process is completed, the training process will continue. The training phase will end when the number of interactions reaches 100,000. Table I has a condensed representation of all of the hyper-parameters. Setting the parameters as shown in Table I can stabilize the model training and speed up the convergence.

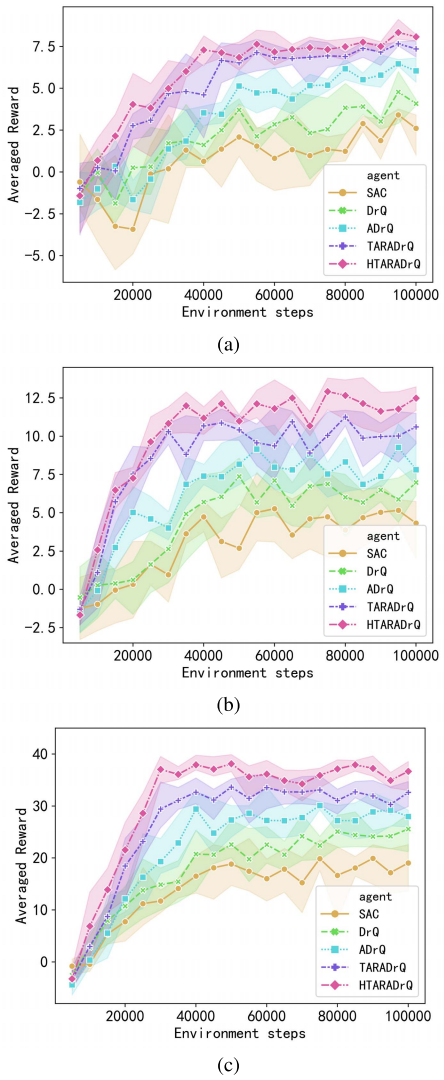

Fig. 5. Average episode reward curve during training. (a) is the training results of each algorithm in environment A. (b) is the training results of each algorithm in environment B. (c) is the training results of each algorithm in environment C.

B. Training Result and Analysis

Firstly, we evaluate the performance of final HTARADrQ model in these three environments and compare it with SAC, DrQ, ADrQ, and TARADrQ. The parameter settings and training process of them are the same. Fig.5 depicts their average episode return during training. Three random seeds were used to train each of the five algorithms. In every 5,000 steps, each algorithm will perform one evaluation roll-out. The solid curves represent the mean of the three seeds, while the shaded zone represents the variation of the three seeds. The solid curve shows that HTARADrQ outperforms the others on the UAV autonomous navigation tasks, including learning speed and overall performance. The shaded region shows that HTARADrQ is more stable than the others during training. According to HTRADrQ, the average estimate function lowers both the likelihood of an incorrect Q value and

TABLE II
AVERAGE NUMBER OF FLIGHT STEPS FOR EACH MODEL IN A COLLISION SITUATION

Algorithm	Environment A	Environment B	Environment C
SAC	37	47	69
DrQ	45	59	81
ADrQ	53	72	95
TARADrQ	66	80	104
HTARADrQ	**71**	**88**	**111**

the variance of the parameter transfer process. The parameters of the model are reduced by replacing the fully connected layer with a GAP. The recurrent mechanism enables DrQ to process the time series data and to extract time series information from it. The temporal attention mechanism can determine the importance of each input frame. It leads to more accurate predictions. The hierarchical framework decomposes long-term learning tasks into multiple sub-problems or hierarchies of sub-tasks. The high-level policies perform tasks by selecting the best sub-tasks as high-level actions. In terms of the sequence of sub-tasks, task decomposition can effectively shorten horizon of the original task. It is easier to learn the sub-tasks. Thus, compared with other algorithms, HTARADrQ is more stable during training and converges faster.

Secondly, we examine the collision avoidance of HTARADrQ comparing to SAC, DrQ, ADrQ, and TARADrQ. The results are shown in Table II. We find that the average collision step increases if an average estimation function, a temporal attention, and a hierarchical framework are added. In other words, the obstacle avoidance ability of the agent is improved. The training process of the algorithm appears more stable if the average estimate function and the global average pooling operation are deployed. This can somehow improve the obstacle avoidance ability of the agent. With the recurrent and temporal attention mechanisms, the agent is able to deal with and extract richer and more critical information from its surroundings. Thus the agent can avoid obstacles and improve accuracy of prediction. The hierarchical framework facilitates the learning of sub-tasks. It allows the agent to acquire a more effective obstacle avoidance strategy.

In order to examine different performance among SAC, DrQ, ADrQ, TARADrQ, and HTARADrQ, we test these models in each environment. In each environment, we test 100 rounds, in which the start and end points of each test are randomly generated. Fig.6 shows the average success rate, average collision rate, and average time-out rate of navigation tasks. The blue bars in the graph represent the average success rate. The orange bars in the graph represent the average collision rate. The green bars in the graph represent the average timeout rate. It can be seen that the autonomous navigation model can learn a better navigation strategy when combined with attention and hierarchical mechanisms. Thus, compared with the SAC, DrQ, ADrQ, and TARADrQ, HTARADrQ can learn a better autonomous navigation strategy and has a higher success rate and a lower collision rate.

C. Attention Mechanism

In order to verify the effectiveness of the temporal attention mechanism, this section uses the trained model to conduct

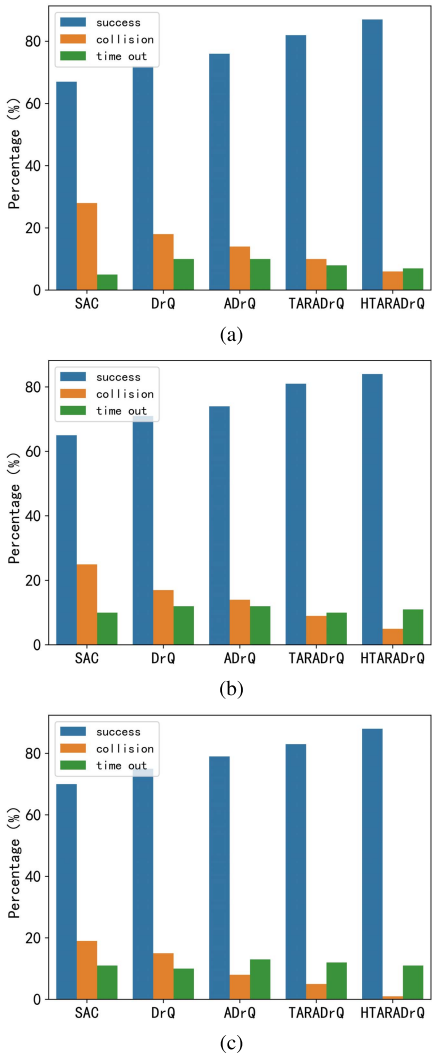

Fig. 6. Testing results. (a) is the testing results of each algorithm in environment A. (b) is the testing results of each algorithm in environment B. (c) is the testing results of each algorithm in environment C.

various experiments in the simulation environment B, such as direct flight experiments, turning experiments, and obstacle avoidance experiments.

Fig.7 shows the prediction results of the navigation model in the direct flight experiment. The first line is the top view of the environment, and the green dot indicates the position of the drone. The second line is the RGB image of the environment. The third line is the depth image of the environment. The fourth line is the attention weights corresponding to the depth image of each frame. It can be seen from the figure that the attention weights continuously increase from the first frame to the fourth frame. The attention module believes that the depth image of the fourth frame is more important than the other three frames, so the maximum weight is given to the fourth frame. Then, based on the attention weights of the four-frame depth image, the prediction of the navigation model is 0.15°/s. At this yaw angular velocity, the UAV can fly forward. The above experimental results show that the navigation model has learned a better policy of direct flight.

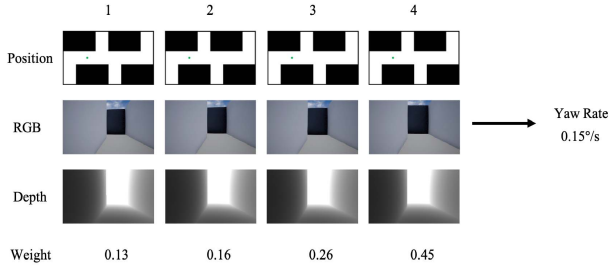

Fig. 7. Prediction results of navigation model in the direct flight experiment.

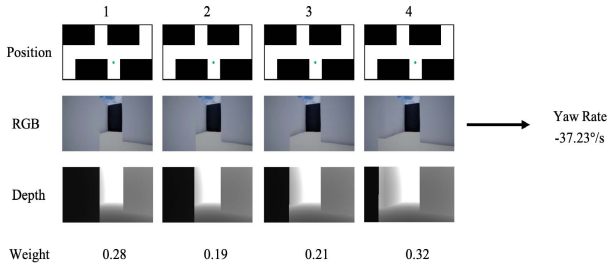

Fig. 8. Prediction results of navigation model in the turning experiment.

Fig.8 shows the prediction results of the navigation model in the turn experiment. As can be seen from the figure, the attention weight of the first frame and the fourth frame is larger than the second frame and the third frame. The attention module believes that the information provided by the fourth frame is the most important, but the information provided by the first frame should also be considered. In the first depth image, almost half of the area is black. It indicates that the UAV is very close to the left wall at this time. Thus the attention module gives a larger weight to the first frame. Then, based on the attention weights of the four-frame depth image, the prediction of the navigation model is -37.23°/s. The above experimental results show that the navigation model has learned a better policy of turning.

Fig.9 shows the prediction results of the navigation model in the obstacle avoidance experiment. It can be seen from the figure that the difference in the attention weights from the first frame to the fourth frame is not very large. It means the attention module believes that the information provided by these four frames are all important. Then, based on the attention weight of the four-frame depth image, the prediction of the navigation model is 59.68°/s. At this yaw angular velocity, the UAV will turn right to avoid the wall on the left to prevent a collision event. The above experimental results show that the navigation model has learned a better policy of obstacle avoidance.

D. Ablation Experiment

In order to illustrate the impact of the hierarchical mechanism on performance, and the effect of combination between the hierarchical mechanism and other improvement methods, we conduct an ablation experiment in this section. The experimental results are shown in Table III. As can be seen from the table, the average success rate of the model in each

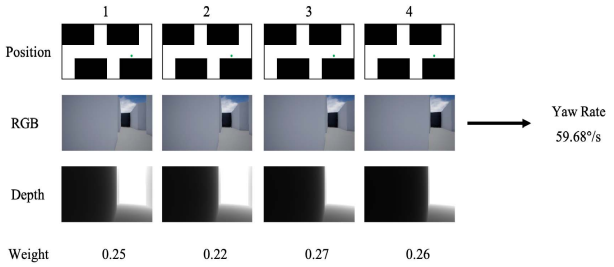

Fig. 9. Prediction results of navigation model in obstacle avoidance experiments.

TABLE III
THE AVERAGE SUCCESS RATE RESULTS OF EACH MODEL IN THE NAVIGATION TASK IN EACH ENVIRONMENT

Hierarchical	Recurrence	Attention	Averaged	Environment A	Environment B	Environment C
—	—	—	—	72%	71%	75%
✓	—	—	—	74%	75%	78%
✓	✓	—	—	78%	77%	80%
✓	✓	✓	—	82%	81%	84%
✓	—	—	✓	77%	79%	81%
✓	✓	✓	✓	**87%**	**84%**	**88%**

environment is improved by 2% to 4% when the hierarchical mechanism is added. Comparing to the algorithm only with hierarchical mechanism, the average success rate of the model is increased by 2% to 8% when the algorithm is added with hierarchical mechanism and other methods. When the four improvement methods are added to the algorithm, the model has the highest average success rate. It can be seen that the hierarchical mechanism can improve the performance to a certain extent. Meanwhile, the hierarchical mechanism can combine with the recurrent mechanism, the attention mechanism, and the average value function to promote each other and further improve the model performance.

V. REAL WORLD FLIGHT TEST

In this section, some real-world tests are carried out in order to verify the performance of our trained UAV autonomous navigation model, as well as for practical results in general.

A. Flight Platform

The UAV flight platform is built based on DJI F550 as shown in Fig.10, and is equipped with CUAV V5 autopilot. We design and equip the Manifold 2C on-board computer with a depth sensor, and also integrate the SLAM algorithm for positioning and navigation. It forms a multi-functional experimental platform with convenient assembly, clear structure, autonomous flight capability, and can be alienated according to different experimental tasks. We also develop a ground station platform which can do route planning and real-time monitoring of UAV status based on QGroundControl.

B. Evaluation in the Real World

First, a simulation experiment in an office environment was carried out. The floor plan of the office environment is shown in Fig.11, where the shaded areas represent obstacles, and the thick lines represent the surrounding walls. The starting and ending positions of this experiment are also shown in

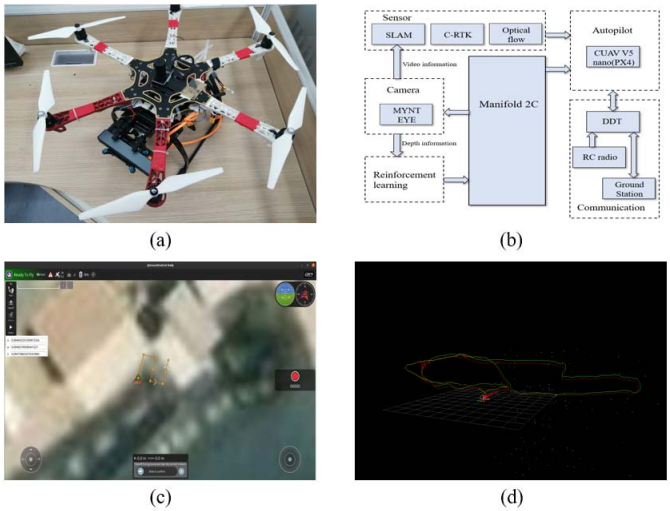

Fig. 10. Flight platform. (a) is our self-assembled UAV. (b) is the navigation system framework. (c) is the ground station platform. (d) is a test result of SLAM system.

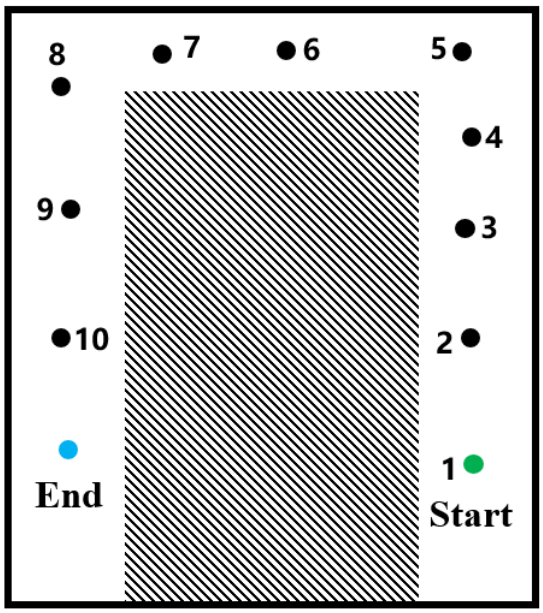

Fig. 11. Top view of test environment.

the figure. The positions marked with numbers in Fig.11 correspond to the positions in Fig.12. The environment depth image and UAV status information are continuously collected during the flight. The collected environmental depth image and UAV state information are input into the navigation model, and the navigation model predicts the yaw angular velocity of the UAV.

The model used in the experiment is the trained HTARADrQ model. The prediction results of the model are shown in Fig.12. The first column is UAV's position number, the second column is the original environment image, the third column is depth image of the environment, and the fourth column is the predicted yaw rate for the navigation model. Negative yaw angular velocity means yaw to the left,

Position	RGB	Depth	Prediction(rad/s)
1			-7.29
2			3.85
3			-1.36
4			-35.24
5			-19.27
6			9.66
7			-45.10
8			-51.51
9			16.37
10			5.48

Fig. 12. Navigation model prediction results.

while positive means yaw to the right. As can be seen from the figure, the predicted yaw rate for the first to the third position is between [−10, 10], because the UAV does not need to make a large angle of deflection at these positions, and it only needs to move forward. At the fourth and fifth positions, it can be seen that the drone should turn left at these positions, and the predicted values of the yaw angular velocity are all large negative values. The sixth yaw rate prediction is between [−10, 10] because at this position, the drone should be flying forward. At the seventh and eighth positions, the UAV should be deflected to the left at these positions. At this time, the predicted value is a large negative value, so as to make the UAV turn to the left and avoid a collision. At the ninth position, it can be seen that there are obstacles on the left, and the predicted yaw rate is positive at this time, so that the drone can deflect slightly to the right and avoid the obstacles. At the last position, one can see that the end position is not far ahead, and the drone only needs to fly forward. At this time, the predicted value of the yaw angular velocity is between [−10, 10].

VI. Conclusion and Future Work

We study the autonomous navigation challenge of a UAV in an unfamiliar environment, and the DRL approach is used to tackle the issue. We propose the HTARADrQ algorithm which allows the UAV to conduct actions in continuous action space better with an averaged estimate function, recurrent mechanism, temporal attention, and a hierarchical framework. A real-time simulation is performed to show the validation of the method, in which the UAV attempts to achieve the target without colliding with any obstacles. Our testing results demonstrate that the technique described in this paper

is acceptable for UAV autonomous navigation, and it also outperforms the original DrQ algorithm. For future work, a more realistic simulation environment can be built to narrow the gap between the simulation environment and the real environment. The energy consumption during flight can also be considered when the reward is calculated. It is possible to further improve the navigation performance of the algorithm in complex environments and paths if the velocity and angular velocity are considered during the flight.

REFERENCES

[1] H. Huang, A. V. Savkin, and W. Ni, "Online UAV trajectory planning for covert video surveillance of mobile targets," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 2, pp. 735–746, Apr. 2022.

[2] A. Singla, S. Padakandla, and S. Bhatnagar, "Memory-based deep reinforcement learning for obstacle avoidance in UAV with limited environment knowledge," *Trans. Intell. Transp. Syst.*, vol. 22, no. 1, pp. 107–118, Jan. 2021.

[3] J. Shi, L. Zhao, X. Wang, W. Zhao, A. Hawbani, and M. Huang, "A novel deep Q-learning-based air-assisted vehicular caching scheme for safe autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 4348–4358, Jul. 2021.

[4] J. Dong, K. Ota, and M. Dong, "UAV-based real-time survivor detection system in post-disaster search and rescue operations," *IEEE J. Miniaturization Air Space Syst.*, vol. 2, no. 4, pp. 209–219, Dec. 2021.

[5] P. Huang, Y. Wang, K. Wang, and K. Yang, "Differential evolution with a variable population size for deployment optimization in a UAV-assisted IoT data collection system," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 4, no. 3, pp. 324–335, Jun. 2020.

[6] H. Cao, H. Yao, H. Cheng, and S. Lian, "A solution for data collection of large-scale outdoor Internet of Things based on UAV and dynamic clustering," in *Proc. IEEE 9th Joint Int. Inf. Technol. Artif. Intell. Conf. (ITAIC)*, Dec. 2020, pp. 2133–2136.

[7] M. Odelga, P. Stegagno, N. Kochanek, and H. H. Bulthoff, "A self-contained teleoperated quadrotor: On-board state-estimation and indoor obstacle avoidance," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 7840–7847.

[8] X. Wang and W. Wang, "Nonlinear signal-correction observer and application to UAV navigation," *IEEE Trans. Ind. Electron.*, vol. 66, no. 6, pp. 4600–4607, Jun. 2019.

[9] J. Tiemann, A. Ramsey, and C. Wietfeld, "Enhanced UAV indoor navigation through SLAM-augmented UWB localization," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, May 2018, pp. 1–6.

[10] J. Kim, J. Guivant, M. L. Sollie, T. H. Bryne, and T. A. Johansen, "Compressed pseudo-SLAM: Pseudorange-integrated compressed simultaneous localisation and mapping for unmanned aerial vehicle navigation," *J. Navigat.*, vol. 74, no. 5, pp. 1–13, 2021.

[11] N. Imanberdiyev, C. Fu, E. Kayacan, and I.-M. Chen, "Autonomous navigation of UAV by using real-time model-based reinforcement learning," in *Proc. 14th Int. Conf. Control, Autom., Robot. Vis. (ICARCV)*, Nov. 2016, pp. 1–6.

[12] C. Wang, J. Wang, J. Wang, and X. Zhang, "Deep-reinforcement-learning-based autonomous UAV navigation with sparse rewards," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6180–6190, Jul. 2020.

[13] L. He, N. Aouf, J. F. Whidborne, and B. Song, "Integrated moment-based LGMD and deep reinforcement learning for UAV obstacle avoidance," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 7491–7497.

[14] T. Guo, N. Jiang, B. Li, X. Zhu, Y. Wang, and W. Du, "UAV navigation in high dynamic environments: A deep reinforcement learning approach," *Chin. J. Aeronaut.*, vol. 34, no. 2, pp. 479–489, Feb. 2021.

[15] S. Zhang, Y. Li, and Q. Dong, "Autonomous navigation of UAV in multi-obstacle environments based on a deep reinforcement learning approach," *Appl. Soft Comput.*, vol. 115, pp. 108–120, Jan. 2022.

[16] B. Xin and C. He, "DRL-based improvement for autonomous UAV motion path planning in unknown environments," in *Proc. 7th Int. Conf. Control Robot. Eng. (ICCRE)*, Apr. 2022, pp. 102–105.

[17] S. F. Abedin, M. S. Munir, N. H. Tran, Z. Han, and C. S. Hong, "Data freshness and energy-efficient UAV navigation optimization: A deep reinforcement learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 9, pp. 5994–6006, Sep. 2021.

[18] M. Chansuparp and K. Jitkajornwanich, "A novel augmentative backward reward function with deep reinforcement learning for autonomous UAV navigation," *Appl. Artif. Intell.*, vol. 36, no. 1, Dec. 2022, Art. no. 2084473.

[19] J. Rafati and D. C. Noelle, "Learning representations in model-free hierarchical reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, no. 1, pp. 10009–10010.

[20] C. Florensa, Y. Duan, and P. Abbeel, "Stochastic neural networks for hierarchical reinforcement learning," in *Proc. ICLR*, 2017, pp. 1–6.

[21] D. Yarats, I. Kostrikov, and R. Fergus, "Image augmentation is all you need: Regularizing deep reinforcement learning from pixels," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–6.

[22] N. Smolyanskiy, A. Kamenev, J. Smith, and S. Birchfield, "Toward low-flying autonomous MAV trail navigation using deep neural networks for environmental awareness," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 4241–4247.

[23] A. Kouris and C.-S. Bouganis, "Learning to fly by MySelf: A self-supervised CNN-based approach for autonomous navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 1–9.

[24] A. Loquercio, A. I. Maqueda, C. R. Del Blanco, and D. Scaramuzza, "DroNet: Learning to fly by driving," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 1088–1095, Apr. 2018.

[25] E. Kaufmann, A. Loquercio, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: Learning agile flight in dynamic environments," in *Proc. Conf. Robot Learn.*, vol. 87, 2018, pp. 133–145.

[26] H. Y. Lee, H. W. Ho, and Y. Zhou, "Deep learning-based monocular obstacle avoidance for unmanned aerial vehicle navigation in tree plantations," *J. Intell. Robot Syst.*, vol. 101, no. 5, pp. 1–18, Dec. 2020.

[27] M. Hii, P. Courtney, and P. Royall, "An evaluation of the delivery of medicines using drones," *Drones*, vol. 3, no. 3, p. 52, Jun. 2019.

[28] S.-Y. Shin, Y.-W. Kang, and Y.-G. Kim, "Obstacle avoidance drone by deep reinforcement learning and its racing with human pilot," *Appl. Sci.*, vol. 9, no. 24, p. 5571, Dec. 2019.

[29] V. J. Hodge, R. Hawkins, and R. Alexander, "Deep reinforcement learning for drone navigation using sensor data," *Neural Comput. Appl.*, vol. 33, no. 6, pp. 2015–2033, Mar. 2021.

[30] C.-C. Li, H.-H. Shuai, and L.-C. Wang, "Efficiency-reinforced learning with auxiliary depth reconstruction for autonomous navigation of mobile devices," in *Proc. 23rd IEEE Int. Conf. Mobile Data Manag. (MDM)*, Jun. 2022, pp. 458–463.

[31] K. Chikhaoui, H. Ghazzai, and Y. Massoud, "PPO-based reinforcement learning for UAV navigation in urban environments," in *Proc. IEEE 65th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2022, pp. 1–4.

[32] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 3675–3683.

[33] A. S. Vezhnevets et al., "Feudal networks for hierarchical reinforcement learning," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, vol. 70, Aug. 2017, pp. 3540–3549.

[34] P.-L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," in *Proc. AAAI Conf. Artif. Intell.*, 2017, vol. 31, no. 1, pp. 270–278.

[35] A. Harutyunyan, P. Vrancx, P.-L. Bacon, D. Precup, and A. Nowe, "Learning with options that terminate off-policy," in *Proc. AAAI Conf. Artif. Intell.*, 2018, vol. 32, no. 1, pp. 1–7.

[36] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine, "Diversity is all you need: Learning skills without a reward function," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–22. [Online]. Available: https://openreview.net/forum?id=SJx63jRqFm

[37] A. Li, C. Florensa, I. Clavera, and P. Abbeel, "Sub-policy adaptation for hierarchical reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–15. [Online]. Available: https://openreview.net/forum?id=ByeWogStDS

[38] C. Chuck, S. Chockchowwat, and S. Niekum, "Hypothesis-driven skill discovery for hierarchical deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 5572–5579.

[39] J. Zhang, H. Yu, and W. Xu, "Hierarchical reinforcement learning by discovering intrinsic options," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–19. [Online]. Available: https://openreview.net/forum?id=r-gPPHEjpmw

[40] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," in *Proc. 34th Int. Conf. Mach. Learn. (JMLR)*, vol. 70, 2017, pp. 1352–1361.

[41] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. 35th Int. Conf. Mach. Learn.*, vol. 80, 2018, pp. 1861–1870.

Zun Liu received the B.S. and Ph.D. degrees from the South China University of Technology in 2011 and 2019, respectively. He has been joining Shenzhen University and also the National Engineering Laboratory for Big Data System Computing Technology of China as a Research-Track Associate Professor since 2021. His current research interests include robotic, multi-agent systems, reinforcement learning, and intelligent transportation systems.

Yuanqiang Cao is currently pursuing the master's degree with Shenzhen University, China. His current research interests include robotic, machine learning, and heterogeneous collaborative robot mapping.

Jianyong Chen received the Ph.D. degree from the City University of Hong Kong, Hong Kong, in 2003. He was a Senior Engineer in network technology with ZTE Corporation, Shenzhen, China, from 2003 to 2006. Then, he joined Shenzhen University, Shenzhen, where he is currently a Professor with the College of Computer Science and Software Engineering. His research interests include artificial intelligence and information security.

Jianqiang Li received the B.S. and Ph.D. degrees from the South China University of Technology in 2003 and 2008, respectively. He is currently the Executive Director of the National Engineering Laboratory for Big Data System Computing Technology of China and also the Vice President with the College of Computer Science and Software Engineering, Shenzhen University. He served as the editorial board of seven journals, including IEEE TRANSACTIONS and has been selected into the list of the world's top scientists' lifelong influence by Stanford University. He led three projects of the National Natural Science Foundation and five projects of the Natural Science Foundation of Guangdong, China. His major research interests include robotic, hybrid systems, the Internet of Things, and embedded systems.