IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 1 July 2022, accepted 7 August 2022, date of publication 16 August 2022, date of current version 6 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3199070

APPLIED RESEARCH

Collaborative Decision-Making Method for Multi-UAV Based on Multiagent Reinforcement Learning

SHAOWEI LI, YUHONG JIA, FAN YANG, QINGYANG QIN, HUI GAO, AND YAOMING ZHOU
School of Aeronautic Science and Engineering, Beihang University, Beijing 100191, China

Corresponding author: Yaoming Zhou (zhouyaoming@buaa.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61903014, and in part by the Aeronautical Science Foundation of China under Grant 20200017051001.

ABSTRACT The collaborative mission capability of multi-UAV has received more and more attention in recent years as the research on multi-UAV theories and applications has intensified. The artificial intelligence technology integrated into the multi-UAV collaborative decision-making system can effectively improve the collaborative mission capability of multi-UAV. We propose a multi-agent reinforcement learning algorithm for multi-UAV collaborative decision-making. Our approach is based on the actor-critic algorithm, where each UAV is treated as an actor that collects data decentralized in the environment. A centralized critic provides evaluation information for each training step during the centralized training of these actors. We introduce a gate recurrent unit in the actor to enable the UAV to make reasonable decisions concerning historical decision information. Moreover, we use an attention mechanism to design the centralized critic, which can achieve better learning in a complex environment. Finally, the algorithm is trained and experimented in a multi-UAV air combat scenario developed in the collaborative decision-making environment. The experimental results show that our approach can learn collaborative decision-making strategies with excellent performance, while convergence performance is better compared to other algorithms.

INDEX TERMS UAV, multi-UAV, collaborative decision-making, multi-agent reinforcement learning.

I. INTRODUCTION

Unmanned aerial vehicles (UAVs) are playing increasingly important roles in military missions [1]. Compared with manned aerial vehicles, UAVs have outstanding advantages in performing boring, dirty, and dangerous tasks [2]. In recent years, due to the rapid development of technologies such as computer technology, electronic technology, and information technology, UAV technology has been greatly improved, and UAV has a stronger autonomous ability. Although the capabilities of individual UAVs have improved significantly, the number of tasks that a single UAV can accomplish alone remains limited due to the mission attributes, flight platforms, sensors, and other factors.

Multiple unmanned aerial vehicles (multi-UAV) are expected to be more capable than a single powerful UAV, which offers significantly enhanced flexibility and robustness [3], [4]. The multi-UAV approach leads to redundant solutions offering greater fault tolerance and flexibility including reconfigurability in case of failures of individual UAV [5]. In the suppression of enemy air Defenses (SEAD) mission [6], for example, even a high-performance UAV cannot attack multiple long-range targets at the same time. A multi-UAV formation, on the other hand, can carry more weapons and attack multiple targets at the same time, making it more efficient than using a single high-performance UAV to perform the mission. At the same time, if the electronic jamming UAVs are in the formation, the cooperation between the two types of UAVs can not only complete the mission efficiently but also reduce our losses. Therefore, it is of great importance to research the collaboration decision-making

The associate editor coordinating the review of this manuscript and approving it for publication was Cong Pu.

strategy between multiple UAVs, to obtain higher operation efficiency [7].

Multi-UAV collaborative decision-making methods can be divided into two types: centralized and distributed. In a centralized system, there is a central node responsible for the decision-making of the multi-UAV system. On the contrary, there is no central node in a distributed system, where members are equal to each other and complete the decision-making through negotiation. The centralized decision-making methods were developed earlier. J. Capitan *et al.* designed a centralized multi-UAV cooperative target surveillance algorithm by combining a partially observable Markov decision process model with an auction algorithm [8]. Huang *et al.* proposed a cross-entropy-based collaborative task assignment method for multiple UAVs, applying cross-entropy to the constrained task assignment problem [9]. Bai *et al.* proposed a collaborative multi-UAV trajectory planning method based on the hybrid algorithm of the artificial bee colony and A* [10]. In the centralized multi-UAV collaborative decision-making method due to the existence of a central UAV, as soon as when this UAV is destroyed or incapacitated, the UAV formation cannot work properly. To solve this problem, Xu *et al.* designed a dynamic selection method for the collaborative decision center of multi-UAV [11]. This approach achieves adaptive selection of the best decision center in deception and interference environments through information interaction between UAVs and the use of cloud models. Ma *et al.* proposed a Double Oracle combined with a neighborhood search algorithm, which was used to solve the occupancy positions problem in cooperative multi-UAV out-of-visual-range air combat [12]. By analyzing the influence of the UAV occupied position on the advantages and threats of both sides, the problem is described as a zero-sum matrix game. By solving the zero-sum games, the UAV position with the least threat is obtained. Although the centralized system is better coordinated, it is less real-time, scalable, and robust.

Many researchers have been conducted on distributed algorithms to cope with the highly dynamic characteristics of multiple UAVs. O. Ilaya proposed a distributed model predictive decision algorithm for the formation of decision-making of multi-UAV in an electronic jamming environment [13]. The algorithm couples neighboring UAVs using an aggregate cost function and coherence protocol by superimposing individual and group tasks linearly, using a neighborhood structure for decentralization. To achieve coherence, a decentralized control scheme based on a decentralized model predictive control strategy is developed and implemented. Zhen *et al.* proposed an intelligent self-organizing algorithm based on an improved distributed ant colony algorithm, which uses a distributed structure to decompose multi-UAV collaborative mission planning into multiple local optimization problems [7]. Based on the distributed partially observable Markov decision architecture, Zhao designed a multi-UAV cooperative target tracking decision algorithm by taking the state of a multi-UAV consistent target by a joint multi-objective probability distribution as the optimization objective [14].

Recently, Deepmind's AlphaStar and OpenAI's OpenAI Five beat top human players in their respective game domains. It shows the broad prospect of multi-agent reinforcement learning. The exploration of multi-agent reinforcement learning has been carried out in the fields of power systems [15], intelligent transportation [16] and communication [17]. Meanwhile, Multi-UAV systems as typical multi-agent systems, many scholars have done a lot of research on the application of multi-agent reinforcement learning algorithms in this field, as in the literature [18], [19], [20], [21], [22].

In this paper, the Multi-agent Transformer-based Actor-Critic algorithm is designed to solve the collaborative decision-making problem of multi-UAV. The algorithm has a shared critic network and multiple actor networks with shared parameters. To speed up the convergence of the algorithm we design a dual experience playback mechanism. Finally, we validate our algorithm in a multi-UAV coordinated air combat environment. The contributions of this paper are as follows:

1) The introduction of Transformer in the centralized critic allows the UAV to better notice the impact of environmental changes and thus rationally assess the actions currently taken.
2) The gate recurrent unit (GRU) model is added to Actor to enable UAVs to use historical decision information as the basis for current decisions. To reduce the policy estimate variance, we designed a new method for computing the multi-agent advantage function.
3) We designed an air combat scenario of multi-UAV based on a collaborative decision-making environment. In this scenario, we completed the training of our method and obtained the air combat strategy. We test the strategies obtained by our algorithm and the strategy of the other three methods respectively. The results show that the strategy obtained by our algorithm is better than other algorithms.

This paper is organized as follows, in the second section, a collaborative decision-making simulation environment is established, which includes the UAV model and attack model. In the third section, we design a multi-agent reinforcement learning algorithm called MATAC. In the fourth section, we validate our algorithm with a multi-UAV air combat scenario and analyze the obtained strategies. Finally, we conclude the paper and give an outlook on future work.

II. COLLABORATIVE DECISION ENVIRONMENT

In this section, we describe a multi-UAV collaborative decision environment, which consists of a UAV model and an attack model.

A. UAV MODEL

We describe the motion of the UAV in space using a simple three-degree-of-freedom model. We define the absolute

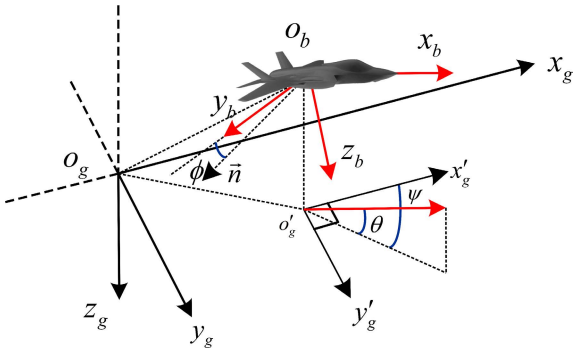


FIGURE 1. Three-degree-of-freedom motion model of the UAV in Earth-fixed reference. (o_g,x_g, y_g,z_g) is the Earth-fixed reference frame F_g, (o_b, x_b,y_b, z_b) is the body reference frame F_b of UAV.

motion of the aircraft in an Earth-fixed reference frame F_g and introduce the body reference frame F_b. Denoting the origin and the coordinate axes of F_g by (o_g, x_g, y_g, z_g), where $o_g x_g$ is chosen northward, $o_g y_g$ points eastward and $o_g z_g$ is directed to the ground. This frame is useful for describing the position and orientation of the UAV. F_b is a reference frame in which the origin is the mass center of the UAV and the axes direction is as shown in Figure 1. The equation of the UAV motion in the F_g is shown in (1):

$$\begin{cases} \dot{x} = v\cos\theta\cos\psi \\ \dot{y} = v\cos\theta\sin\psi \\ \dot{z} = -v\sin\theta \end{cases} \tag{1}$$

where $\dot{x}$, $\dot{y}$, and $\dot{z}$ represents the change rate of position on the axis of $o_g x_g$, $o_g y_g$, and $o_g z_g$ respectively. v represents the UAV's speed in the F_g reference system. θ and ψ represent pitch angle and yaw angle respectively.

It is assumed that there is no sideslip during the motion of the UAV, the engine thrust is forward along x-axis direction of the UAV, and the UAV is flying in a no-wind environment. The overload of the thrust and aerodynamic forces acting on the UAV in flight can be decomposed into tangential and normal overloads. The dynamic equations for the center of mass of the UAV are shown as (2).

$$\begin{cases} \dot{v} = g\,(n_x - \sin\theta) \\ \dot{\theta} = \dfrac{g}{v}\,(n_z\cos\phi - \cos\theta) \\ \dot{\psi} = \dfrac{g n_z \sin\phi}{v\cos\theta} \end{cases} \tag{2}$$

where n_x and n_z represent the tangential overload and normal overload of the UAV. g is the gravitational acceleration. $\dot{v}$, $\dot{\theta}$, and $\dot{\psi}$ denote acceleration, pitch rate, and yaw rate respectively.

B. ATTACK MODEL

We design a probability-based attack model that enables UAVs to lock, attack, and destroy targets using weapons. The UAV attack area is a cone in the nose direction, and the attack determination condition is that the target must be within the UAV attack area. Given that the target's position under the F_b reference system is $P\,(x, y, z)$, the attack azimuth angle ψ_P and attack elevation angle θ_P are shown in (3).

$$\begin{cases} \psi_P = \text{atan2}(P_y, P_x) \\ \theta_P = \text{atan2}(P_z, \sqrt{P_x^2 + P_y^2}) \end{cases} \tag{3}$$

The attack condition can be expressed as follows:

$$\begin{cases} |\psi_P| \le \psi_{firemax} \\ |\theta_P| \le \theta_{firemax} \\ \sqrt{P_x^2 + P_y^2 + P_z^2} \le \mathrm{D}_{firemax} \end{cases} \tag{4}$$

where $\psi_{firemax}$, $\theta_{firemax}$, and $D_{firemax}$ represent the maximum attack azimuth angle, elevation angle, and distance set by the simulation, respectively.

We use a hit noise cone based on Gaussian distribution to determine whether the attack hits or not, and the probability of hitting depends on the distance and the relative position between the UAV and the current target. By adding a Gaussian noise $\epsilon_{fire} \sim \mathrm{N}(0, 1)$ to the azimuth and pitch angles, as shown in (5), we can obtain the conditions for destroying the target.

$$\begin{cases} \left|\psi_P + \epsilon_{fire}\right| \le \pi \cdot \exp(-\dfrac{r}{D_{hit}}) \\ \left|\theta_P + \epsilon_{fire}\right| \le \pi \cdot \exp(-\dfrac{r}{D_{hit}}) \end{cases} \tag{5}$$

where $\mathrm{D_{hit}}$ is the effective attack distance.

III. METHODS

In this section, we propose a new multi-UAV collaborative decision-making method called Multi-agent Transformer-based Actor-Critic (MATAC) with the structure shown in Figure 2. The MATAC contains a centralized critic, multiple distributed actors, and a reply buffer. It learns the decision strategy through centralized training, while the execution is distributed. The training process is divided into two parts: data collection and algorithm update. The data collection process is represented by the blue connecting line, where each UAV generates data by interacting with the environment, which is analyzed and stored in the replay buffer. The winning data is copied to the won data set. When the data reaches a specified quantity the update of the UAV strategy starts, at this time a batch of data is taken out from the replay buffer to update the critic and actor respectively according to the update formula. The update process is represented by the green connecting line. When the algorithm is executed, each UAV obtains the current situation information based on its sensors and data link, where the data communication mode is broadcast. The process is the same as the data collection process, except that there is no data storage operation.

A. MARKOV GAMES

We focus on policy learning in fully cooperative multi-agent tasks, which can be modeled as decentralized partially observable Markov decision processes (Dec-POMDPs) [23].

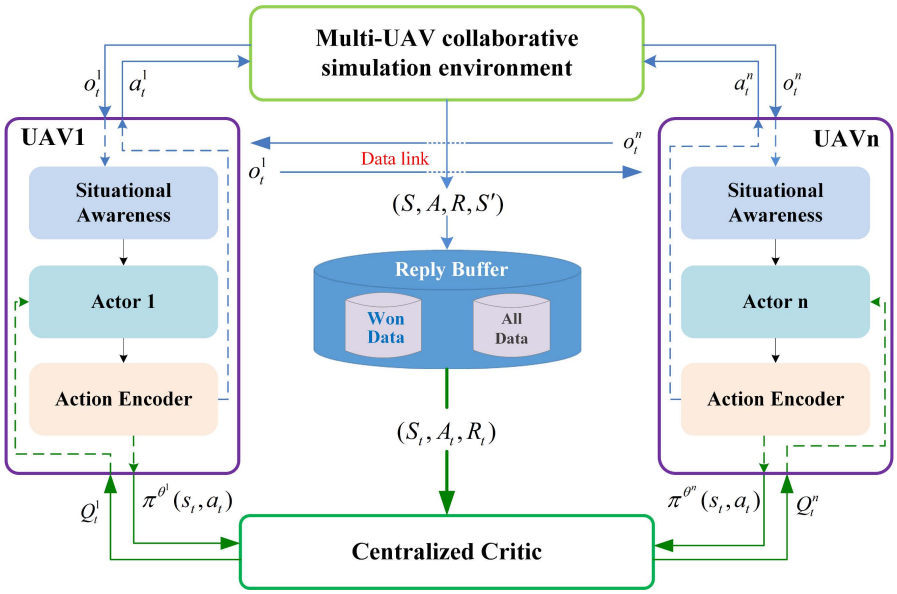


FIGURE 2. The MATAC algorithm architecture contains a multi-UAV co-simulation environment, a centralized critic, multiple distributed actors, and a reply buffer. The data collection process is shown by the blue connecting line, and the algorithm update process is shown by the green connecting line. Observation data can be shared between UAVs through a data link.

We consider an infinite-horizon Dec-POMDP G, defined by the tuple $G = (S, A, P, R, Z, O, N, \gamma)$, in which $s \in S$ describes the true state of the environment, $N \equiv 1, \ldots, n$ denotes the finite set of agents and $\gamma \in [0, 1)$ is the discount factor. The observation function $Z(s, i)$:$S \times N \rightarrow p(O)$ represents the probability distribution of the observations $o^i \in O^i$ for each agent $i \in N$. At each time step, each agent i selects its action $a_i \in A_i$ based on its local observation o^i according to its policy π_i:$O_i \times A_i \rightarrow [0, 1]$, and joint action $u \in A$ consisted of all the action a_i. The state transition function $P(s_{t+1} \mid s_t) : S \times A \times S \rightarrow [0, 1)$ represents the probability of the next state $s_{t+1} \in S$ given the current state $s_t \in S$ and joint action $u_t \in A$. The environment emits a bounded reward $R : S \times A \rightarrow [R_{min}, R_{max}]$ on each transition. Each agent learns a stochastic policy $\pi_i(a_i \mid O\tau_i)$ or a deterministic policy $u_i(\tau_i)$, conditioned only on its local action-observation history $\tau_i \in Z \times A$. The joint policy π induces a joint action-value function: $Q^{\boldsymbol{\pi}}(s_t, u_t) = \mathbb{E}_{s_{t+1:\infty}, U_{t+1:\infty}}[R_t \mid Os_t, u_t]$, where $R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$ is the discounted return. The advantage function is written as $A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s)$. In this paper, we study a full-cooperative environment where all agents have the same reward function, aiming to maximize the expected reward shown in (6).

$$\mathcal{J}(\pi) \triangleq \mathbb{E}_{a_t, s_t}[\sum\nolimits_t \gamma^t R(s_t, a_t)] \tag{6}$$

B. CENTRALIZED TRAINING AND DECENTRALIZED EXECUTION

Multi-intelligent reinforcement learning is divided into three architectures: centralized, distributed, and centralized training and decentralized execution (CTDE) [24], [25]. In distributed architectures, each agent is trained independently of the other agents and its policy network outputs the actions to be taken based on local observations. Each agent in the distributed architecture views the other agents as part of the environment, which causes the transfer function in the environment to change when the policies of the other agents changed. Hence the system is dynamic and non-Markovian. The centralized architecture can solve the non-stationarity of the environment by learning a joint policy of all the agents. Its input is the joint observation of all the agents and its output is the joint action of all the agents. However, it has the problem that the input and output space is huge and difficult to adapt to large-scale multi-agent systems.

To solve the above problems, the current dominant multi-agent reinforcement learning(MARL) methods adopt the framework of CTDE, such as MADDPG [26] and COMA [27]. The centralized manner is used during training, and after the training, the agent solely makes decisions based on its local observation utilizing the trained policy network. In CTDE, each agent during training maintains a centralized critic that takes the joint state-action as the input and outputs the estimation of the expected reward. Thus, the agent can obtain the other agents' information, which avoids the challenge of non-Markovian and non-stationary environments during training. Meanwhile, it learns a decentralized policy that only depends on the local state for execution. The policy does not require the information of other agents, which helps to mitigate the problems of the large-scale agents. This makes CTDE an effective framework for applying MARL and successful applications have been achieved in many real-world tasks. As a result, CTDE is an effective framework for multi-agent reinforcement learning and has been successfully applied in many practical tasks.

C. ATTENTION MECHANISMS

Attention mechanisms have become an integral part of neural network models that capture global dependencies. The operating principle of the attention mechanism is similar to a differentiable key-value memory model. Self-attention calculates the response of a specific position in the sequence by paying attention to all positions in this sequence [28]. Vaswani *et al.* designed a transformer model based on attention and self-attention mechanism, which achieves state-of-the-art performance in the field of natural language processing [29]. Attention has three matrices, K, Q, V representing a set of keys, queries and, values respectively. The attention is computed as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{7}$$

where d_k is a scaling factor equal to the dimension of the key.

The attention model has only one attention head which can't represent information of all positions. Compared with the attention model, multi-head attention allows the model to jointly focus on information from different representation subspaces from different positions. The multi-head attention is computed as follows:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1,\ldots,\text{head}_2)W^o,$$

where

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^k, VW_i^V) \tag{8}$$

where W^o, W^Q, W^K, and W^V are parameters of the Concat function, Q, K, and V respectively.

D. MULTI- ACTOR-ATTENTION-CRITIC

Multi-actor-attention-critic (MAAC) [30] applies an attention mechanism to the CTDE framework. The main idea of MAAC is to learn centralized critics with an attention mechanism that helps agents understand which other agents they should pay attention to. Attention critics can dynamically select which agents to attend to at each time during training, improving performance in multi-agent domains with complex interactions. In addition, MAAC has an input space linearly increasing with respect to the number of agents, it improves scalability by lowering the impact of increasing the number of agents.

In MAAC, all critics are updated by minimizing a joint regression loss function using off-policy temporal-difference learning. This joint regression loss function takes the following form:

$$\mathcal{L}_Q(\psi) = \sum_{i=1}^{N} \mathbb{E}_{(o,a,r,o')\sim D}\left[(Q_i^\psi(o,a) - y_i)^2\right],$$

where

$$y_i = r_i + \gamma \mathbb{E}_{a'\sim\pi_{\bar{\theta}}(o')}\left[Q_i^{\bar{\psi}}(o',a') - \alpha \log(\pi_{\bar{\theta}_i}(a_i'\,|o_i'))\right] \tag{9}$$

where ψ, $\bar{\psi}$, and $\bar{\theta}$ are the parameters of the critic, target critics, and target policies respectively. The temperature parameter α determines the balance between maximizing entropy and maximizing rewards. Q_i^ψ denotes the critic of the agent i which receives observations and actions for all agents.

The individual policies are updated by ascent with the following gradient:

$$\begin{aligned}&\nabla_{\theta_i} J(\pi_\theta)\\ &\quad= \mathbb{E}_{o\sim D, a\sim\pi}[\nabla_{\theta_i}\log(\pi_{\theta_i}(a_i\,|s_i))\\ &\qquad(-\alpha\log(\pi_{\theta_i}(a_i\,|o_i)) + Q_i^\psi(o,a) - b(o,a_{i^-}))]\end{aligned} \tag{10}$$

where θ is the parameters of the policy, $b(o,a)$ is the multi-agent baseline used to calculate the advantage function, the subscript i^- indicates all agents except agent i.

E. TRANSFORMER-BASED CENTRALIZED CRITIC FOR MULTI-AGENT REINFORCEMENT LEARNING

In this section, we propose an approach called Multi-agent Transformer-based Actor-Critic (MATAC) that uses a centralized critic and multiple distributed actors to learn the collaboration decision strategy of multi-UAV. We start by introducing the structure and the idea of learning about the transformer-based central critic. We then discuss our distributed actor and its learning principles. Finally, we describe the dual experience replay mechanism.

1) TRANSFORMER-BASED CENTRALIZED CRITIC

we design a centralized critic based on the transformer structure, which is shared by all agents. To obtain the Q-value function $Q_i^\psi(o,a)$ for the agent i, the critic receives the observations $o = (o_1,\ldots,o_n)$, and actions $a = (a_1,\ldots,a_n)$, for all agents indexed by $i \in \{1,\ldots,n\}$. Solving the Q-value function of the agent i can essentially be viewed as a regression problem. Devlin designed the BERT [31] based on the transformer-encoder, which has achieved state-of-the-art performance in the field of natural language processing and demonstrated the great potential of the transformer in regression problems. We designed the critic network structure based on the transformer-encoder, as shown in Figure 3. The centralized critic includes embedding, transformer-encoder, and multilayer perceptron (MLP). Among them, embedding and MLP are composed of forward fully connected layers. The transformer-encoder is composed of a stack of N identical layers, each of which has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple fully connected feed-forward network. We represent observation and action embedding of agent i as $e_i = f(o_i, a_i)$, and all agents except i as $e_j = f(o_j, a_j)$. We denote the embedding layers as R, where $R_i = \{(e)_I, e_j\}$.

$$\begin{aligned}Q_i, K_i, V_i &= F_{Q,K,V}(R_i)\\ C &= \text{Multihead}(Q_i, K_i, V_i)\end{aligned} \tag{11}$$

where the functions F are used to compute K, Q, and V. Then we use the function g to map the last transformer layer

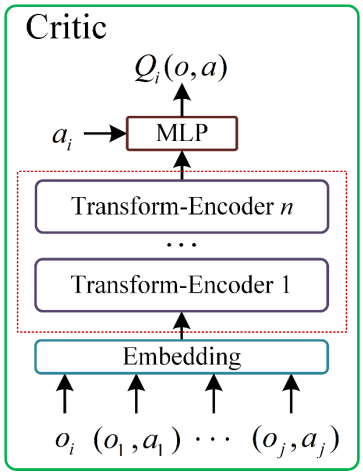


FIGURE 3. Calculating Q for each UAV i. Each UAV encodes its observations and actions that are sent to a transformer-encoder model, which then receives an evaluation of the current action a_i.

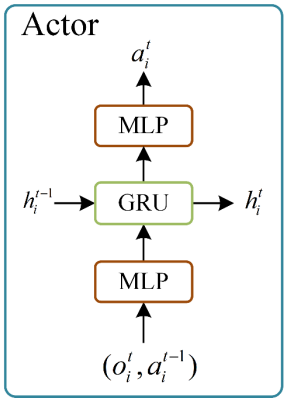


FIGURE 4. Calculating the action a_i for each UAV i. Each UAV sends to the actor for its observation and previous moment action, then receives the action that should be taken in the current situation.

$C = \{c_1, \ldots, c_n\}$ to the output space of the value function Q_i^{ψ}.

$$Q_i^{\psi}(o,a) = g(c_1) \tag{12}$$

To reduce the training cost of our algorithm, we refer to deep Q-learning (DQN) [32], where the action of agent i is removed from the input of critic, and output is a Q-value for each action of agent i. All critics are updated in the same way as MAAC, by minimizing a joint regression loss function, as shown in (9).

2) POLICY

The actor consists of a situational awareness layer, a decision layer, and an action coding layer. The decision layer consists of a GRU network, two MLPs, and a layer of SoftMax functions, denoted by π_θ, as shown in Figure 4. The decision network obtains s_t and outputs the action a_t in that state, which can be expressed as $\pi_{\theta^j}(a_t^j|s_t^j)$. The action coding layer converts the actions generated by the model into the actions required by the environment.

In this paper, we focus on the collaborative decision-making problem of homogeneous UAVs, so we use the shared intelligent body decision layer network parameters to avoid the emergence of lazy individuals in the collaborative game. The decision network that performs well in the test is selected as the seed network after every 100 updates, and its parameters are synchronized to other individuals. The Actor is updated by (13):

$$\nabla_{\theta_i} J(\pi_\theta) = \mathbb{E}_{o\sim D, a\sim\pi}[\nabla_{\theta_i} \log(\pi_{\theta_i}(a_i|o_i)) (-\alpha \log(\pi_{\theta_i}(a_i|o_i)) + A_i(o,a))], \tag{13}$$

where

$$A_i(o,a) = Q_i^{\psi}(o,a) - \sum_{a_i'\in A_i} \pi(a_i'|o_i) Q_i^{\bar{\psi}}(o,(a_i', a_{i^-}))$$

where the subscript i denotes the agent i and α is the temperature coefficient used to balance the maximum entropy and the payoff. Like MAAC, A_i denotes a advantage function using a baseline that can help solve the multi-agent credit allocation problem [30]. Instead of MAAC, we use the target critic network to compute the advantage function, which can increase the algorithm stability.

3) DOUBLE REPLAY BUFFER

The multi-UAV collaborative decision problem is a long-period continuous decision problem. Due to the high complexity of this problem, the environmental returns are sparse, thus making it difficult for the algorithm to obtain effective returns at the early stage of training which leads to poor convergence. Therefore, a dual experience replay mechanism is used in this paper to alleviate this problem. We design two experience pools, which are the won data experience pool B^w and the general experience pool B. Among them, the won data pool stores the experience data of winning trajectories during data collection, and the general experience pool stores all data. In the early stage of algorithm training, data are collected from the won data experience pool, and the weight of data collected from the won data experience pool is gradually reduced with the training process. The weighting relationship is shown in (14).

$$n = sw + s; \quad sw \in B^w, \ s \in B \tag{14}$$

where n represents the amount of data in the batch, sw, and s are given by (15).

$$sw = \begin{cases} 0; & \text{if } m \le n \\ n \cdot \exp\left(-\dfrac{4(x-b)}{a}\right); & \text{if } m > n \end{cases} \tag{15}$$

where m represents the number of trajectories in the priority experience pool, a represents the total sampling length during the training process, and b represents the number of generations sampled when the data in the won data pool is greater than n.

The MATAC algorithm operation flow is shown in Algorithm 1.

IV. EXPERIMENTS

To test our decision-making method, we designed a 4Vs4 multi-UAV coordinated air combat environment. Algorithm training was completed in this environment, and we also compared our algorithm with MAAC, COMA, and MADDPG for experiments, where we modify the MADDPG to accomm- odate the discrete action space. Finally, we analyze the adversarial strategies generated by our method.

Algorithm 1 Agent Training

Initialize environment with N agents
Initialize replay buffer, D and D^w
Initialize parameter vectors $\psi, \bar{\psi}, \theta, \bar{\theta}$
for episode $= 1, M$ **do**
  Reset environments, and get initial o_I for each agent i
  for $t = 1, T$ **do**
    Select action $a_i \sim \pi_i(\cdot \mid o_i)$ for each agent i
    Send actions to environments, get r and each agent observation o_i'
    Add step to buffer D
  end for
  if won **then**
    Store episode in D^w
  end if
  Sample minibatch B from $\langle D, D^w \rangle$
  $\psi \leftarrow \psi - \lambda_Q \nabla_\psi L_Q(\psi)$
  $\theta \leftarrow \theta - \lambda_\pi \nabla_\theta J(\pi_\theta)$
  $\bar{\psi} = \tau\bar{\psi} + (1-\tau)\psi$
  $\bar{\theta} = \tau\bar{\theta} + (1-\tau)\theta$
end for

A. EXPERIMENTAL SCENARIOS

We have set an engagement area with a diameter of 20 km, which contains the red side and the blue side. Both the red side and the blue side have four isomorphic UAVs, as shown in Figure 5. The UAVs on both sides have the same flight performance and attack capability, the difference is that strategy of the red side is generated by our decision algorithm and the blue side is a fixed rule strategy. Both sides aim to destroy the UAVs on the other side. Each UAV tries to launch an attack by taking advantage of a favorable attack position through maneuvering cooperation between teammates. The attack model then determines the attack conditions and calculates whether this attack destroys the opponent's UAV. In our algorithm training process, the UAVs of the red camp randomly appear in the left half of the engagement area, and the UAVs of the blue camp randomly appear in the right half of the engagement area. The environment parameters are set as shown in Table 1.

B. ACTION SPACE

The actions of UAVs can be divided into continuous and discrete actions, where continuous actions are mainly flight control-related parameters of UAVs, such as pitch control, roll control, and thrust control. Discrete actions are mainly intermittent control commands, such as firing weapons and switching radar. In the 4V4 adversarial environment, we only control the flight and attack of the UAV. We use an attack model to provide an attack strategy for the UAV and calculate whether this attack destroys the opponent's UAV. Meanwhile, we discretize the flight control quantity of UAV to form a discrete action space.

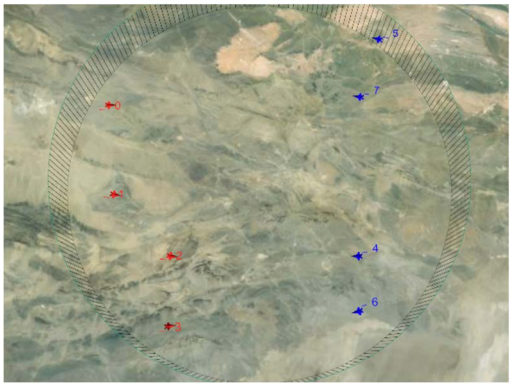


FIGURE 5. The 4Vs4 multi-UAV air combat simulation scenario.

TABLE 1. Environmental parameters.

Variable name	Value	Variable name	Value
ψ (rad)	$\left[-\frac{\pi}{3}, \frac{\pi}{3}\right]$	ψ_{firmax} (rad)	$\frac{\pi}{6}$
θ (rad)	$\left[-\frac{\pi}{3}, \frac{\pi}{3}\right]$	θ_{firmax}(rad)	$\frac{\pi}{6}$
v (m/s)	$[200, 240]$	D_{firmax}(km)	2
D_{hit}(km)	2		

The flight control quantities of the UAV are desired yaw angle ψ_e, desired pitch angle θ_e, and desired speed v_e, which are calculated as shown in (16). The $[\psi_e, \theta_e, v_e]$ is converted into the control quantities $[\phi, n_z, n_x]$ of the UAV model by the designed PID controller.

$$\begin{cases} \psi_e = \max(\psi_{min}, \min(\psi_{max}, \psi + \Delta\psi)) \\ \theta_e = \max(\theta_{min}, \min(\theta_{max}, \theta + \Delta\theta)) \\ v_e = \max(v_{min}, \min(v_{max}, v + \Delta v)) \end{cases} \tag{16}$$

where ψ_{max} and ψ_{min} are the maximum and minimum roll angles of the UAV, θ_{max} and θ_{min} are the maximum and minimum pitch angles of the UAV, and similarly v_{max} and v_{min} are the maximum and minimum speeds of the UAV, respectively. $\Delta\psi$, $\Delta\theta$, and Δv represent the increment of desired yaw, desired pitch, and desired speed respectively. The $[\Delta\psi, \Delta\theta, \Delta v]$ is discretized as shown in Table 2, where the discretized action is used as the output of the actor network.

C. OBSERVATION SPACE

To describe the dominance or disadvantage relationship between our UAV and the target UAV, we reference several

TABLE 2. Discretized action apace.

Variable name	Actions				
$\Delta\psi$ (rad)	$\frac{\pi}{3}$	$\frac{\pi}{6}$	0	$-\frac{\pi}{6}$	$-\frac{\pi}{3}$
$\Delta\theta$ (rad)	$\frac{\pi}{3}$	$\frac{\pi}{6}$	0	$-\frac{\pi}{6}$	$-\frac{\pi}{3}$
Δv (m/s)	100	50	0	-50	-100

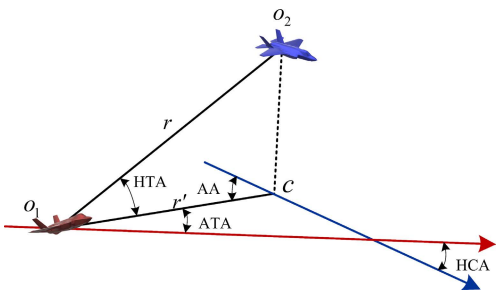


FIGURE 6. The geometric relationship between the aerial relative positions of the red UAV and the blue UAV.

definitions from the geometry of air combat [33]. These are aspect angle (AA), antenna train angle (ATA), horizontal crossing angle (HCA), height angle (HA), and distance of two UAVs r, which are shown in Figure 6. AA shows the position of the red with respect to the blue. During the battle, zero AA means that the red is right on the tail of the blue. In other cases when the AA is 90° or −90°, the location of the blue is right-wing or left-wing side of the red. On the other hand, ATA means the angle between the heading of the red UAV and the line connecting the positions of the red and blue UAVs projected in the horizontal plane. Finally, HA indicates the angle between the line of the red and blue UAV positions and the horizontal plane. To attain air superiority, AA, ATA, and HTA should all be minimized to zero in the ideal case. Zero AA, ATA, and HTA mean that the red UAV is chasing the blue UAV aircraft from the tail.

In our simulation environment, the red UAV can detect all the blue UAVs and red UAVs through radar, as well as obtain information about the red UAVs' actions through communication. Similarly, the blue UAVs can also obtain information about the red UAVs. We refer to the radar and communication of the red UAV as the situational awareness module.

Hence, the observation space will contain all the information of both sides' UAVs, and the observation function is defined as shown in (17).

$$s_i = \left\{ \begin{array}{l} (p_e, v, \phi, \psi, \theta)_i \\ (p_b, v, AA_b, ATA_b, HA_b)_{i^-} \\ (p_b, v, AA_b, ATA_b, HA_b)_j \end{array} \right\} \tag{17}$$

where the subscript i denotes UAV i on the red side, subscript i^- denotes all UAVs on the red side except UAV i, and subscript j denotes all UAVs on the blue side. S_I represents

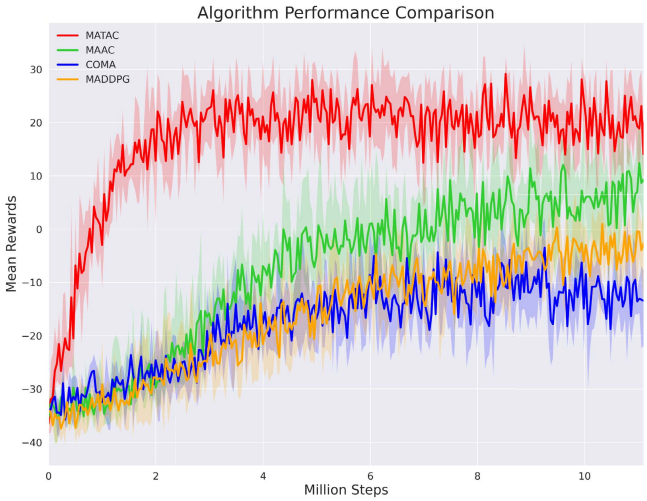


FIGURE 7. Algorithm performance comparison: the red, green, blue and yellow lines indicate the average scores of MATAC, MAAC, COMA and MADDPG algorithms during the training process, respectively.

the observation of red UAV i. p_e is the position of UAV i in the reference system F_g. p_b denotes the position relative to UAV i under the UAV i body reference system F_b. AA_b, ATA_b, HA_b denote AA, ATA, HA respectively between the red UAV i and the blue UAV.

D. REWARDS FUNCTION

In reinforcement learning, the agent optimizes intending to obtain the maximum reward, so the design of the reward function should reflect the optimization direction we expect for the agent. In this paper we want the red camp to eliminate all UAVs in the blue camp with minimum loss. We consider numerous principles to achieve this purpose. numerous principles to achieve this purpose.

First of all, the reward function should have the global maximum value in the reward space. The reward function should take the maximum value when the red has no losses and the blue is destroyed. Otherwise, the red will not optimize its strategy towards eliminating the blue while reducing its losses.

Secondly, the reward function should be continuous in the reward space. Actor and critic network optimization is achieved by continuously exploring the state-action space. Additional exploration of the state-action space is required if the reward values do not converge to the target state. Therefore, a small number of rewards are used to guide the strategy to converge to the target strategy to improve the convergence speed of the training process. Nevertheless, the proportion of guidance rewards in the reward function should be carefully determined. Since, if the guidance reward is large enough to drown out the target strategy reward, then the agent will collect the guidance reward instead of collecting the target strategy reward in the long run. Therefore, the size of the guidance reward should only be sufficient to guide the agent to achieve the basic guidance purpose.

Finally, the reward function should have a penalty term. If the agent takes an impermissible or unrealistic action, the

reward for that action should be the opposite of the expected value of the current strategy for that action. The actual reward should be negative, for instance, when the red side is shot down in combat. The red UAV will reduce the probability of taking that action in that state by changing the expectation of the corresponding state-action.

Consequently, our reward function is divided into three parts.

1) If a red UAV i shoots down a blue UAV, it scores $+10$ points and the blue UAV is removed from the simulation. Conversely, if the red UAV i is shot down by a Blue UAV, it scores -10 points and UAV i is removed from the simulation.
2) To avoid UAVs flying out of the battlefield area, a penalty is applied to this behavior. If a UAV flies out of the flight airspace, it scores -10 points and is removed from the simulation.
3) To avoid too slow training convergence caused by sparse reward and also to guide the red UAVs to learn to attack, we set an angle reward. When a red UAV i puts a blue UAV within attack range by maneuvering, it scores $+0.01$ points. Conversely, if the red UAV i was placed within the attack range of the blue UAV it scored -0.01 points.

E. STRATEGY OF THE BLUE CAMP

We design a simple but effective attack strategy for the blue UAV. Briefly, each blue UAV will select the nearest red UAV as its target, solve the attitude angle and speed expectation from the target position and speed information, and start the attack when the firing conditions are available. The blue UAV will continuously detect the change in the red UAV's posture and automatically switch the target to the current closest target when the closest target changes. This strategy effectively concentrates the Blue's firepower advantage, especially when the Red UAV is alone, and ensures a local advantage.

F. AGENT TRAINING AND TESTING

In the MATAC, the number of hidden units per layer in the actor is 256. In the critic, the transformer encoder parameter embedding is set to 256, the attention head is set to 2, and the depth is set to 2. Both of them use Adam as the optimizer, where the learning rate of the actor is 0.0003 and the critic is 0.0001. The discount factor γ is set to 0.99, the size of the replay buffer is 10^6, the training batch size is 32, the temperature parameter α is 1/15, the target smoothing coefficient τ is 0.001 and the ReLU is used as the activation function. The CPU of the algorithm training platform is AMD Ryzen 9 5950X and the GPU is Nvidia GTX 2080s.

In the training, a total of 11 million steps of air combat simulation data were collected. As shown in the Figure 7, our algorithm trends upward for the first two million steps after which it gradually converges. At one million steps, the average reward reaches near 0, which indicating that the red side already have the ability to fight with the blue side. It converges at three million steps, and the final score is stable at about 20. We also trained the MAAC, COMA, and MADDPG algorithm, as shown in Figure 7. The score of MAAC algorithm continues to increase in the first 4.5 million steps. In the middle of 4.5-7 million steps, the score is stable. The final average score is around 5. The average reward of COMA converges after 4 million steps, with the final reward fluctuating around -13. The reward of the MADDPG continues to grow between 0 and 6 million steps, rising to near -10 at 6 million steps. After that it slowly rises and finally fluctuates around -5. It is show that the data utilization of the MATAC algorithm is very efficient.

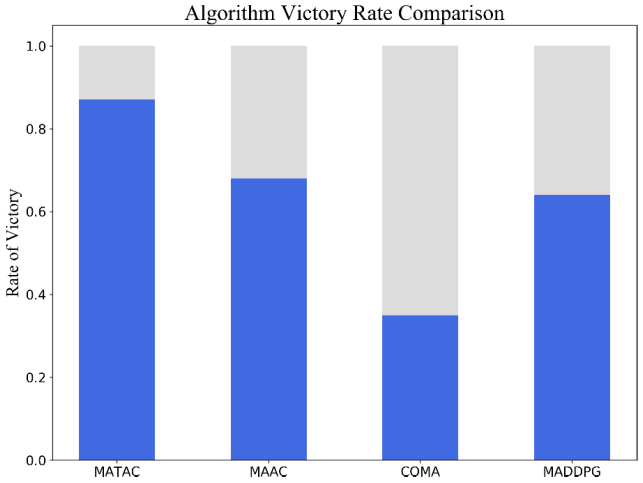


FIGURE 8. The average victory rate of MATAC, MAAC, COMA and MADDPG was obtained after 300 tests.

We also tested the strategies obtained from the training. The strategies of MATAC, MAAC, COMA, and MADDPG algorithm were tested simultaneously for 300 respectively. The actual performance of those strategies is obtained by counting the rate of victory in the test. Each round in this test has different random seeds to ensure the randomness of the test. When the test, we obtained the winning rate of each algorithm, as shown in the Figure 8. It can be seen that the MATAC algorithm has a win rate of 87%, MAAC has a win rate of 68%, COMA has a win rate of 35%, and MADDPG has a win rate of 63%. Therefore, MATAC, MAAC, and MADDPG algorithms can obtain effective strategies.

G. STRATEGIES ANALYSIS

We replay the obtained strategies and could find that MATAC was able to learn several effective strategies: pursuit strategy, maneuver strategy, and baiting strategy.

1) Maneuver strategy: when the red UAV is being chased by the blue UAV during the combat, the red UAV should realize that it is being chased and maybe at a disadvantage position in the next moment. At this point, the red UAV should take timely maneuvering strategies to avoid the blue UAV's pursuit or even occupy a favorable attack position through maneuvering. Figure 9 shows that the red drone is able to master the maneuvering strategy.

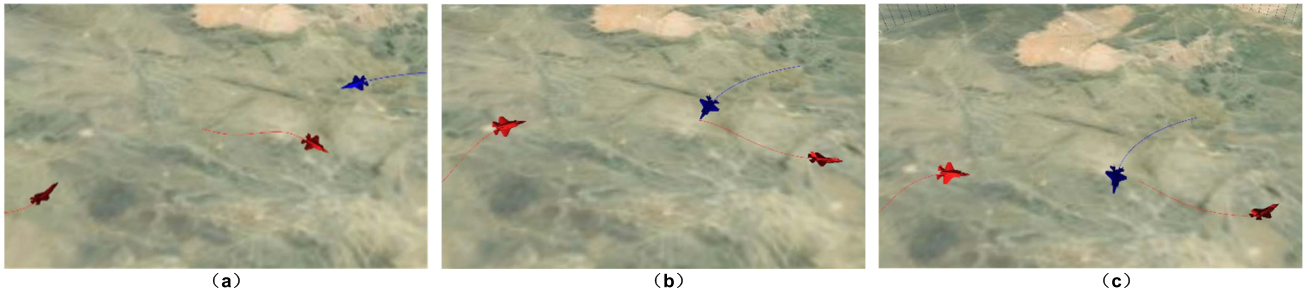


FIGURE 9. When the red UAV in the center of (a) senses that it is being chased by the blue UAV, it breaks away from the blue UAV chase by fast maneuvering, while making its teammates occupy a superior position relative to the blue UAV, as shown in (b) and (c).

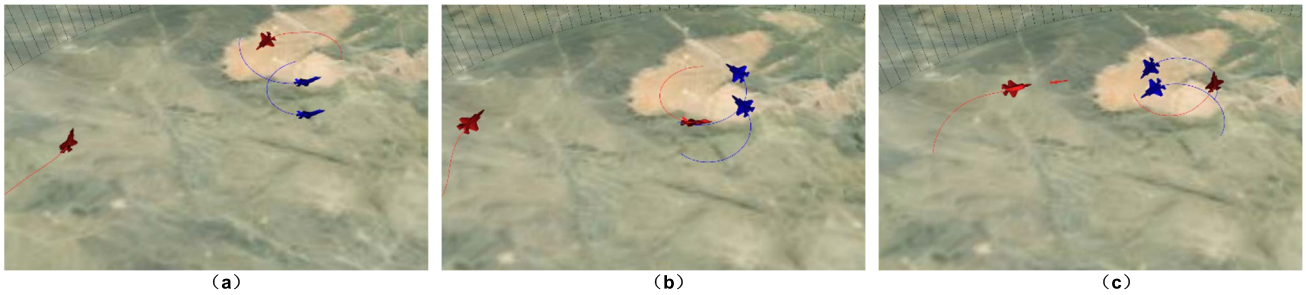


FIGURE 10. The red UAVs choose to pursue the blue UAVs based on the acquired position of the blue UAV. The red UAV on the left in (a) gets the position of the blue UAVs, so it turns its nose direction to maneuver towards the blue UAV's location, as shown in (b).

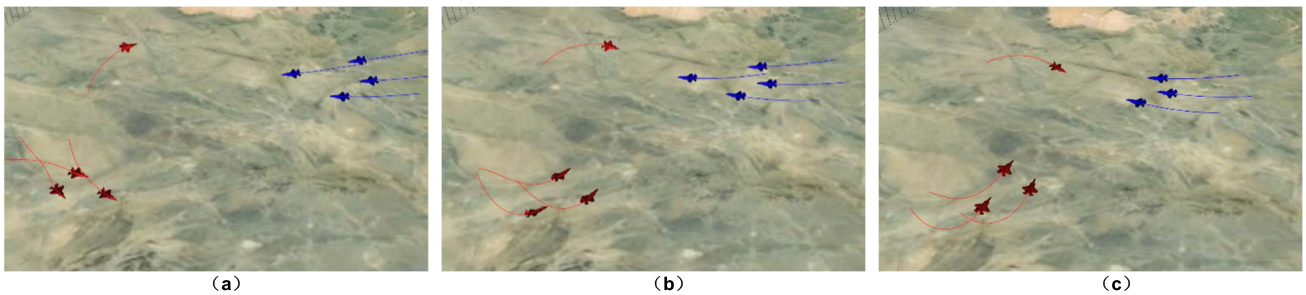


FIGURE 11. From (a) to (c) shows a red UAV deliberately guiding a blue UAV to attack itself, thus allowing its teammate to take a favorable attack position.

2) Pursuit strategy: the red side UAV must first learn to track the blue side UAVs. Only when it can continuously track the blue UAV can it have the opportunity to occupy the attack position to launch an attack. From Figure 10, we can see that the red UAVs have been able to track the blue UAVs quickly.
3) Baiting strategy: By one red UAV attracting the attention of the blue UAVs, other red UAVs take advantage of the opportunity to occupy a favorable attack position. Shown in Figure 11, a red UAV rapidly approaches the blue formation from the right flank of the blue UAVs, guiding the blue UAVs to follow that red UAV. The other red UAVs maneuver to occupy a favorable attack position on the left flank. This strategy can effectively eliminate the blue UAVs while minimizing their losses. At the same time, this strategy demonstrates that MATAC can control multiple aircraft to explore and enable complex warfare.

V. CONCLUSION

To solve the multi-UAV collaborative decision-making problem, we designed a multi-agent reinforcement learning method MATAC based on a multi-agent Markov decision-making architecture with centralized training distributed execution, which has a centralized critic incorporated into Transformer and multiple GRU-based actors. We build a multi-UAV collaboration simulation environment and develop a multi-UAV collaborative air combat scenario in this environment. We performed algorithm training in this scenario and compared the results with MAAC, COMA, and MADDPG. The results show that both MATAC, MAAC, and MADDPG can learn effective air combat strategies and their performance is better than that of COMA. The convergence of MATAC is faster and more effective than MAAC, MADDPG, and COMA. However, we have only studied the collaborative decision problem in discrete action space, in the next step, we hope to apply MATAC to the collaboration

decision problem of multiple UAVs in continuous action space.

REFERENCES

[1] K. Fregene, "Unmanned aerial vehicles and control," *IEEE Control Syst. Mag.*, vol. 32, no. 5, pp. 32–34, Oct. 2012, doi: 10.1109/mcs.2012.2205474.

[2] V. Roberge, M. Tarbouchi, and G. Labonté, "Fast genetic algorithm path planner for fixed-wing military UAV using GPU," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 54, no. 5, pp. 2105–2117, Oct. 2018, doi: 10.1109/TAES.2018.2807558.

[3] S.-J. Chung, A. A. Paranjape, P. Dames, S. Shen, and V. Kumar, "A survey on aerial swarm robotics," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 837–855, Aug. 2018, doi: 10.1109/tro.2018.2857475.

[4] Z. Yao, M. Li, Z. Chen, and R. Zhou, "Mission decision-making method of multi-aircraft cooperatively attacking multi-target based on game theoretic framework," *Chin. J. Aeronaut.*, vol. 29, no. 6, pp. 1685–1694, Dec. 2016, doi: 10.1016/j.cja.2016.09.006.

[5] Y. Zhou, B. Rao, and W. Wang, "UAV swarm intelligence: Recent advances and future trends," *IEEE Access*, vol. 8, pp. 183856–183878, 2020.

[6] P. Ogren and M. Winstrand, "Combining path planning and target assignment to minimize risk in SEAD missions," in *Proc. AIAA Guid., Navigat., Control Conf. Exhibit*, Aug. 2005, p. 5865.

[7] Z. Zhen, D. Xing, and G. Chen, "Cooperative search-attack mission planning for multi-UAV based on intelligent self-organized algorithm," *Aerosp. Sci. Technol.*, vol. 76, pp. 402–411, May 2018, doi: 10.1016/j.ast.2018.01.035.

[8] J. Capitan, L. Merino, and A. Ollero, "Cooperative decision-making under uncertainties for multi-target surveillance with multiples UAVs," *J. Intell. Robotic Syst.*, vol. 84, nos. 1–4, pp. 371–386, Dec. 2016, doi: 10.1007/s10846-015-0269-0.

[9] L. Huang, H. Qu, and L. Zuo, "Multi-type UAVs cooperative task allocation under resource constraints," *IEEE Access*, vol. 6, pp. 17841–17850, 2018.

[10] X. Bai, P. Wang, Z. Wang, and L. Zhang, "3D multi-UAV collaboration based on the hybrid algorithm of artificial bee colony and A," in *Proc. Chin. Control Conf. (CCC)*, Jul. 2019, pp. 3982–3987, doi: 10.23919/ChiCC.2019.8866197.

[11] J. Xu, Q. Guo, and Z. Li, "Dynamic selection method for cooperative decision-making center of multi-UAV system based on cloud trust model," in *Proc. IEEE 3rd Adv. Inf. Technol., Electron. Autom. Control Conf. (IAEAC)*, Oct. 2018, pp. 922–926.

[12] Y. Ma, G. Wang, X. Hu, H. Luo, and X. Lei, "Cooperative occupancy decision making of multi-UAV in beyond-visual-range air combat: A game theory approach," *IEEE Access*, vol. 8, pp. 11624–11634, 2020, doi: 10.1109/ACCESS.2019.2933022.

[13] O. Ilaya, C. Bil, and M. Evans, "Distributed and cooperative decision making for multi-UAV systems with applications to collaborative electronic warfare," in *Proc. 7th AIAA ATIO Conf, 2nd CEIAT Int. Conf. Innov. Integr. Aero Sci., 17th LTA Syst. Tech. Conf., Followed 2nd TEOS Forum*, Sep. 2007, p. 7885.

[14] Y. Zhao, X. Wang, C. Wang, Y. Cong, and L. Shen, "Systemic design of distributed multi-UAV cooperative decision-making for multi-target tracking," *Auto. Agents Multi-Agent Syst.*, vol. 33, nos. 1–2, pp. 132–158, Mar. 2019, doi: 10.1007/s10458-019-09401-5.

[15] D. Cao, W. Hu, J. Zhao, Q. Huang, Z. Chen, and F. Blaabjerg, "A multi-agent deep reinforcement learning based voltage regulation using coordinated PV inverters," *IEEE Trans. Power Syst.*, vol. 35, no. 5, pp. 4120–4123, Sep. 2020, doi: 10.1109/TPWRS.2020.3000652.

[16] T. Chu, J. Wang, L. Codecà, and Z. Li, "Multi-agent deep reinforcement learning for large-scale traffic signal control," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 3, pp. 1086–1095, Mar. 2020, doi: 10.1109/tits.2019.2901791.

[17] C. Xu, S. Liu, C. Zhang, Y. Huang, Z. Lu, and L. Yang, "Multi-agent reinforcement learning based distributed transmission in collaborative cloud-edge systems," *IEEE Trans. Veh. Technol.*, vol. 70, no. 2, pp. 1658–1672, Feb. 2021, doi: 10.1109/TVT.2021.3055511.

[18] Z. Sun, H. Piao, Z. Yang, Y. Zhao, G. Zhan, D. Zhou, G. Meng, H. Chen, X. Chen, B. Qu, and Y. Lu, "Multi-agent hierarchical policy gradient for air combat tactics emergence via self-play," (in English) *Eng. Appl. Artif. Intell.*, vol. 98, Feb. 2021, Art. no. 104112, doi: 10.1016/j.engappai.2020.104112.

[19] J. Yang, X. You, G. Wu, M. M. Hassan, A. Almogren, and J. Guna, "Application of reinforcement learning in UAV cluster task scheduling," *Future Gener. Comput. Syst.*, vol. 95, no. 1, pp. 140–148, 2019, doi: 10.1016/j.future.2018.11.014.

[20] W. Zhao, H. Chu, X. Miao, L. Guo, H. Shen, C. Zhu, F. Zhang, and D. Liang, "Research on the multiagent joint proximal policy optimization algorithm controlling cooperative fixed-wing UAV obstacle avoidance," *Sensors*, vol. 20, no. 16, p. 4546, Aug. 2020.

[21] X. Wei, L. Yang, G. Cao, T. Lu, and B. Wang, "Recurrent MADDPG for object detection and assignment in combat tasks," *IEEE Access*, vol. 8, pp. 163334–163343, 2020, doi: 10.1109/Access.2020.3022638.

[22] 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," *IEEE Access*, vol. 7, pp. 146264–146272, 2019, doi: 10.1109/access.2019.2943253.

[23] F. A. Oliehoek and C. Amato, *A Concise Introduction to Decentralized POMDPs*. Berlin, Germany: Springer, 2016.

[24] F. A. Oliehoek, M. T. J. Spaan, and N. Vlassis, "Optimal and approximate Q-value functions for decentralized POMDPs," *J. Artif. Intell. Res.*, vol. 32, no. 1, pp. 289–353, 2008, doi: 10.1613/jair.2447.

[25] L. Kraemer and B. Banerjee, "Multi-agent reinforcement learning as a rehearsal for decentralized planning," *Neurocomputing*, vol. 190, pp. 82–94, May 2016, doi: 10.1016/j.neucom.2016.01.031.

[26] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–12.

[27] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proc. AAAI Conf. Artif. Intell.*, 2018, vol. 32, no. 1, pp. 1–9.

[28] A. P. Parikh, O. Täckström, D. Das, and J. Uszkoreit, "A decomposable attention model for natural language inference," 2016, *arXiv:1606.01933*.

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *Presented at the 31st Int. Conf. Neural Inf. Process. Syst.*, Long Beach, CA, USA, 2017.

[30] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2961–2970.

[31] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

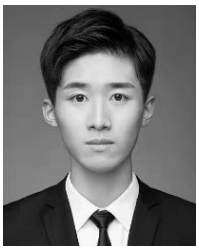
[32] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, and S. Petersen, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015, doi: 10.1038/nature14236.

[33] P. Bonanni, *The Art of the Kill*. Alameda, CA, USA: Spectrum HoloByte, 1993.

SHAOWEI LI was born in Hebei, China, in 1991. He is currently pursuing the Ph.D. degree in aircraft design with Beihang University, Beijing, China. His research interests include multi-UAVs collaborative decision making, mission planning, and multi-agent reinforcement learning.

YUHONG JIA received the Ph.D. degree in aircraft design from Beihang University. She is currently a Professor and a Ph.D. Supervisor at Beihang University. Her research interests include aircraft design, mechanical design, aircraft landing gear design, and active control technology.

FAN YANG was born in 1997. He is currently pursuing the master's degree in aircraft design with Beihang University. His current research interests include intelligent air combat and deep reinforcement learning.



QINGYANG QIN was born in Shandong, Liaocheng, China, in 1998. He is currently pursuing the master's degree with Beihang University. His research interests include intelligent decision, intelligent air combat, and reinforcement learning.

HUI GAO was born in Henan, China, in 1995. He is currently pursuing the Ph.D. degree in aircraft design with Beihang University. His research interests include dynamic obstacle avoidance and intelligent path planning for UAVs.

YAOMING ZHOU is currently an Associate Professor and a Ph.D. Supervisor at Beihang University. His research interests include intelligent decision-making of UAV, UAV path planning, and intelligent UAV design.