MMOTS: A Multi-UAV Pursuit-Evasion Game Training Strategy Relying on Offline Reinforcement Learning

Zekai Zhang[1,*], Xiangjin Li[1,*], Ziyu Chen[2,*], Kangxin Hu[3], Miao Peng[4], Jingjing Wang[5(✉)], and Yong Ren[6]

[1] Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen, China
[2] College of Biosystems Engineering and Food Science, Zhejiang University, Hangzhou, China
[3] College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou, China
[4] Ocean College, Zhejiang University, Hangzhou, China
[5] School of Cyber Science and Technology, Beihang University, Beijing, China
drwangjj@buaa.edu.cn
[6] Department of Electronic Engineering, Tsinghua University, Beijing, China

Abstract. Multiple unmanned aerial vehicles (UAVs) pursuit-evasion game is a prominent approach for achieving air superiority. In this paper, a multi-agent independent soft actor-critic (MAISAC) and multi-agent independent decision transformer (MAIDT)-based offline reinforcement learning training strategy (MMOTS) is proposed to train multiple UAVs to complete the pursuit-evasion game, encompassing a two-stage design. In the first stage, we develop the MAISAC algorithm to facilitate policy improvement and offline dataset generation. In the second stage, MAIDT is proposed to realize the model training and pursuit-evasion tasks. Finally, simulation results demonstrate that the proposed MMOTS can achieve pursuit success rate exceeding 70% when the target distance ranges from 6 to 8, and the number of pursuit UAVs ranges from two to four. These outcomes underscore the outstanding generality, scalability and performance of MMOTS in the context of multi-UAV pursuit-evasion games. To accelerate relevant research in this direction, the code for simulation will be released as open-source.

Keywords: Multiple unmanned aerial vehicles· Offline reinforcement learning· Decision transformer · Pursuit-evasion game.

1 Introduction

The outstanding maneuverability of unmanned aerial vehicles (UAVs) has garnered increasing attention from researchers, and they have been widely used in

* These authors contributed equally to this work.
✉ Corresponding author.

various fields such as aerial surveillance [4], environmental management, transportation and search [10] and especially multi-UAV pursuit-evasion game [7]. A multi-UAV pursuit-evasion game involves the coordinated use of multiple UAVs to swiftly capture a single evading UAV, aiming to minimize the time required for capture. This approach serves as a primary method to achieve air superiority. Nonetheless, it frequently encounters unique challenges such as high test cost [3] and real-time obstacle avoidance difficulties [1]. These challenges significantly impede the advancement of multi-UAV pursuit-evasion game.

Traditionally, pursuit-evasion strategies for multiple UAVs have been relying based on mathematical models [6] or by employing rule-based or centralized control algorithms [2]. However, the performance of the strategies based on mathematical model often deteriorates when the optimal parameters of the controller change due to the change of the environment model. And rule-based or centralized control algorithms are limited by the inability to know the future maneuvering information and behavioral strategies of non-cooperative targets, which leads to limited scalability and adaptability.

In recent years, the emergence of reinforcement learning (RL) provides a feasible solution for multi-UAV pursuit-evasion game. Combined with RL, UAVs can acquire control strategies autonomously in the environment. This significant advancement enhances the efficiency of cooperative operations while simultaneously reducing the associated task execution risks [15, 13, 16]. However, in the process of exploration, researchers are aware of two main challenges: (1) Existing multi-agent reinforcement learning (MARL) methods are commonly based on centralized training with decentralized execution (CTDE), which makes the algorithm less extensible. (2) RL methods need frequent interactions between the agents and the environment, which costs a great deal of time and computing resources [11, 18]. The above challenges greatly limit the application of RL in the multi-UAV pursuit-evasion game.

Fortunately, the emergence of offline RL effectively addresses this issue. It leverages pre-exising offline dataset for training, enhancing training efficiency while conserving computational resources and time [12, 17]. However, the traditional algorithms rely on the time difference algorithm, which introduces three fatal elements (bootstrap, off-policy and approximation), all of which can compromise training stability. Decision transformer (DT) is proposed to introduce the transformer into offline RL for the first time, and it effectively avoids this series of shortcomings [5]. But it is difficult to extend it to multi-agent systems, and the existing simulation environment lacks adequate data collection capabilities and lacks high-precision simulation capabilities for multi-UAV pursuit-evasion game [14].

Based on the above analysis, in this work we put forward multi-agent independent soft actor-critic (MAISAC) and multi-agent independent decision transformer (MAIDT)-based offline RL training strategy (MMOTS) to train multiple UAVs to successfully complete the pursuit-evasion game, improving the scalability, generalization capabilities and training efficiency. Additionally, we develop a high-precision simulation environment based on robotic operating

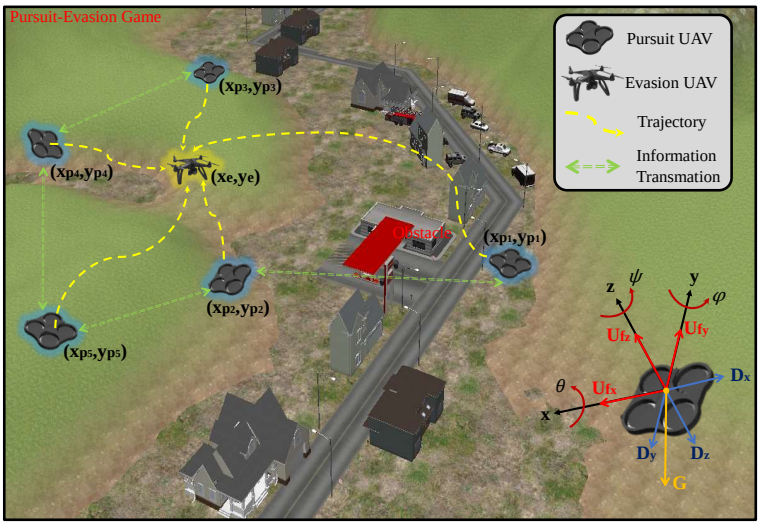

Fig. 1. The diagram of multi-UAV pursuit-evasion game and its system and motion model.

system (ROS), and conduct simulation experiments to verify the excellent performance of MMOTS. To accelerate relevant research in this direction, the code for simulation will be released as open-source.

2 Model, Environment and Problem Formulation

In this section, we first briefly present the system and motion model in Section 2.1 to better understand how multi-UAV pursuit-evasion works. Next, we provide a specific description of the integrated simulation environment in Section 2.2 to explain how it works. Finally, problem formulation is introduced in Section 2.3 to describe the optimization objectives and constraints.

2.1 System and Motion Model

In this paper, we categorize UAVs into two distinct types, i.e. the pursuit UAVs (PUAVs), and the evasion UAVs (EUAVs). PUAVs are located at $\mathbf{p}_{p_i}(t) = [x_{p_i}(t), y_{p_i}(t), z_{p_i}(t)]^T \in \mathbb{R}^3 (i = 1, 2, \dots n)$ pursuing EUAVs located at $\mathbf{p}_e(t) = [x_{e_i}(t), y_{e_i}(t), z_{e_i}(t)]^T \in \mathbb{R}^3 (i = 1, 2, \dots n)$. Assuming all UAVs fly at a fixed altitude, and the position of the EUAV can be captured by the radar installed on the PUAVs.

Generally, the UAV i is idealized as a 6-DoF rigid body, whose sketch is shown in Fig. 1. The inertial system is established by taking the centroid of the UAV as the origin of coordinates. And the forces and moments on the UAV are mainly composed of gravity $\boldsymbol{G}_i \in \mathbb{R}^3$, air drag $\boldsymbol{D}_i \in \mathbb{R}^3$ in the inertial frame, and

total lifting force $\boldsymbol{U}_f \in \mathbb{R}^3$ generated by motors in the body frame. Therefore, the dynamic equation of the UAV i can be written as follows

$$\boldsymbol{\mathcal{T}}_i(t)\boldsymbol{U}_f(t) + \boldsymbol{G}_i + \boldsymbol{D}_i(t) = m_i \boldsymbol{A}_i(t), \tag{1}$$

$$\boldsymbol{\mathcal{M}}_i(t) = \boldsymbol{I}_i \dot{\boldsymbol{\omega}}_i(t) + \boldsymbol{\omega}_i(t) \times (\boldsymbol{I}_i \cdot \boldsymbol{\omega}_i(t)), \tag{2}$$

where $\boldsymbol{\mathcal{T}}_i(t) \in \mathbb{R}^{3\times3}$ is the rotation matrix of each UAV, which can be expressed as

$$\boldsymbol{\mathcal{T}}_i(t) = \begin{bmatrix} c_{\theta_i}c_{\psi_i} & -c_{\varphi_i}s_{\psi_i} + s_{\varphi_i}s_{\theta_i}c_{\psi_i} & s_{\varphi_i}c_{\psi_i} + c_{\varphi_i}s_{\theta_i}c_{\psi_i} \\ c_{\theta_i}s_{\psi_i} & c_{\varphi_i}c_{\psi_i} + s_{\varphi_i}s_{\theta_i}c_{\psi_i} & -s_{\varphi_i}c_{\psi_i} + c_{\varphi_i}s_{\theta_i}s_{\psi_i} \\ -s_{\theta_i} & c_{\theta_i}s_{\varphi_i} & c_{\theta_i}c_{\varphi_i} \end{bmatrix}, \tag{3}$$

where $\mathbf{c}_i = \left[c_{\theta_i}, c_{\psi_i}, c_{\varphi_i}\right]^T = \left[\cos\theta_i, \cos\psi_i, \cos\varphi_i\right]^T \in \mathbb{R}^3$, $\boldsymbol{s}_i = \left[s_{\theta_i}, s_{\psi_i}, s_{\varphi_i}\right]^T = \left[\sin\theta_i, \sin\psi_i, \sin\varphi_i\right]^T \in \mathbb{R}^3$, and θ, ψ, φ stands for pitch angle, yaw angle, roll angle of the UAV respectively, and they change over time t. $\boldsymbol{A}_i(t) \in \mathbb{R}^3$ stands for the acceleration on each UAV, m_i represents the mass. $\boldsymbol{\mathcal{M}}_i(t) \in \mathbb{R}^3$, $\boldsymbol{I}_i \in \mathbb{R}^{3\times3}$, $\boldsymbol{\omega}_i(t) \in \mathbb{R}^3$ stands for the total moments, inertia matrix, angular velocity of the UAV respectively.

2.2 Integrated Simulation Environment

Based on ROS, in this paper we build up an integrated simulation environment, which is dedicated to the multi-UAV pursuit-evasion game and offline RL. We start by using the scenario that has been built by other researchers. Then we import the UAV models into Gazebo, and write offline RL framework based on Pytorch. During the simulation, the offline RL framework first sends the control signals, which are transmitted to Gazebo through the nodes in ROS to realize the control of UAVs. Meanwhile, the interaction information between UAVs and the environment in Gazebo will be returned to the offline RL framework in turn and saved. The details and principles are shown in Fig. 2.

2.3 Problem Formulation

This paper aims to train multiple UAVs to complete pursuit-evasion game using MMOTS, so we can use Markov game process (MGP) to model the pursuit-evasion game. Since multiple UAVs constitute a multi-agent system and their behavior depends only on the current state, the interaction between them and the environment can be modeled as a MGP, which can be expressed as a $(2N+5)$ tuple

$$\mho = (N, \boldsymbol{S}_{p_1}, \cdots, \boldsymbol{S}_{p_n}, \boldsymbol{S}_{e_1}, \cdots, \boldsymbol{S}_{e_n}, \boldsymbol{\mathcal{A}}_{p_1}, \cdots, \boldsymbol{\mathcal{A}}_{p_n}, \boldsymbol{\mathcal{A}}_{e_1}, \cdots, \boldsymbol{\mathcal{A}}_{e_n}, \mathcal{R}_p, \mathcal{R}_e, \mathcal{P}, \gamma), \tag{4}$$

where N is the total number of UAVs. $\boldsymbol{S}_{p_1}, \cdots, \boldsymbol{S}_{p_n}, \boldsymbol{S}_{e_1}, \cdots, \boldsymbol{S}_{e_n}, \boldsymbol{\mathcal{A}}_{p_1}, \cdots, \boldsymbol{\mathcal{A}}_{p_n}$, $\boldsymbol{\mathcal{A}}_{e_1}, \cdots, \boldsymbol{\mathcal{A}}_{e_n}$, $\mathcal{R}_p$ and $\mathcal{R}_e$ represent the state space, the action space and reward function of the PUAVs and EUAVs, respectively. $\mathcal{P} \in (0, 1)$ is state transition function, while $\gamma \in (0, 1]$ stands for the discount factor.

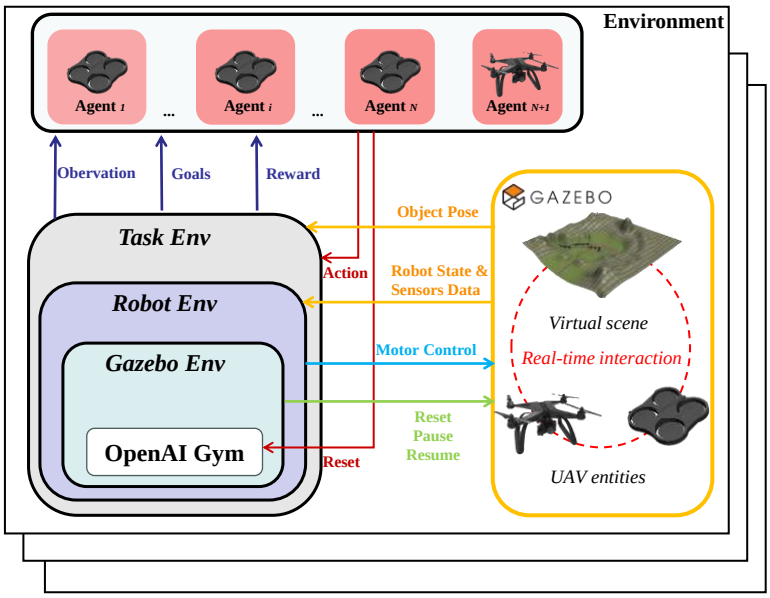

Fig. 2. The framework of the integrated simulation environment based on ROS.

Specifically, the state $\boldsymbol{s}_i(t)$ in the state space $\boldsymbol{S}_i$ can be represented by

$$\boldsymbol{s}_i(t) = [\boldsymbol{l}(t), l_{p_i-e}(t), \min(\boldsymbol{l}(t)), \alpha_{c_i}(t), \alpha_{p_i-e}(t)]\,, \tag{5}$$

where each $\boldsymbol{s}_i(t)$ contains 28 elements, and vector $\boldsymbol{l}(t)$ is made up of the first 24 elements $l_{1_i}(t), l_{2_i}(t), \cdots, l_{24_i}(t)$, which are the distance to the obstacle detected by the LIDAR sending 24 beams to the plane at an interval of 15°. The last 4 beams are composed of the distance between the centroid coordinates of the PUAV and the centroid coordinates of the EUAV, the distance of the nearest obstacle, the orientation angle of the UAV, and the yaw angle of the nearest obstacle, respectively.

The action $\boldsymbol{a}_i(t)$ in the action space $\boldsymbol{\mathcal{A}}_i$ can be given by

$$\boldsymbol{a}_i(t) = [a_i(t), \varsigma_i(t)]\,. \tag{6}$$

Based on the above analysis, the multi-UAV pursuit-evasion game can be decomposed into the following objectives.

Policy optimization objective. At this stage, each UAV is trained for policy improvement to gain optimal policy for the generation of offline dataset. For each UAV, other UAVs are equivalent to dynamic obstacles in the environment, and PUAVs should move close to EUAVs to gain reward r_{t_i}. To optimize the policy π_i of each UAV, we should ensure that the total expected reward is maximized,

which can lead to the formulation of the constraint optimization problem

$$\max_{\pi_{\theta_i}} J\left(\theta_i\right) = \max_{\pi_{\theta_i}} E\left[\sum\nolimits_{t=1}^{T=\infty} r_{t_i}\right], \tag{7}$$

where θ_i stands for the parameter of the policy π_i.

Motion constraint objective. In view of the constraints of the structure and power of the UAV itself, the action can be taken involve acceleration $\|a_i(t)\| = \sqrt{a_{x_i}(t)^2 + a_{y_i}(t)^2} \in [a_{\min_i}, a_{\max_i}]$ and angular acceleration $\|\varsigma_i(t)\| \in [\varsigma_{\min_i}, \varsigma_{\max_i}]$. And the speed and angular speed of each UAV may also be limited to a certain range

$$v_{\min_i} \leq \|v_i(t)\| = \sqrt{\dot{x}_i(t)^2 + \dot{y}_i(t)^2} \leq v_{\max_i}, \tag{8}$$

$$\omega_{\min_i} \leq \|\omega_i(t)\| = \ddot{\psi}_i(t) \leq \omega_{\max_i}. \tag{9}$$

On one hand, in order to ensure that there is no collision between UAVs and between UAVs and obstacles, we need to restrict the distance between UAVs, and between each UAV and obstacles, i.e.

$$\min(\boldsymbol{l}(t)) \geq l_{\min}^{i \leftrightarrow j}, \forall\ i, j \in \mathcal{N}, i \neq j. \tag{10}$$

On the other hand, when the PUAV approaches the EUAV, it needs to make sure that the distance is less than a certain value to complete the goal and get reward $r_i(t)$, and we have

$$l_{p_i - e}(t) \leq l_{\max}^{i \leftrightarrow e}, \forall\ i \in \mathcal{N}_p, \tag{11}$$

where $\mathcal{N}$ is the set of UAVs and obstacles, and $\mathcal{N}_p$ is the set of PUAVs.

3 MAISAC and MAIDT-Based Offline RL Training Strategy

In this section, the MAISAC and MAIDT-based offline RL training strategy (MMOTS) is investigated to train multiple UAVs for pursuit-evasion game and designated constraints. As observed from Fig. 3, it comprises two stages. At the first stage, we first introduce the MAISAC to realize policy improvement, and then optimal policy π^* is selected to generate the offline dataset. At the second stage, offline dataset is used to train the model of MAIDT, which will be applied to multi-UAV to complete the pursuit-evasion game.

3.1 Multi-Agent Independent Soft Actor-Critic

Inspired by the original SAC algorithm [9, 8], we propose the MAISAC algorithm and employ it to the multi-UAV situation, and train UAVs for policy improvement via the framework of decentralized training with decentralized execution (DTDE). MAISAC involves modeling two action value functions, Q_{1_i} and Q_{2_i},

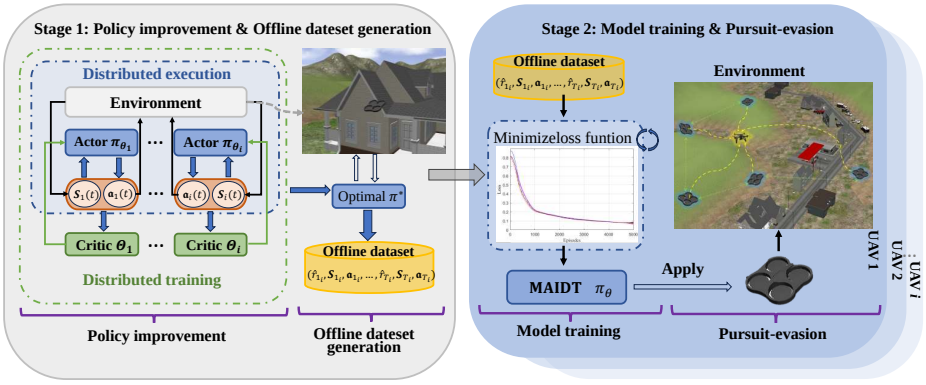

Fig. 3. The framework of the proposed MMOTS, which includes two stages: policy improvement & offline dataset generation and model training & pursuit-evasion. At the first stage, we first introduce the MAISAC to realize policy improvement and generate offline dataset, while at the second stage, the dataset is utilized to train the model of MAIDT, which will be applied to multi-UAV to complete the pursuit-evasion game.

along with a policy function π_{θ_i} for each UAV. To address the issue of Q value overestimation, we utilize two critic networks, Θ_{1_i} and Θ_{2_i}, as well as their respective target networks, $\Theta_{1_i}^-$and $\Theta_{2_i}^-$. Consequently, the loss function of Q can be formulated as

$$L_{Q_{1_i}}\left(\Theta_{1_i}\right)=E_{\left(\boldsymbol{s}_t,\mathfrak{a}_t,r_t,\boldsymbol{s}_{t+1}\right)\sim\mathcal{D}_i}[1/2(Q_{\Theta_{1_i}}\left(\boldsymbol{s}_t,\mathfrak{a}_t\right)-(r_t+\gamma V_{\Theta_{1_i}^-}\left(\boldsymbol{s}_{t+1}\right)))^2, \quad (12)$$

$$L_{Q_{2_i}}\left(\Theta_{2_i}\right)=E_{\left(\boldsymbol{s}_t,\mathfrak{a}_t,r_t,\boldsymbol{s}_{t+1}\right)\sim\mathcal{D}_i}[1/2(Q_{\Theta_{2_i}}\left(\boldsymbol{s}_t,\mathfrak{a}_t\right)-(r_t+\gamma V_{\Theta_{2_i}^-}\left(\boldsymbol{s}_{t+1}\right)))^2, \quad (13)$$

where $\mathcal{D}_i$ denotes the replay buffer to store collected data, while $V_{\Theta_{1_i}^-}\left(\boldsymbol{s}_t\right)$ and $V_{\Theta_{2_i}^-}\left(\boldsymbol{s}_t\right)$ represent the state value function with parameters $\Theta_{1_i}^-$ and $\Theta_{2_i}^-$, respectively. To prevent the UAV i from getting stuck in local optimal policy, we introduce entropy regularization and the $V_{\Theta_{1_i}^-}\left(\boldsymbol{s}_{t+1}\right)$ and $V_{\Theta_{2_i}^-}\left(\boldsymbol{s}_{t+1}\right)$ can be given by

$$V_{\Theta_{1_i}^-}\left(\boldsymbol{s}_{t+1}\right)=\min_{j=1,2} Q_{\Theta_{j_i}^-}\left(\boldsymbol{s}_{t+1},\mathfrak{a}_{t+1}\right)-\alpha_i\log\pi_{\theta_i}\left(\mathfrak{a}_{t+1}\mid\boldsymbol{s}_{t+1}\right), \quad (14)$$

$$V_{\Theta_{2_i}^-}\left(\boldsymbol{s}_{t+1}\right)=\min_{j=1,2} Q_{\Theta_{j_i}^-}\left(\boldsymbol{s}_{t+1},\mathfrak{a}_{t+1}\right)-\alpha_i\log\pi_{\theta_i}\left(\mathfrak{a}_{t+1}\mid\boldsymbol{s}_{t+1}\right), \quad (15)$$

where α_i stands for the regularization coefficient that controls the importance of entropy. Subsequently, we can derive the loss function of policy for UAV i from the KL divergence, which is simplified to

$$L_{\pi_{\theta_i}}\left(\theta_i\right)=E_{\boldsymbol{s}_t\sim\mathcal{D}_i,\mathfrak{a}_t\sim\pi_{\theta_i}}\left[\alpha_i\log\left(\pi_{\theta_i}\left(\mathfrak{a}_t\mid\boldsymbol{s}_t\right)\right)-\min_{j=1,2} Q_{\theta_{j_i}}\left(\boldsymbol{s}_t,\mathfrak{a}_t\right)\right]. \quad (16)$$

Additionally, considering that sampling action is not derivable according to Gaussian distribution $\mathfrak{n}$, the reparameterization trick is introduced. So the policy

function can be expressed as $\mathfrak{a}_t = f_{\theta_i}(\epsilon_t; \boldsymbol{s}_t)$, where ϵ_t is the noise random variable. Considering two action value functions at the same time, the loss function of the policy can be rewritten as

$$L_{\pi_{\theta_i}}(\theta_i) = E_{\boldsymbol{s}_t \sim \mathcal{D}_i, \epsilon_t \sim n}\Big[\alpha_i \log\left(\pi_{\theta_i}(f_{\theta_i}(\epsilon_t; \boldsymbol{s}_t) \mid \boldsymbol{s}_t)\right) - \min_{j=1,2} Q_{\Theta_{j_i}}(\boldsymbol{s}_t, f_{\theta_i}(\epsilon_t; \boldsymbol{s}_t))\Big]. \tag{17}$$

To adjust the entropy regular term automatically. We constrain the entropy to enable its mean is greater than H_0, and obtain the loss function of α_i, i.e.

$$L(\alpha_i) = E_{\boldsymbol{s}_t \sim \mathcal{D}_i, \mathfrak{a}_t \sim \pi_{\theta_i}(\cdot|\boldsymbol{s}_t)}\left[-\alpha_i \log \pi_{\theta_i}(\mathfrak{a}_t \mid \boldsymbol{s}_t) - \alpha_i H_0\right]. \tag{18}$$

Relying on Eq. (18), we can accurately control the explore-exploit trade-off. Meanwhile taking advantage of the DTDE architecture, MAISAC gains strong scalability, improves the training efficiency and ensures the quality of the offline dataset. Next, to generate the offline dataset, we take the policy corresponding to the maximum total reward during the training process as the optimal policy π^*, which is used for UAVs to interact with the environment for data collection, saving all the trajectories during the process as an offline dataset, which can be expressed as

$$\tau_i = (\hat{r}_{1_i}, \boldsymbol{s}_{1_i}, \mathfrak{a}_{1_i}, \hat{r}_{2_i}, \boldsymbol{s}_{2_i}, \mathfrak{a}_{2_i}, \ldots, \hat{r}_{T_i}, \boldsymbol{s}_{T_i}, \mathfrak{a}_{T_i}), \tag{19}$$

where $\hat{r}_{t_i} = \sum_{t'=t}^{T} r_{t'_i}$ stands for returns-to-go of UAV i.

3.2 Multi-Agent Independent Decision Transformer

Similar to [5], DT is introduced to abstract offline RL problems into seq2seq problems and use sequences to model targets. And we expand DT to MAIDT.

MAIDT is mainly based on transformer architecture, which consists of stacked self-attention layers with residual connections. Each self-attention layer receives n embeddings $\{x_i\}_{i=1}^n$ corresponding to unique input tokens, and outputs n embeddings $\{z_i\}_{i=1}^n$, preserving the input dimensions. The i-th token is mapped via linear transformations to a key k_i, query q_i, and value v_i. The i-th output of the self-attention layer is given by weighting the values v_j by the normalized dot product between the query q_i and other keys k_i, which can be shown as

$$z_i = \sum_{j=1}^{n} \text{softmax}\left(\{< q_i, k_{j'} >\}_{j'=1}^{n}\right)_j \cdot v_j. \tag{20}$$

When training the model, we sample n batches of sequence with length K from the offline dataset. The prediction head corresponding to the input token $\boldsymbol{s}_i(t)$ is trained to predict action $\hat{\mathbf{a}}_i(t)$ with mean-squared error L_{MSE} for continuous actions. So the model training objective is to minimize the error, and the error for each timestep is averaged, and we have

$$\max_{\pi_{\theta'_i}} J'(\theta'_i) = \min_{\pi_{\theta'_i}} L_{MSE}(\theta'_i) = \min_{\pi_{\theta'_i}} \left[-\frac{1}{N}\sum_{j=1}^{N}(\mathfrak{a}_j - \hat{\mathfrak{a}}_j)^2\right]. \tag{21}$$

Algorithm 1 MMOTS Framework

Initialize the training environment, including the replay buffer $\mathcal{D}_i$, critic network and corresponding target network, policy network, MAIDT model parameters Θ_{1_i}, Θ_{2_i}, $\bar{\Theta}_{1_i}$, $\bar{\Theta}_{2_i}$, ϕ_i, α_i, and θ_i' of UAV i.
for each episode k **do**
  Reset the training environment and total reward.
  for each time step t **do**
    Sample an action according to the policy: $\mathfrak{a}_{t_i} \sim \pi_{\theta_i}\left(\mathfrak{a}_{t_i} \mid \boldsymbol{s}_{t_i}\right)$;
    Collect the next state from environment: $\boldsymbol{s}_{t+1_i} \sim \mathcal{P}\left(\boldsymbol{s}_{t+1_i} \mid \boldsymbol{s}_{t_i}, \mathfrak{a}_{t_i}\right)$;
    Calculate reward r_{t_i};
    Store sampling tuple $\left(\boldsymbol{s}_{t_i}, \mathfrak{a}_{t_i}, r_{t_i}, \boldsymbol{s}_{t+1_i}\right)$ into $\mathcal{D}_i$.
    Extract N batches tuple of data from $\mathcal{D}_i$.
    $\Theta_{j_i} \leftarrow \Theta_{j_i} - \lambda_{\Theta_{j_i}} \nabla_{\Theta_{j_i}} J_{\Theta_{j_i}}\left(\Theta_{j_i}\right), j = 1, 2.$
    $\theta_i \leftarrow \theta_i - \lambda_{\theta_i} \nabla_{\theta_i} J_{\theta_i}\left(\theta_i\right).$
    $\alpha_i \leftarrow \alpha_i - \lambda_{\alpha_i} \nabla_{\alpha_i} J_{\alpha_i}\left(\alpha_i\right).$
    $\bar{\Theta}_{j_i} \leftarrow \kappa \Theta_{j_i} + (1-\kappa)\bar{\Theta}_{j_i}, j = 1, 2$
  end for
end for
Collect trajectories τ_i using optimal policy by (19).
Sample n batches of sequence with length K from the offline datasets τ_i.
for each gradient step j **do**
  Update the models of MAIDT by Adam updating on θ_i' on $L_{MSE}\left(\theta_i'\right)$ by (21).
end for

Then the trained model of MAIDT is used to make real-time prediction of the actions of each UAV by inputting the initial state and expected returns-to-go, and finally completes the pursuit-evasion game. The overall framework pseudo-code of MMOTS is detailed in Algorithm 1.

4 Simulation and Analysis

In this section, we will verify the validity of MMOTS by simulating the whole process of training multiple UAVs to conduct the pursuit-evasion game. First, we will introduce the setting of the simulation experiment, and then the whole process of the experiment will be introduced in detail. Finally, the results of simulation experiments are described and analyzed, and the performance of the proposed algorithm will be discussed.

4.1 Experiment Settings

In the simulation experiment, we divide the parameters into three parts and consider them respectively: UAV parameters, simulation parameters and algorithm parameters.

For the UAV and simulation parameters, we consider the experimental site of 430m $\times$ 610m, and take the center of the site as the coordinate origin, the

Table 1. Parameters of UAV, Simulation and Algorithms

	Parameters	Values
UAV parameters	Mass (m_i)	0.85
	Max velocities ($\boldsymbol{v}_m$)	6.0 m/s
	Max acceleration ($\boldsymbol{a}_m$)	5.0 m/s^2
	Max angular velocity ($\boldsymbol{\omega}_m$)	1.5 rad/s
	Max angular acceleration (ς_m)	3.0 rad/s^2
	Inertia moments (I_{xxi}, I_{yyi}, I_{zzi})	0.006, 0.006, 0.001 kgm^2
Simulation parameters	Experimental site size	430m × 610m
	Safe distance ($l_{\min}^{i\leftrightarrow j}$)	6m
	Target distance ($l_{\max}^{i\leftrightarrow e}$)	8m
Algorithm parameters	Learning rate λ	3×10^{-4}
	Discount factor γ	0.99
	Soft updating rate κ	0.01
	Regularization coefficient α	0.2
	Memory capacity C	5×10^5
	Sample batch size B	256
	Training episodes $\mathcal{E}$	150
	Time step per episode Δt	0.5
	Iterations	5000

boundary of the site is treated as a barrier to prevent UAVs from going beyond the boundary. And for the algorithm parameters, in the first stage using MAISAC, we mainly refer to the parameters setting of SAC and modify them slightly [9]. Then in the second stage using MAIDT, we mainly refer to the commonly used DT parameters in [5]. The parameters mentioned in the above are shown in Table 1.

4.2 Process of Experiment and Results analysis

The two stages of MMOTS together constitute the whole process of simulation experiment. In each stage, with the help of radar, UAVs know the specific locations of each other, but don't know the locations of obstacles in advance. At the first stage, we first use MAISAC to train two PUAVs to pursue two EUAVs respectively. Considering that the environment is unstable at this time, the velocity of EUAVs is set to 0 to reduce the difficulty of training. The PUAVs will obtain the real-time reward according to the reward function $\mathcal{R}_p$ and $\mathcal{R}_e$, and they can be respectively written as (22) and (23)

$$\mathcal{R}_p\left(\boldsymbol{s}_{p_i}(t), \mathfrak{a}_{p_i}(t)\right) = \begin{cases} 2, & l_{p_i-e}(t-1) > l_{p_i-e}(t), \\ -2, & l_{p_i-e}(t-1) < l_{p_i-e}(t), \\ -500, & \min(\boldsymbol{l}(t)) \le l_{\min}^{i\leftrightarrow j}, \\ 80, & l_{p_{i-e}}(t) \le l_{\max}^{i\leftrightarrow e}, \\ 0, & \text{else .} \end{cases} \tag{22}$$

$$\mathcal{R}_e\left(\boldsymbol{s}_{e_i}(t), \mathfrak{a}_{e_i}(t)\right) = \begin{cases} -500, \min(\boldsymbol{l}(t)) \le l_{\min}^{i\leftrightarrow j}, \\ 0, \quad \text{else .} \end{cases} \tag{23}$$

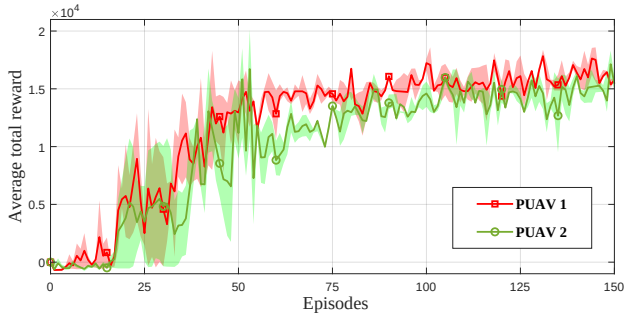

Fig. 4. Average total reward curve of PUAVs utilizing MAISAC for policy improvement.

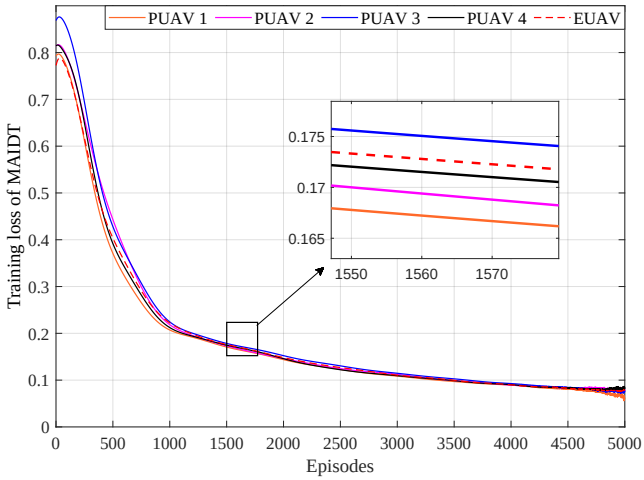

Fig. 5. Training loss curves of the MAIDT model when the target distance $l_{\max}^{i\leftrightarrow e}$ is 8m and the number of PUAVs is 4.

When the distance between the PUAV and the EUAV is less than $l_{\max}^{i\leftrightarrow e}$, the pursuit will be regarded as successful, and the position of the EUAV will be randomly initialized nearby, which drives the corresponding PUAV to continue the pursuit. When the UAV encounters an obstacle or maximum steps per episode is over 3000, the environment will be reset and proceed to the next episode. Then we change the random seed and repeat the experiment three times. With the increase of the number of training episodes, the average total return curves of the two PUAVs show a gradual upward trend, which is shown in Fig. 4.

At episode 120, when the curve is nearly flat, the UAVs' policies can be considered to have been improved to expert policies. Then the policy corresponding to the highest total reward (22514) is selected as the optimal policy to collect data and make offline dataset.

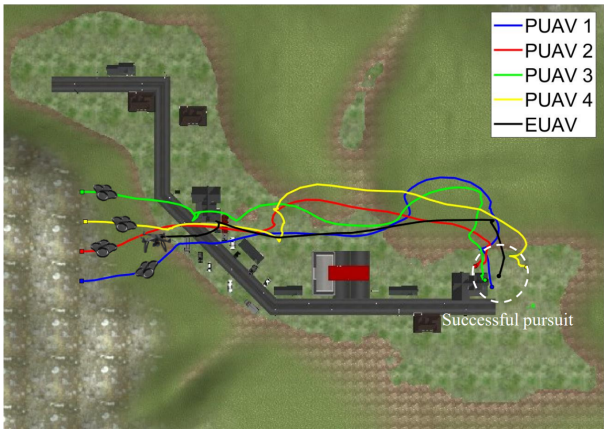

Fig. 6. Trajectories of each UAV for one successful pursuit episode in the simulation environment.

Next, we carry out the second stage of MMOTS, while conducting multiple experiments by changing the target distance $l_{\max}^{i\leftrightarrow e}$ ranging from 6 meters to 8 meters, and the number of PUAVs ranging from two to four. For the convenience of discussion, we take the target distance $l_{\max}^{i\leftrightarrow e}$ as 8m and the number of PUAVs as 4 for an example. The offline dataset generated at the first stage is used to train the MAIDT model, and the train loss curves of the model are obtained, which are shown in Fig. 5.

Next, to make four PUAVs and one EUAV complete the pursuit-evasion game, we apply the trained MAIDT model to UAV. The highest total reward and the initial position of each UAV (280,80), (250,80), (190,80), (220,84), (235,160) are input into the model, which enables the model to predict the next action based on the current returns-to-go and state. After running 100 episodes in simulation environment, we can draw out the trajectories of each UAV in one of the episodes, which are shown in Fig. 6.

As observed from Fig. 6, the EUAV is located in front of PUAVs, and they start to fly at the same time to perform their respective tasks while avoiding obstacles. After 2400 steps, the EUAV is rounded up by PUAVs, which indicated that PUAVs complete the mission. In contrast, when the EUAV is still able to move smoothly after 3000 steps, it indicates that the EUAV completes its task. In addition, the average total reward curves of each UAV corresponding to these 100 episodes can be obtained, as shown in Fig. 7.

By observing Fig. 7, it can be found that: The average total reward of each PUAV is basically concentrated in 17000 and fluctuates up and down slightly. However, the average total reward of the EUAV is concentrated in -500 and is equal to 0 in rare cases. This indicates that the PUAVs successfully completes pursuit most of the time. Furthermore, we also conduct the ablation experiment with the target distance ranging from 6 meters to 8 meters, and the number of PAUVs ranging from two to four. And finally we can obtain the average success

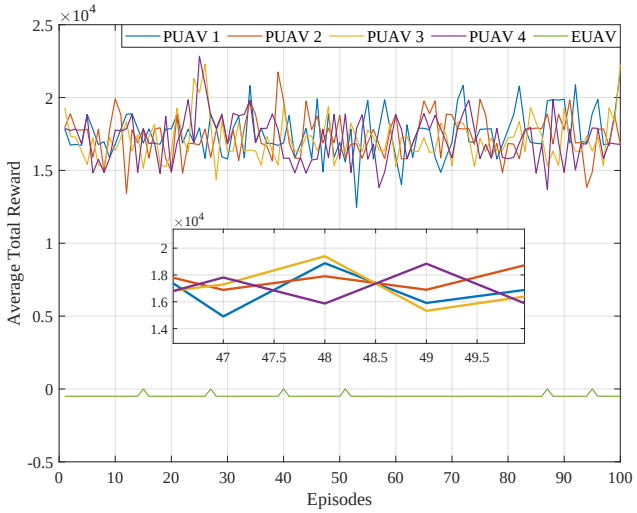

Fig. 7. Average total reward curves of each UAV when the target distance $l_{\max}^{i\leftrightarrow e}$ is 8m and the number of PUAVs is 4.

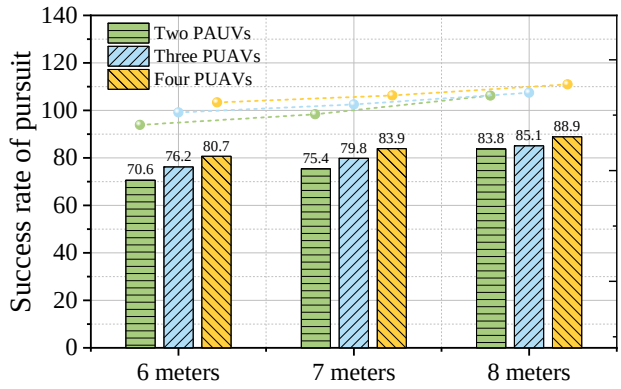

Fig. 8. The success rate of pursuit with the target distance $l_{\max}^{i\leftrightarrow e}$ ranging from 6m to 8m, and the number of PUAVs ranging from two to four.

rate of pursuit through these experiments, and present the simulation experiment results, as shown in Fig. 8 and Table 2.

The results of the simulation in the above show that, with the increase of target distance and PUAVs number, the success rate of pursuit shows an increasing trend, and all cases achieve a success rate of more than 70%, which indicates the excellent generalization and performance of MMOTS. In addition, only a small number of UAVs are used for policy improvement and offline dataset generation. As a result, the pursuit-evasion game can be extended to accommodate UAVs of

Table 2. Results of Experiments

Target distance	PUAVs number	MMOTS
6 meters	Two	70.6%±6.7%
6 meters	Three	76.2%±5.2%
6 meters	Four	80.7%±3.7%
7 meters	Two	75.4%±6.7%
7 meters	Three	79.8%±4.5%
7 meters	Four	83.9%±3.1%
8 meters	Two	83.8%±4.6%
8 meters	Three	85.1%±2.8%
8 meters	Four	88.9%±2.4%

any quantity, demonstrating the excellent scalability of MMOTS and enhancing training efficiency.

5 Conclusion

In this paper, we present a special training strategy named MMOTS, which is designed to train multiple UAVs to complete the pursuit-evasion game within a self-designed integrated simulation environment. There are two stages in the MMOTS: the first stage employs MAISAC to facilitate policy improvement and offline dataset generation, while the second stage utilizes MAIDT to conduct model training and pursuit-evasion simulations. Simulation results validate the excellent generation and scalability of MMOTS, demonstrating strong performance in the context of multi-UAV pursuit-evasion game. As a part of future work, we are planning to further improve the realism of the simulation, and conduct both simulation and real-world experiments in even more complex tasks.

Acknowledgement. This work of Jingjing Wang was partly supported by the National Natural Science Foundation of China under Grant No. 62222101, partly supported by supported by Aeronautical Science Foundation of China (ASFC) under Grant No. 2022Z071051013, partly supported by the Beijing Natural Science Foundation under Grant No. L232043 and No. L222039, and partly supported by the Fundamental Research Funds for the Central Universities. This work of Yong Ren was partly supported by the National Natural Science Foundation of China under Grant 62127801, partly supported by the National Key Research and Development Program of China under Grant 2020YFD0901000.

References

1. Aldao, E., González-deSantos, L.M., Michinel, H., González-Jorge, H.: Uav obstacle avoidance algorithm to navigate in dynamic building environments. Drones **6**(1) (2022)

2. Altan, A., Hacıoğlu, R.: Model predictive control of three-axis gimbal system mounted on UAV for real-time target tracking under external disturbances. Mechanical Systems and Signal Processing **138**, 106548 (2020)
3. Azar, A.T., Koubâa, A., Mohamed, N.A., Ibrahim, H.A., Ibrahim, Z.F., Kazim, M., Ammar, A., Benjdira, B., Khamis, A.M., Hameed, I.A., Casalino, G.: Drone deep reinforcement learning: A review. Electronics **10**, 999 (2021)
4. Chakraborty, N., Chao, Y., Li, J., Mishra, S., Luo, C., He, Y., Chen, J., Pan, Y.: Rtt-based rogue uav detection in iov networks. IEEE Internet of Things Journal **9**(8), 5909–5919 (2022)
5. Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., Mordatch, I.: Decision transformer: Reinforcement learning via sequence modeling. In: Advances in Neural Information Processing Systems. vol. 34, pp. 15084–15097 (2021)
6. Chung, S.J., Paranjape, A.A., Dames, P., Shen, S., Kumar, V.: A survey on aerial swarm robotics. IEEE Transactions on Robotics **34**(4), 837–855 (2018)
7. Fu, X.W., Zhu, J.D., Wei, Z.Y., Wang, H., Li, S.L.: A UAV pursuit-evasion strategy based on ddpg and imitation learning. International Journal of Aerospace Engineering **2022**, 1–14 (2022)
8. Haarnoja, T., Tang, H., Abbeel, P., Levine, S.: Reinforcement learning with deep energy-based policies. In: Proceedings of the 34th International Conference on Machine Learning. pp. 1352–1361 (2017)
9. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: Proceedings of the 35th International Conference on Machine Learning. vol. 80, pp. 1861–1870 (2018)
10. Huang, H., Savkin, A.V.: Deployment of heterogeneous uav base stations for optimal quality of coverage. IEEE Internet of Things Journal **9**(17), 16429–16437 (2022)
11. Kumar, A., Fu, J., Soh, M., Tucker, G., Levine, S.: Stabilizing off-policy q-learning via bootstrapping error reduction. In: Advances in Neural Information Processing Systems (2019)
12. Levine, S., Kumar, A., Tucker, G., Fu, J.: Offline reinforcement learning: Tutorial, review, and perspectives on open problems. arXiv preprint arXiv:2005.01643 (2020)
13. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. In: International Conference on Learning Representations (2016)
14. Mathur, A., Atkins, E.M.: Design, modeling and hybrid control of a quadplane. In: AIAA Scitech 2021 Forum (2021)
15. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
16. Pan, X., Yao, J., Kou, H., Wu, T., Xiao, C.: Harmonicnerf: Geometry-informed synthetic view augmentation for 3d scene reconstruction in driving scenarios. In: ACM Multimedia 2024 (2023)
17. Yao, J., Lai, Y., Kou, H., Wu, T., Liu, R.: Qe-bev: Query evolution for bird's eye view object detection in varied contexts. In: ACM Multimedia 2024 (2024)
18. Yao, J., Qian, Q., Hu, J.: Multi-modal proxy learning towards personalized visual multiple clustering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14066–14075 (2024)