ADVANCED INTELLIGENT SYSTEMS
Open Access
www.advintellsyst.com

Multiple Unmanned Aerial Vehicle Formation Control through Deep Reinforcement Learning with Offline Sample Correction

Zhongkai Chen, Han Zhou, Chao Yan, Yihao Sun, and Xiaojia Xiang*

Multiple unmanned aerial vehicle (multi-UAV) formation plays a pivotal role in both military and civilian applications. However, enabling multi-UAV to efficiently acquire collaborative formation policies is challenging, primarily due to the complex kinematics and dynamic environmental factors involved. Herein, a deep reinforcement learning-based multi-UAV formation control approach is proposed. The multi-UAV formation control problem as a Markov decision process to capture the inherent sequential decision-making process is first modeled. Furthermore, an enhanced actor-critic algorithm called offline sample correction actor-critic (OSCAC) to learn the formation control policy is proposed. The core insight behind OSCAC is to optimize the utilization of historical data through correcting offline samples. Specifically, OSCAC adjusts the importance weights of the samples based on the discrepancies between the current policy and previous policies during each update step, making better use of past experience and improving the learning performance. The effectiveness and reliability of the proposed approach are validated through numerical simulations, software-in-the-loop simulations, and real-world experiments.

1. Introduction

With the rapid development of technologies such as artificial intelligence and wireless communication, unmanned aerial vehicle (UAV) systems have become increasingly intelligent and practical. UAV systems have been widely applied in both military and civilian domains, including surveillance,[1] agriculture,[2] target search, and tracking in complex environments.[3,4] Nowadays, it has been recognized that multiple UAV (multi-UAV) systems can tackle complex problems with lower costs compared to single-UAV systems. Furthermore, multi-UAV systems exhibit superior robustness, fault tolerance, and economic efficiency,[5–7] making them a viable solution for addressing various challenging tasks.

Formation control has been a research hotspot in multi-UAV systems, aiming to maintain a specific geometric structure among UAVs during flight for collaborative task execution. Conventional formation control methods include leader–follower approach,[8] virtual structure method,[9] behavior-based control,[10] and algebraic graph method.[11] Chen et al.[12] proposed a leader–follower formation framework for UAVs in global positioning system (GPS) denied environments, utilizing two leaders and direction-only measurements to minimize sensing needs. Complex maneuvers are enabled by interior angle constraints, and an estimation-based control algorithm ensures stability and accuracy. Liang et al.[13] studied 3D leader–follower formation control for fixed-wing UAV swarms. They simplified the model using Frenet–Serret transformation, ensured stability through Lyapunov theory, and incorporated adaptive disturbance observers and artificial potential fields to handle disturbances and collision avoidance. Guo et al.[14] proposed a single-route virtual structural formation method that achieves precise formation control of UAVs without the need for real-time data communication. Li et al.[15] introduced a method that integrates formation decision-making, pigeon-flock-inspired trajectory planning, along with a hierarchical warning system, enabling UAV swarms to adapt their formations to environmental changes, guaranteeing efficient collision avoidance. Tanveer et al.[16] presented a graph-based formation metric method to address the problem of autonomously executing large-scale drone formation flights in dense environments.

Obstacle avoidance method combines an enhanced APF algorithm with consensus formation control, overcoming traditional APF′s local minima and target unreachable issues, thus allowing UAVs to effectively avoid both static and dynamic obstacles.

Traditional control methods often rely on precise models of platform dynamics and disturbances for control law design. These models, however, tend to be complex, nonlinear, and time-varying, posing significant challenges for the design of control laws.[17] Additionally, sensor inaccuracies and environmental

Z. Chen, H. Zhou, C. Yan, Y. Sun, X. Xiang
College of Intelligence Science and Technology
National University of Defense Technology
Changsha 410073, China
E-mail: yanchao@nuaa.edu.cn

C. Yan
College of Automation Engineering
Nanjing University of Aeronautics and Astronautics
Nanjing 211106, China

The ORCID identification number(s) for the author(s) of this article can be found under https://doi.org/10.1002/aisy.202500208.

DOI: 10.1002/aisy.202500208

disturbances further complicate the modeling process, thus limiting the range of applications for multi-UAV swarms. In recent years, model-free reinforcement learning (RL) methods have received significant attention. RL, a subfield of machine learning, involves the agent interacting with the environment and receiving rewards or punishments that guide its behavior toward learning an optimal policy. The RL framework operates independently of environmental models, making it suitable for decision-making and addressing complex control problems.[18,19]

The effectiveness of RL methods in formation control has been preliminarily verified. Pan et al.[20] integrated multi-agent deep deterministic policy gradient algorithm[21] with the leader–follower method to achieve flexible and efficient multi-agent circular formation control. Li et al.[22] designed a distributed circle formation algorithm that combines deep RL with artificial potential law, facilitating formation generation, formation transformation, and collision avoidance. Shi et al.[23] explored formation control for fixed-wing UAVs, introducing a dynamic model and utilizing deep RL to efficiently handle nonlinear and coupled control challenges while circumventing singularity issues. Wang et al.[24] presented a heuristic RL approach that integrates multi-agent RL with heuristic functions, solving the issue of inefficient formation transformation. Xu et al.[25] proposed a centralized collision-free formation-keeping method based on the proximal policy optimization (PPO) algorithm[26] and developed an efficient communication protocol to optimize the formation flight and exploration efficiency of UAVs. Yu et al.[27] introduced a PPO-based method with hierarchical management mechanism for UAV swarm formation control, achieving stable and reliable management of UAV groups in dynamic environments. However, these methods do not efficiently utilize experience samples during training, which often leads to inefficient training and slow convergence.

To address this limitation, importance sampling is a potential solution. The core idea of importance sampling is to use samples from one distribution and adjust them with appropriate weights to estimate the expected value under another distribution.[28] Luo et al.[29] combined prioritized experience replay with deep RL to improve the success rate of path planning for UAVs in complex dynamic environments. Hu et al.[30] presented asynchronous curriculum experience replay, which uses multithreading and true priority assignments to enhance experience diversity, significantly boosting UAVs' learning efficiency and task performance in complex environments. Zhong et al.[31] integrated delayed prioritized experience replay with soft actor critic algorithm[32] to tackle UAV path planning problems. Ma et al.[33] introduced a dynamic prioritized experience relay algorithm, enabling UAVs to quickly learn formation strategies in a GPS-denied environment. However, these methods cannot use limited experiences to generate more reliable experiences.

Inspired by prior research, we propose a novel offline sample correction actor-critic (OSCAC) algorithm specifically tailored for multi-UAV formation control. We design an offline sample correction mechanism to enhance the utility of existing experience samples. This is achieved by correcting offline samples and dynamically adjusting their importance weights in response to evolving policy differences during the learning process. Additionally, we enhance the deep deterministic policy gradient (DDPG) algorithm[34] by integrating clipped double Q-learning[35] to further improve performance. To verify the performance of our proposed approach, we perform numerical simulations, software-in-the-loop (SITL) simulations, and real-world experiments. The primary contributions of this article are summarized as follows. 1) A multi-UAV formation control framework based on a Markov Decision Process (MDP) is constructed, taking into account disturbances and errors encountered in actual flights. 2) A novel multi-UAV formation control method based on OSCAC is proposed, which improves learning efficiency and stability. 3) The effectiveness of the proposed method is verified through SITL simulations and real-world flight experiments. The rest of this article is organized as follows: Section 2 introduces the background of the RL and DDPG algorithms. Section 3 formulates the MDP model for formation control. Section 4 introduces the proposed formation control method based on the OSCAC algorithm. Section 5 analyzes and discusses the results from both simulation experiments and real-word experiment. Finally, Section 6 concludes this article.

2. Preliminaries

In this section, we first introduce the basic concept of RL. Then, we briefly describe the DDPG algorithm.

2.1. Reinforcement Learning

In the RL paradigm, problems are typically modeled using MDPs, which are represented by a tuple $<\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}>$. Here, $\mathcal{S}$ denotes the state space, $\mathcal{A}$ signifies the action space, $\mathcal{P}:\mathcal{S}\times\mathcal{A}\rightarrow\mathcal{S}$ refers to the state transition function, and $\mathcal{R}:\mathcal{S}\times\mathcal{A}\rightarrow\mathbb{R}$ is the reward function. The agent learns a policy through interaction with the environment, with the objective of maximizing cumulative reward $U_t=\sum_{k=0}^{T}\gamma^k r_{t+k}$, where T is the time horizon and γ is the discount factor. Assuming a discrete time series $t=1,2,\ldots,$ at each time step t, the agent observes the current state $s_t\in\mathcal{S}$ from the environment and selects an action $a_t\in\mathcal{A}$ based on the current policy $\pi(a_t|s_t)$. After executing the action, the agent transitions to a new state $s_{t+1}\in\mathcal{S}$ according to the state transition function $\mathcal{P}$ and then receives a reward $r_t\in\mathcal{R}$ from the environment. Through the iterative process, the agent refines its policy to determine the optimal π_θ that maximizes the expected return $\mathbb{E}[U_t]$.

2.2. Deep Deterministic Policy Gradient

DDPG[34] is RL algorithm tailored for efficient learning in high-dimensional, continuous action environments. It combines the advantages of both value-based and policy-based methods by using an actor-critic architecture. The actor network generates a deterministic action policy, while the critic network assesses the action-state value, thus enabling the refinement of the actor′s policy parameters via gradient ascent based on expected return.

DDPG introduces an experience replay buffer D to store experience transitions (s_t, a_t, r_t, s_{t+1}) obtained during iterations. In the training phase, the parameter ω of the value function $Q_\omega(s,a)$ can be undated by minimizing the following loss function

$$\mathcal{L}(\omega) = \mathbb{E}[(y - Q_{\omega}(s_t, a_t))^2] \tag{1}$$

The target Q-value y is defined as

$$y = r_t + \gamma \cdot Q'_{\omega'}(s_{t+1}, a_{t+1}) \tag{2}$$

where $Q'_{\omega'}$ represents the target critic network. s_{t+1} is the next time state of the agent, and $a_{t+1} \approx \pi'_{\theta}(s_{t+1})$ is the action selected by the target action network π'_{θ}.

The policy can be updated as follows

$$\nabla_{\theta} J(\theta) = \mathbb{E}[\nabla_{a_t} Q^{\pi}(s_t, a_t)|_{a_t=\pi_{\theta}(s_t)} \nabla_{\theta} \pi_{\theta}(s_t)] \tag{3}$$

where s_t and a_t denote the current state and action of the agent, respectively. The parameters of each target network can be updated as

$$\begin{cases} \omega' = \tau\omega + (1-\tau)\omega' \\ \theta' = \tau\theta + (1-\tau)\theta' \end{cases} \tag{4}$$

where τ is the soft update rate.

3. Problem Formulation

Herein, the problem of multi-UAV formation control is formulated. We begin by presenting a statement of the formation problem. Then, we incorporate disturbances into our considerations and construct the corresponding kinematic equations. Lastly, we model our formation control problem as a MDP.

3.1. Problem Statement

Herein, a leader–follower topology is employed. The leader operates fully autonomously and independently of other agents, flying along a predefined route. A formation controller is designed for the followers, which are tasked with maintaining the formation architecture. Based on the flight status information received from the leader, the followers independently decide their flight actions to ensure that the overall formation adheres to a designated structural configuration. The objective optimization function can be defined as

$$\begin{aligned} &\arg\min_{\pi_{\theta}^{f,i}} \ \mathbb{E}\left[\sum_{t=0}^{T_{\max}} \sum_{i=1}^{N} \left\| p_t^{f,i} - p_t^{l} - E_t \right\|_2\right] \\ &\text{s.t.} \quad (p_t^{f,i}, p_t^{l}) \in D \end{aligned} \tag{5}$$

where N denotes the number of followers; $p_t^{f,i}$ and p_t^{l} represent the positions of the follower and leader at time t, respectively; E_t is the desired offset vector;[36] and domain D indicates the flyable area.

3.2. Kinematic Model

In our formation scenario, we consider the UAVs as homogeneous and flying at a fixed altitude while neglecting the pitch and roll movements of the rotor drones. Consequently, the kinematic model of the UAVs can be simplified onto a two-dimensional plane[37]

$$\frac{d}{dt}\begin{bmatrix} x_i \\ y_i \\ \psi_i \\ v_i \end{bmatrix} = \begin{bmatrix} v_i \cos\psi_i \\ v_i \sin\psi_i \\ \omega_i + \eta_{\psi} \\ u_i + \eta_{v} \end{bmatrix} \tag{6}$$

where $[x_i, y_i]^T \in \mathbb{R}^2$ denotes the global position of the UAV; $\psi_i \in [-\pi, \pi]$ represents the global yaw angle; the perturbation terms (η_{ψ}, η_{v}) that follow normal distributions $\mathbb{N}(\overline{\eta}_{\psi}, \sigma_{\psi}^2)$ and $\mathbb{N}(\overline{\eta}_{v}, \sigma_{v}^2)$, respectively, are used to simulate the disturbances arising from various factors, such as environmental conditions or sensor noise, affecting UAVs.

To ensure realistic and safe operation of the UAVs, we impose constraints on their speed, angular velocity, and acceleration[38]

$$\begin{cases} 0 \le v_i \le v_{\max} \\ -\omega_{\max} \le \omega_i \le \omega_{\max} \\ -u_{\max} \le u_i \le u_{\max} \end{cases} \tag{7}$$

where $v_{\max}$ denotes the maximum speed, $\omega_{\max}$ represents the maximum angular velocity, and $u_{\max}$ signifies the maximum acceleration, respectively.

3.3. MDP Model

The formation task presented in this article can be modeled as an MDP. We proceed to define it in terms of state space, action space, and reward function.

3.3.1. State Space

In our formation scenario, the target point of the follower is shown in **Figure 1**. Based on Equation (6), the state of the UAVs can be defined as $\xi := (x, y, \psi, v)$, where x and y are the global position coordinates, ψ is the yaw angle, and v is the forward velocity. Moreover, we introduce subscripts f and l to denote variables with respect to the follower and the leader. In particular, the state of the leader is $\xi_l := (x_l, y_l, \psi_l, v_l)$, and the state of the follower is $\xi_{f,i} := (x_i, y_i, \psi_i, v_i)$. Assuming that there is a UAV at the target position, whose motion state is consistent with that of the leader, that is, the state of the target point for each follower is $\xi_{tar,i} := (x_{tar,i}, y_{tar,i}, \psi_{tar,i}, v_{tar,i})$. Therefore, we define the state vector s_i of the follower i as follows.

$$s_i := (\psi_l, v_l, \psi_i, v_i, x_{tar,i} - x_i, y_{tar,i} - y_i) \tag{8}$$

where $[\psi_l, v_l]$ and $[\psi_i, v_i]$ represent the global yaw angle and velocity of the leader and the follower, respectively, and

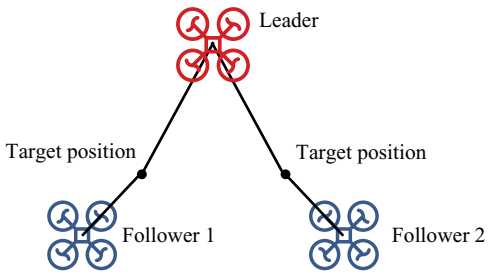

Figure 1. The target position of the followers.

$[x_{tar,i} - x_i, y_{tar,i} - y_i]$ denotes the relative distance between the follower and their corresponding target point.

3.3.2. Action Space

Each UAV is controlled through angular velocity $\omega_i \in \mathbb{R}$ and linear acceleration $u_i \in \mathbb{R}$, defined within the body coordinate frame, with continuous action spaces and subject to kinematic constraints Equation (7), specifically $\omega_i \in [-\omega_{\max}, \omega_{\max}]$ and $u_i \in [-u_{\max}, u_{\max}]$.

3.3.3. Reward Function

In RL, it is crucial to design a reward function that aligns with the actual situation. We introduce a crafted reward function aimed at forming and maintaining the formation of UAVs. The formation reward function is determined as[27,39]

$$U_t^{F,i} = 3 - \rho_t^i \tag{9}$$

where ρ_t^i represented the distance between the follower and the target point. As shown in **Figure 2**, a higher reward value is attained when the follower is closer to the target point, whereas a lower reward value is obtained when it is farther away. In order to make the UAV fly more smoothly and reduce energy consumption, we design an energy reward function[40]

$$\begin{aligned} U_t^{E,i} &= -0.1\left[E(s_t^i) - E(s_{t-1}^i)\right]_+ \\ E(s_t^i) &= \tfrac{1}{2}(m_i v_{t,i}^2 + I_i \omega_{t,i}^2) \end{aligned} \tag{10}$$

where $[\cdot]_+ = \max(x, 0)$; m_i and I_i represent the mass and rotational inertia of the UAVs, respectively; and $E(s_t^i)$ denotes the energy consumption of the followers. The total reward function is

$$U_t^i = U_t^{F,i} + U_t^{E,i} \tag{11}$$

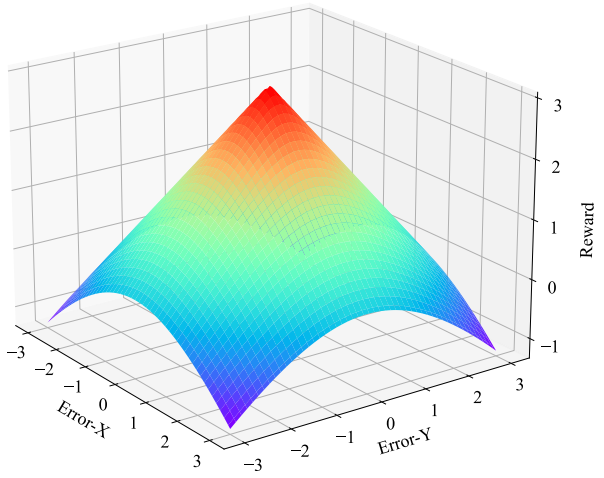

Figure 2. Formation reward function. Error-x and Error-y, respectively, represent the distance differences in the x and y directions between the follower and the target point.

4. Approach

In this section, we provide a detailed design of the OSCAC algorithm. We begin by introducing the overall structure of OSCAC, then present the offline sample correction mechanism, and finally describe the training process of the OSCAC algorithm.

4.1. Overview

The overview of our approach is illustrated in **Figure 3**. The follower first observes the state of leader and combines it with its own state to form an environment joint state s_t for the RL input. Action a_t is chosen based on the Actor network. Both the leader′s and follower′s yaw and velocity responses are incorporated into kinematic equations to derive the next state s_{t+1}. The reward r_t is obtained through the reward function, and tuple (s_t, a_t, r_t, s_{t+1}) is then stored in the experience replay buffer D. In the training process, a small batch of samples is randomly sampled from the experience replay buffer to update both the actor and critic networks. To further enhance the training effectiveness and stability, we introduce an offline sample correction mechanism to assess the significance λ of the samples stored in the experience replay buffer, especially for adjusting the actor network parameters.

4.2. Offline Sample Correction

During the training process in RL, the actor-critic method commonly utilizes a random sampling technique to retrieve samples from the experience replay buffer for network training. Nevertheless, in the early phases of training, the agent′s limited knowledge of the environment might result in collecting experience samples of poor quality. Employing these samples directly in network training may cause instability or inefficient learning. As the agent progressively acquires a more profound comprehension of the environment and refines its policy, the quality and relevance of the collected experiences for training increase. However, this progression may lead to a mismatch in the distribution between the current policy and previously stored experiences. To tackle the aforementioned issues, we introduce an importance sampling mechanism[41] that assigns varying weights to different samples based on their significance. This mechanism optimizes the utilization of experience samples in network training, thereby enhancing its efficiency and effectiveness.

The experience replay buffer D comprises two distinct categories of samples: 1) online samples, gathered through interactions between the current policy and the environment, and 2) offline samples, which are generated and stored by a previous policy. The experience replay buffer can be expressed as the union of these two types of samples

$$(S^{|B|\times m}, A^{|B|\times n}, R^{|B|\times 1}, S'^{|B|\times m}) \approx \mathcal{D} \tag{12}$$

where $|B|$ denotes the batch size of samples for each training iteration; and $S^{|B|\times m}$, $A^{|B|\times n}$, $R^{|B|\times 1}$, and $S'^{|B|\times m}$ represent the matrices of the state, action, reward, and next state, respectively. According to Equation (3), the policy update formula incorporating importance sampling weight can be expressed as follows

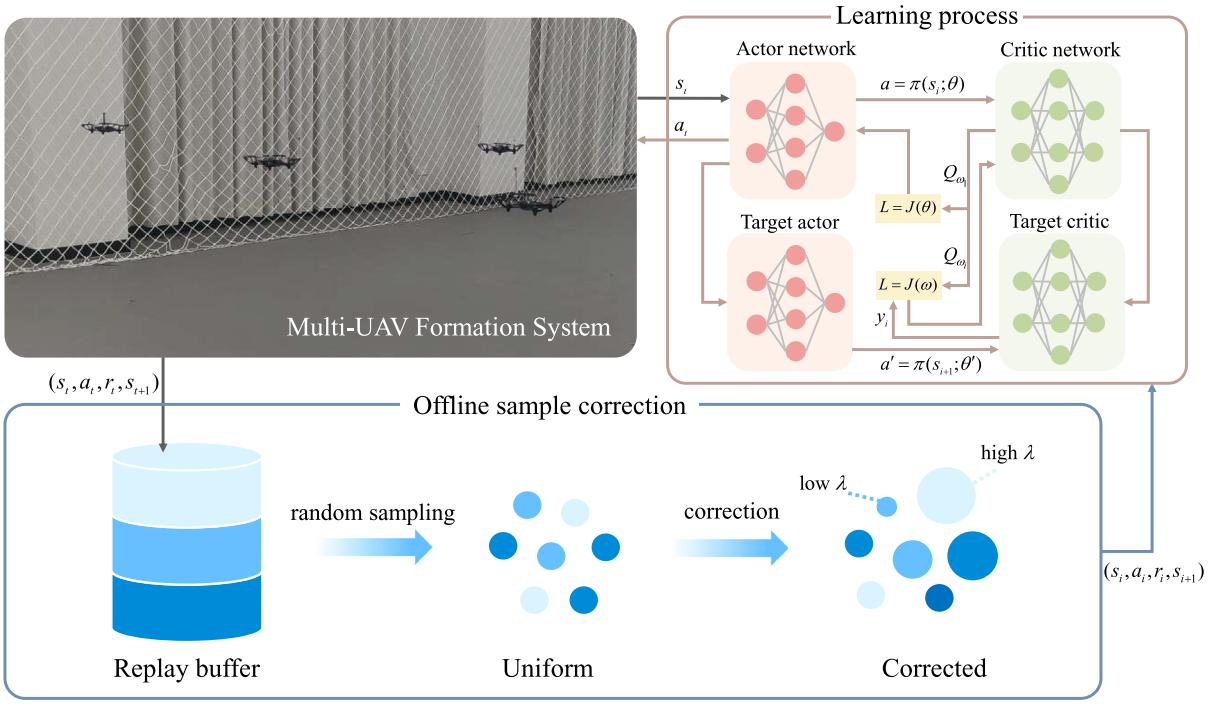

Figure 3. Overview of the offline sample correction actor-critic (OSCAC).

$$\nabla_\theta J(\theta) = \frac{\lambda}{|B|}\sum \nabla_a Q_\omega(S^{|B|\times m}, a)|_{a=\pi_\theta(s)} \nabla_\theta \pi_\theta(S^{|B|\times m}) \tag{13}$$

where λ denotes the sample importance weight coefficient. If there exists a significant discrepancy between the behavior policy in the sampled batch and the current policy, the gradient step size for updating the policy network will be accordingly reduced, indicating that the policy update will be constrained. To measure the degree of correspondence between the current policy and the sampling policy employed for the batch of samples, we first calculate the difference in action selection between the two policies

$$\begin{aligned} \dot{A}^{|B|\times n} &= A^{|B|\times n} - \hat{A}^{|B|\times n} \\ \hat{A}^{|B|\times n} &= \pi_\theta(S^{|B|\times m}) \end{aligned} \tag{14}$$

where $A^{|B|\times n}$ and $\hat{A}^{|B|\times n}$ denote the actions selected by the previous policy and the current policy. To account for differences in actions across distinct batches, we construct a multivariate Gaussian distribution $\mathcal{N}(\dot{\mu}^{1\times n}, \Sigma^{n\times n})$, where

$$\begin{cases} \dot{\mu}^{1\times n} = \frac{1}{|B|}\sum_{i=1}^{|B|} \dot{A}_i^{|B|\times n} \\ \sum^{n\times n} = \frac{1}{|B|-1}\sum_{i=1}^{|B|} (\dot{A}_i^{|B|\times n} - \dot{\mu}^{1\times n})^T (\dot{A}_i^{|B|\times n} - \dot{\mu}^{1\times n}) \end{cases} \tag{15}$$

is the mean value and the covariance matrix, respectively.

We model the previous policies and current policies as multivariate Gaussian distributions, with $P \approx \mathcal{N}(\dot{\mu}^{1\times n}, \Sigma^{n\times n})$ and $C \approx \mathcal{N}(0^{1\times n}, \sigma I^{n\times n})$, where σ denotes the standard deviation of exploration noise. $I^{n\times n}$ represents the identity matrix. The degree of correlation between these two policies can be measured using the Jensen–Shannon divergence[42]

$$\begin{aligned} \rho &= JSD(P\|C) \\ &= \frac{1}{2}(D_{KL}(P\|M) + D_{KL}(C\|M)) \end{aligned} \tag{16}$$

$$\begin{aligned} M &= \frac{1}{2}(P+C) \approx \mathcal{N}\left(\mu_m{}^{1\times n}, \Sigma_m^{n\times n}\right) = \mathcal{N}\left(\frac{1}{2}\dot{\mu}^{1\times n}, \frac{1}{2}\left(\Sigma^{n\times n} + \sigma I^{n\times n}\right)\right. \\ &\quad \left. + \frac{1}{4}\left(\dot{\mu}_1^2 + \ldots + \dot{\mu}_n^2\right)\mathbf{1}^{n\times n}\right) \\ D_{KL}(P\|M) &= \frac{1}{2}\left[\log\frac{|\Sigma_m|}{|\Sigma|} + \mathrm{tr}\left(\Sigma_m^{-1}\Sigma\right) + \left(\dot{\mu} - \mu_m\right)^T \Sigma_m^{-1}\left(\dot{\mu} - \mu_m\right) - n\right] \\ D_{KL}(C\|M) &= \frac{1}{2}\left[\log\frac{|\Sigma_m|}{|\sigma I^{n\times n}|} + \mathrm{tr}\left(\Sigma_m^{-1}\sigma I^{n\times n}\right) + \mu_m^T \Sigma_m^{-1}\mu_m - n\right] \end{aligned} \tag{17}$$

where M is the mixed distribution; $D_{KL}(\cdot\|\cdot)$ is the KL divergence; $\mathbf{1}^{n\times n}$ is the matrix of one; $tr(\cdot)$ is the trace of matrix; and $(\dot{\mu}_1, \ldots, \dot{\mu}_n)$ is the element in $\dot{\mu}^{1\times n}$. If the distribution of the previous policy matches that of the current policy, then $\rho = 0$. In cases where there is partial alignment between the distributions of the previous and current policies, then $\rho \in (0, +\infty)$. To map the similarity measure to a bounded weight interval of $[0, 1]$, we employ a nonlinear transformation for computing the value of λ

$$\lambda = e^{-\rho} \tag{18}$$

The aforementioned formula serves as a metric to quantify the correlation between two policies. Specifically, $\lambda = 1$ signifies perfect alignment between the policies, whereas $\lambda = 0$ denotes complete disagreement.

4.3. Learning Process

We now present our OSCAC algorithm in detail. The DDPG algorithm often suffers from the problem of overestimation, which can lead to the accumulation of errors and consequently result in learning a suboptimal policy. To solve this issue, we integrate clipped double Q-learning[35] into our OSCAC algorithm. Specifically, we train two distinct critic networks, $Q_{\omega_1}, Q_{\omega_2}$, each of which employs its respective target network for the computation of Q-values. Subsequently, we select the minimum of the two computed Q-values as the TD target, ensuring a more conservative estimation and mitigating the potential for overestimation bias

$$y = r_t + \gamma \min_{i=1,2} Q'_{\omega'_i}(s_{t+1}, a_{t+1}) \tag{19}$$

where r_t represents the current reward and a_{t+1} is the action selected by the target actor network π'_θ. To enhance the reliability of the value function estimation, random noise $\varepsilon \approx \text{clip}(N(0,\sigma), -c, c)$ is introduced into the target actions a_{t+1} as defined in Ref. [43], where c represents the upper bound of $|\varepsilon|$, decreasing uniformly from 0.5 to 0.05. The target action is represented as

$$a_{t+1} = \text{clip}(\pi'_\theta(s_{t+1}) + \varepsilon, a_{\text{low}}, a_{\text{high}}) \tag{20}$$

where $a_{\text{low}}, a_{\text{high}}$ denote the minimum and maximum values of the action, respectively. The parameters ω_i of critic networks can be updated as follows

$$\nabla_{\omega_i} J(\omega_i) = \frac{1}{B} \sum \nabla_{\omega_i} (y - Q_{\omega_i}(s_t, a_t))^2, i = 1, 2 \tag{21}$$

where B denotes the batch size of the sample. As illustrated in Section 4.2, the offline sample correction mechanism is introduced to enhance the training of actor network. According to Equation (13), the parameter θ of the actor network can be updated as follows

$$\nabla_\theta J(\theta) = \frac{\lambda}{B} \sum \nabla_{a_t} Q_{\omega_1}(s_t, a_t)|_{a_t=\pi_\theta(s_t)} \nabla_\theta \pi_\theta(s_t) \tag{22}$$

where λ denotes the importance weights of the current training samples. The parameters $\omega'_{i=1,2}$ and θ' of the corresponding target network can be updated as

$$\begin{cases} \omega'_{i=1,2} = \tau\omega_{i=1,2} + (1-\tau)\omega'_{i=1,2} \\ \theta' = \tau\theta + (1-\tau)\theta' \end{cases} \tag{23}$$

where τ is the soft update rate. We intentionally update the actor network at a lower frequency compared to the critic network during training. By doing less frequent policy updates, we can reduce the variance in value estimation, making the critic network more stable and decreasing errors before using it to update the actor network.

Detailed implementation steps are illustrated in **Algorithm** 4.3, and the architecture of the actor-critic network is shown in **Figure 4**.

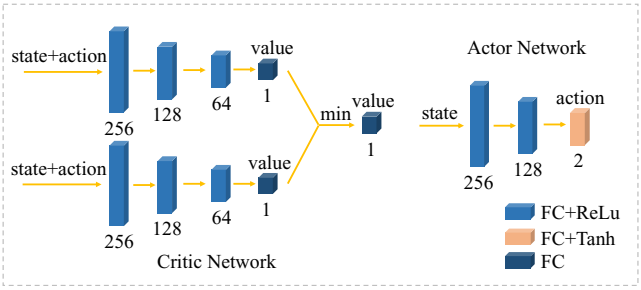

Figure 4. The architecture of the network. The input of the policy network is the environment joint state, and the output of the network is the action. The value networks share the same architecture, where their inputs consist of the environment joint state and the action, and the output of these networks is the estimated Q-value.

Algorithm 1 OSCAC

Initialize the parameters of the actor network π_θ, critic networks $Q_{\omega_1}(s,a)$ and $Q_{\omega_2}(s,a)$
Initialize target networks $\theta' \leftarrow \theta; \omega'_1 \leftarrow \omega_1; \omega'_2 \leftarrow \omega_2$
Initialize the experience replay buffer $\mathbb{D}$
for n to $N_{\max}$ **do**
  Reset the environment
  for $t = 1 : T_{\max}$ **do**
    Select action with exploration noise $a_t = \text{clip}(\pi_\theta(s_t) + \varepsilon, a_{\text{low}}, a_{\text{high}})$, where $\varepsilon \sim \mathcal{N}(0,\sigma)$
    Observe the next state ξ'_f and leader's next state ξ'_l, construct the environment joint state s_{t+1} with Equation (8) and calculate the reward r_t with Equation (11)
    Store the transition tuple (s_t, a_t, r_t, s_{t+1}) in D
    Randomly sample $N_B$$N_{B}$ transitions from D
    Compute TD target through Equation (19
    Update ω_i by the gradient descent using Equation (21)
    if $n \bmod f$
      Calculate the sample importance through Equation (18)
      Update θ by the gradient descent using Equation (22)
      Update target networks parameters θ', ω'_i with Equation (23)
    end if
  end for
end for

4.4. Complexity Analysis

The complexity computation in DRL can be considered in the training and execution phases. During the training phase, compared to conventional DRL algorithms, we introduce the offline sample correction mechanism to compute the importance weights for samples, the complexity of which can be expressed as $O(N_{max} \cdot T_{max}/f \cdot (B \times (m \times h_1 + h_1 \times h_2 + h_2 \times n + n^2) + n^3))$, where N_{max} is the total number of training episodes, T_{max} is the trajectory length per episode, B is the batch size of training samples, f is the actor network update interval,

m and n are the feature dimension of the state and action vector, respectively, h_1 and h_2 are the hidden layers for the actor network. During the execution phase, the follower′s action selection depends solely on the forward propagation of the actor network with complexity $O(m \times h_1 + h_1 \times h_2 + h_2 \times n)$.

5. Simulations and Experiment Results

In this section, we validate our proposed approach through numerical simulations, SITL simulations, and real-world experiments.

5.1. Training Settings

The formation control method proposed in this article can take into account the minimum spanning tree condition.[44] It means that the formation control problem with chain topology can be decomposed into several subproblems. Hence, during the training process, we focus on a simplified scenario consisting of one leader and one follower to enhance efficiency. In the simulations, the learned formation policy can be seamlessly applied to each homogeneous follower. Our formation control policy utilizes the leader–follower topology, where the followers' formation maintenance tasks are assigned based on the position of the leader. The leader acts as an information hub, and as long as the followers can access the state information of the leader, they can autonomously complete and adapt to different formation tasks. And the results indicate that this approach not only facilitates the training process but also enables easy scalability to more complex formation problems involving multiple followers.

During the training phase, we establish the initial conditions for each episode by randomly assigning the starting positions of the leader and follower and predetermining the formation configuration. Episodes end automatically when the training step threshold is reached, without specific termination conditions. We train the policy on a computer configured with an NVIDIA RTX 3060 GPU and an AMD Ryzen 7 5800 H processor with Radeon Graphics. The parameter settings are shown in **Table 1**.

Table 1. Parameter settings.

Parameter	Value	Parameter	Value
Actor learning rate	0.001	Critic learning rate	0.001
Discount factor γ	0.95	Replay buffer size	30000
Batch size N_B	64	Max episode step T_{max}	150
Soft update rate τ	0.01	Delay step f	4
Action noise ε	0.5 → 0.05	Maximum speed v_{max}	5 m s^{-1}
Maximum angular velocity ω_{max}	$\pi/4$	Maximum acceleration u_{max}	1 m s^{-2}
IAPF position term coefficient κ_p	0.30	IAPF proximity effect coefficient κ_a	0.05
IAPF velocity term coefficient κ_v	0.20	DFCM position term factor ι_p	0.30
DFCM velocity term factor ι_v	0.15	DFCM scaling factor β	0.07

5.2. Numerical Simulations

5.2.1. Training Results

Under the same experimental and parameter settings, we select three learning-based control methods for comparison: DDQN,[45] DDPG,[34] and TD3.[43] **Figure 5** illustrates the average reward curves of four learning-based algorithms (i.e., DDQN, DDPG, TD3, and OSCAC) throughout the training process. Our proposed method shows faster convergence in the early stages of training and achieves higher average rewards per episode. This is attributed to its dynamic adjustment of sample importance, which optimizes the utilization of historical experience. The training results indicate that the proposed offline sample correction mechanism improves training efficiency and performance.

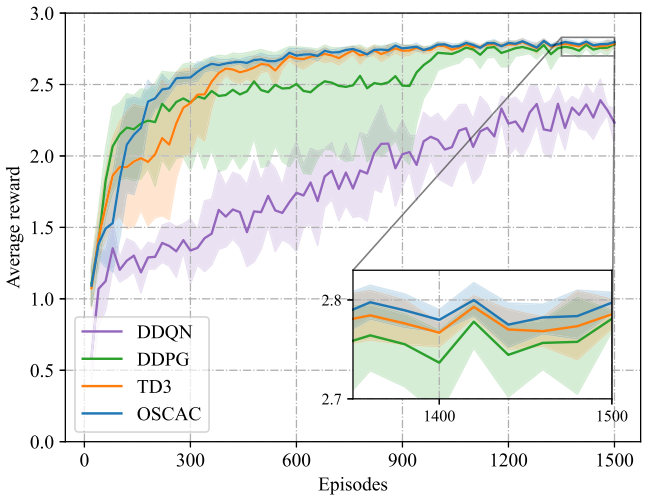

Figure 5. The average reward curves of four algorithms.

5.2.2. Test Results

After training, we test the formation policy learned with our OSCAC algorithm in a specific task. As shown in **Figure 6**, the leader is instructed to fly along a particular trajectory, whereas the followers are tasked with maintaining the square formation configuration. Figure 6a,b presents the trajectories in which the leader follows the path of a straight line and an S-shaped curve, respectively. As the flight progresses, it can be observed that the formation errors of the followers gradually decrease in both scenarios. The reduction in the formation errors over time indicates that the followers are able to accurately track the leader′s path and adjust their positions accordingly, resulting in a coordinated formation. This demonstrates the effectiveness of our OSCAC algorithm in achieving and maintaining formation during both straight-line and curved trajectories. Figure 6c illustrates the trajectory of the leader as it navigates a Z-shaped curve. Evidently, during this complex flight path,

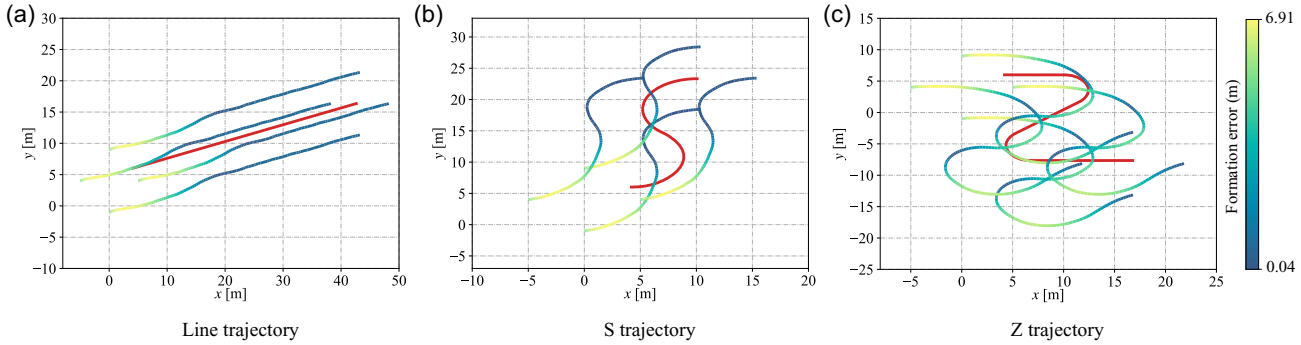

Figure 6. Formation trajectories of five UAVs. In the numerical simulation, the leader flies along straight line a), S-shaped curve (b), and Z-shaped curve (c) trajectories, respectively. The followers are tasked with maintaining square formation configuration. The red line represents the current position of the leader while the blue color bar indicates the formation errors of the followers in their current position.

there are two prominent turns where the followers experience a lag in response, leading to initial increases in formation errors. However, as the flight unfolds, the followers gradually synchronize with the leader, successfully maintaining the formation configuration. This underscores the robustness of the OSCAC algorithm, as it is able to adapt and recover from initial errors, ensuring stable formation flight even during challenging trajectories.

5.2.3. Generalization Results

We then conduct experiments with different environmental settings to validate the generalization performance of OSCAC. We include two conventional methods: improved artificial potential field (IAPF)[46] and distance-based formation control methods (DFCM),[47] in addition to the aforementioned learning-based methods (i.e., DDQN, DDPG, TD3). The two conventional baseline methods are configured as follows

IAPF:[46] the total virtual force is defined as $F_f = -\kappa_p \frac{\boldsymbol{p}_f - \boldsymbol{p}_{tar}}{||\boldsymbol{p}_f - \boldsymbol{p}_{tar}||} - \kappa_a \sum_{f \neq n} \left(1 - \frac{||\boldsymbol{p}_f - \boldsymbol{p}_{tar}||^2}{||\boldsymbol{p}_f - \boldsymbol{p}_n||}\right) \left(\frac{\boldsymbol{p}_f - \boldsymbol{p}_n}{||\boldsymbol{p}_f - \boldsymbol{p}_n||}\right) - \kappa_v \frac{\boldsymbol{v}_f - \boldsymbol{v}_{tar}}{||\boldsymbol{v}_f - \boldsymbol{v}_{tar}||}$, where $\boldsymbol{p}_{tar}$ and $\boldsymbol{v}_{tar}$ denote the position and velocity vectors of the target point in global coordinates, and $\boldsymbol{p}_f$ and $\boldsymbol{v}_f$ represent the corresponding vectors for the follower, $\boldsymbol{p}_n$ symbolizes the velocity vector of the neighbor relative to the follower. κ_p, κ_a, and κ_v are the positive coefficients, determined in Table 1.

DFCM:[47] the control law for the follower is determined as $u_f = \iota_p \frac{\boldsymbol{p}_{tar} - \boldsymbol{p}_f}{||\boldsymbol{p}_{tar} - \boldsymbol{p}_f||} - \iota_v \frac{\boldsymbol{v}_f - \boldsymbol{v}_{tar}}{||\boldsymbol{v}_f - \boldsymbol{v}_{tar}||} + \beta \mathrm{sgn}\left(\int_0^t \left(\frac{\boldsymbol{p}_{tar} - \boldsymbol{p}_f}{||\boldsymbol{p}_{tar} - \boldsymbol{p}_f||} - \frac{\boldsymbol{v}_f - \boldsymbol{v}_{tar}}{||\boldsymbol{v}_f - \boldsymbol{v}_{tar}||}\right) ds - \frac{\boldsymbol{v}_f - \boldsymbol{v}_{tar}}{||\boldsymbol{v}_f - \boldsymbol{v}_{tar}||}\right)$, where ι_p, ι_v, and β are the positive coefficients, specified in Table 1.

In the generalization experiment, we conduct 500 episodes of simulation flight tests for three distinct formations: a square formation with four followers, an H-shape formation with six followers, and an S-shape formation with eight followers, respectively. Each episode consists of 500 time steps, corresponding to a simulation time of 50 s. We set the leader fly randomly while the follower is required to follow the leader and maintain the specific formation configuration. When the follower maintains consistent movement with the leader and stays within the set distance threshold from the target position, we deem the formation to be stable. To quantitatively evaluate the performance of various methods, we record the average formation establishment time T_e, average formation error ρ, and average formation energy consumption E, with detailed data presented in **Table 2**. It is evident that, among the six different control methods, our proposed method excels in terms of formation time, quickly transitioning from the initial state to a stable formation configuration. Although IAPF exhibits the minimum formation error in the square formation experiment, its formation errors increase significantly as the configuration of the formation becomes more complex. At the same time, OSCAC shows relatively low formation errors and energy consumption across all three formation experiments. These experimental results demonstrate the

Table 2. Generalization performance comparison of formation control methods (The bold values represent the best performance achieved in the generalization experiments).

Algorithm	Square formation			H-shaped formation			S-shaped formation		
	T_e (s)	ρ (m)	E (J)	T_e (s)	ρ (m)	E (J)	T_e (s)	ρ (m)	E (J)
IAPF	23.4 ± 9.3	$\mathbf{0.24} \pm 0.47$	0.11 ± 0.03	21.2 ± 9.4	0.87 ± 1.95	0.11 ± 0.03	30.3 ± 10.3	1.29 ± 0.25	0.12 ± 0.03
DFCM	25.1 ± 10.2	0.63 ± 1.21	0.11 ± 0.03	25.0 ± 11.1	0.66 ± 1.33	0.11 ± 0.03	27.5 ± 12.3	0.64 ± 0.92	0.11 ± 0.03
DDQN	16.3 ± 11.3	1.73 ± 0.45	0.09 ± 0.03	15.7 ± 11.1	1.67 ± 0.40	0.09 ± 0.03	16.6 ± 11.2	1.70 ± 0.42	0.09 ± 0.03
DDPG	12.4 ± 8.5	$0.56 \pm \mathbf{0.14}$	0.09 ± 0.02	12.6 ± 8.7	$0.55 \pm \mathbf{0.13}$	0.09 ± 0.02	12.6 ± 8.7	$0.57 \pm \mathbf{0.14}$	0.09 ± 0.02
TD3	$11.4 \pm \mathbf{8.3}$	0.49 ± 0.24	0.08 ± 0.02	$11.8 \pm \mathbf{8.4}$	0.50 ± 0.23	0.08 ± 0.02	12.1 ± 9.0	0.51 ± 0.23	0.08 ± 0.02
OSCAC (Ours)	$\mathbf{11.0} \pm 8.5$	0.40 ± 0.22	$\mathbf{0.07} \pm \mathbf{0.02}$	$\mathbf{11.8} \pm 9.0$	$\mathbf{0.42} \pm 0.22$	$\mathbf{0.07} \pm \mathbf{0.02}$	$\mathbf{11.5} \pm \mathbf{8.3}$	$\mathbf{0.44} \pm 0.23$	$\mathbf{0.07} \pm \mathbf{0.02}$

effectiveness and generalization capability of our proposed formation method.

5.3. SITL Simulations

Herein, we build a high-fidelity SITL simulation system based on XTDrone[48] for evaluating our OSCAC formation policy. XTDrone is a multidrone collaborative simulation platform based on ROS, Gazebo, and Px4. The architecture of SITL simulation system is illustrated in **Figure 7**. The human interaction layer issues the overall mission to the UAV formation, followed by the coordination layer assigning specific submissions to each drone within the formation. Using RL, high-level control layer generates motion commands based on the current joint system state and transmits them to the low-level controller through MAVROS. The low-level control layer subsequently relays the drones' rotor speeds to Gazebo's dynamics model via MAVLink. Concurrently, Gazebo performs rigorous calculations to accurately simulate variations in the drones' motion states and their corresponding responses.

We conduct a formation flight simulation involving seven UAVs, primarily focusing on the formation generation, maintenance, and dynamic reconfiguration. During the simulation, we designate the leader to fly along a predefined route. The followers are tasked with maintaining the formation configuration and perform formation reconfiguration according to instructions. The total simulation time is 60 s, with formation reconfiguration commands sent to all drones every 20 s, and the high-level control layer dispatches motion commands to each drone every 0.1 s. Snapshots of the formation configurations of seven UAVs during the simulation are illustrated in **Figure 8**. As shown in **Figure 9**, when formation reconfiguration commands are issued to each follower, the formation error of the followers gradually decreases. Within approximately 10 s, a new formation configuration forms. During the process of formation maintenance, the formation error of each follower consistently stays within a 0.5 m range. This demonstrates that the followers can quickly respond to changes in target point, execute movements toward the new targets, and ultimately maintain a formation shape stably.

5.4. Real-world Experiments

We finally conduct real-world experiments on the formation generation, maintenance, and dynamic reconfiguration of four

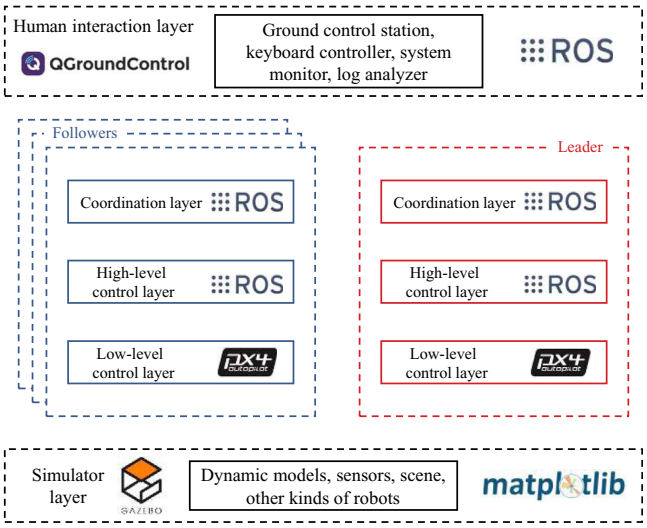

Figure 7. Architecture of SITL simulation system.

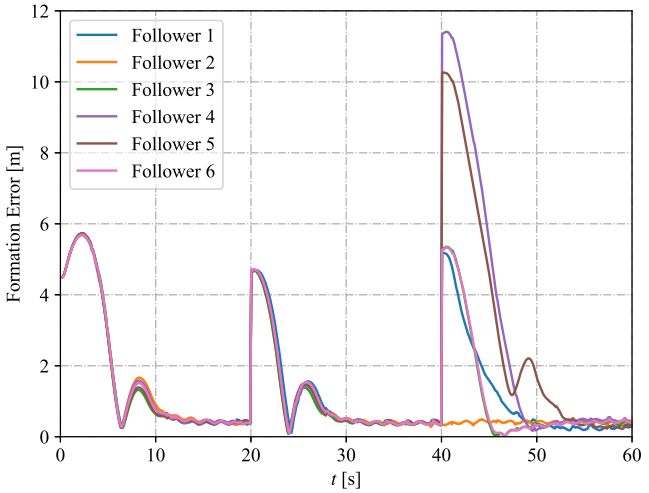

Figure 9. Formation error curves during the entire process for six followers.

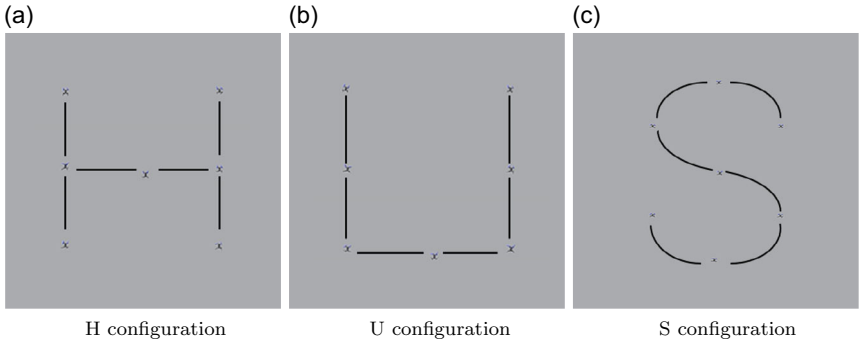

Figure 8. Formation configurations of seven UAVs. In the simulation, the UAV formation is expected to maintain configurations a-c) for 0–20 s, 20–40 s, 40–60 s, respectively. A video about this SITL simulation can be found at https://youtu.be/EecJ5iEvh4w.

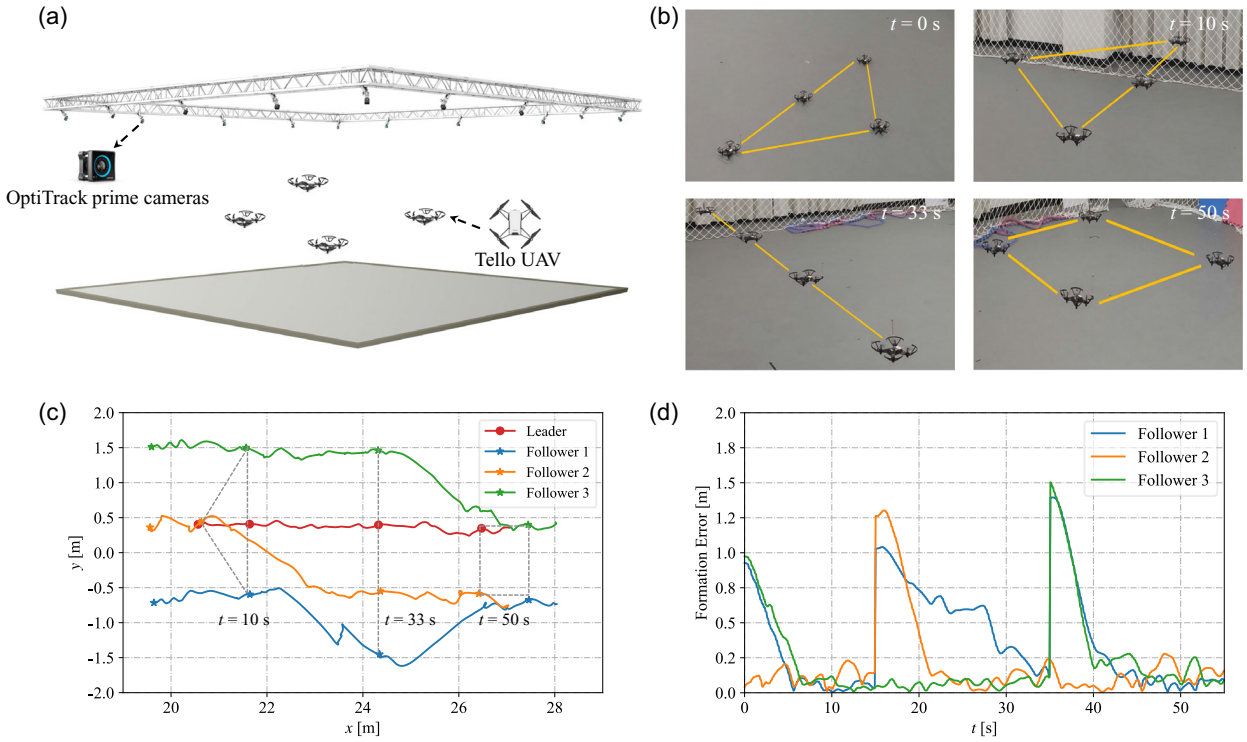

Figure 10. Real-word experiment system setting and results of four-UAV formation. a) Experiment system setting. b) Snapshots of the desired formation configuration for UAVs during the time intervals 0–15 s, 15–35 s and 35–55 s. c) The complete trajectories of the UAVs. d) Formation errors of followers throughout the entire experiment. A video about this real-world experiment can be found at https://youtu.be/EecJ5iEvh4w.

UAVs, leveraging CoFlyers platform, OptiTrack motion capture system, and Tello UAVs, as presented in **Figure 10**a.

In the real-world experiments, we configure the leader to fly a straight trajectory while instructing the UAV formation to change configurations at certain time intervals: $t = 0$ s, $t = 15$ s, and $t = 35$ s, as shown in Figure 10b. The trajectories of four UAVs throughout the entire process are presented in Figure 10c. Figure 10d illustrates that, except for a period when follower 1 encounters relatively large formation errors during the establishment of a linear configuration from $t = 15$ s to 35 s, the entire formation is able to successfully complete the formation task at all other instances. In particular, upon receiving a command to change their configuration, the followers in the formation respond promptly, taking approximately 5 s to form the new configuration and subsequently maintaining it during the remainder of the flight. This demonstrates the reliability and stability of the proposed formation control approach in real-world flight missions.

6. Conclusion

Herein, we propose a deep RL approach for multi-UAV formation control. We first model this problem as a MDP. To better utilize the historical data, we then introduce the OSCAC algorithm. OSCAC optimizes the use of historical data by correcting offline samples and adjusting their importance weights based on policy discrepancies. Through numerical simulations, SITL simulations, and real-world experiments, we demonstrate the effectiveness and reliability of our approach in improving learning efficiency and stability for multi-UAV formation control. In future, we will explore 3D formation tasks for multi-UAV in complex environments with obstacles.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant 62403240 and in part by the Natural Science Foundation of Jiangsu Province under Grant BK20241396.

Conflict of Interest

The authors declare no conflict of interest.

Data Availability Statement

Research data are not shared.

Keywords

deep reinforcement learning, formation control, multi-unmanned aerial vehicle systems, offline sample corrections

Received: February 20, 2025
Revised: June 17, 2025
Published online:

[1] J. Fan, L. Lei, S. Cai, G. Shen, P. Cao, L. Zhang, *IEEE Trans. Veh. Technol.* **2024**, *73*, 1736.
[2] K.-V. Nguyen, C.-H. Nguyen, T. V. Do, C. Rotter, *IEEE Trans. Ind. Inform.* **2023**, *19*, 11664.
[3] J. Xiao, P. Pisutsin, M. Feroskhan, *Adv. Intell. Syst.* **2024**, *6*, 2300761.
[4] J. Xiao, P. Pisutsin, M. Feroskhan, *IEEE Trans. Neur. Net. Lear. Syst.* **2023**, *36*, 313.
[5] J. Zhang, Y. Liu, L. Gao, Y. Zhu, X. Zang, H. Cai, J. Zhao, *Adv. Intell. Syst.* **2024**, *6*, 2300709.
[6] C. Gao, J. Ma, T. Li, Y. Shen, *Complex. Intell. Syst.* **2023**, *9*, 1929.
[7] F. Xiong, H. Zheng, L. Ruan, H. Wang, L. Tang, X. Dong, A. Li, *IEEE Trans. Veh. Technol.* **2020**, *69*, 9002.
[8] L. Consolini, F. Morbidi, D. Prattichizzo, M. Tosques, *Automatica* **2008**, *44*, 1343.
[9] M. A. Lewis, K.-H. Tan, *Auton. Robot.* **1997**, *4*, 387.
[10] C. W. Reynolds, in *Proc. 14th Annual Conf. on Computer Graphics and Interactive Techniques* (Ed: M. C. Stone), ACM, Anaheim, USA **1987**, p. 25.
[11] J. P. Desai, *J. Robot. Syst.* **2002**, *19*, 511.
[12] L. Chen, J. Xiao, R. C. H. Lin, M. Feroskhan, *IEEE Trans. Contr. Syst. Technol.* **2023**, *31*, 1733.
[13] Y. Liang, Q. Dong, Y. Zhao, *Chin. J. Aeronaut.* **2020**, *33*, 2972.
[14] J. Guo, Z. Liu, Y. Song, C. Yang, C. Liang, *IEEE Access* **2023**, *11*, 126027.
[15] S. Li, X. Fang, *Aerosp. Sci. Technol.* **2021**, *114*, 106736.
[16] F. Tanveer, M. B. Kadri, *IEEE Access* **2024**, *12*, 1.
[17] C. Yan, C. Wang, X. Xiang, Z. Lan, Y. Jiang, *IEEE Trans. Ind. Inform.* **2022**, *18*, 1260.
[18] H. Liu, Z. Huang, X. Mo, C. Lv, *IEEE Trans. Intell. Veh.* **2024**, *9*, 4405.
[19] C. Yan, C. Wang, X. Xiang, K. H. Low, X. Wang, X. Xu, L. Shen, *IEEE Trans. Neur. Net. Lear. Syst.* **2024**, *35*, 10894.
[20] C. Pan, X. Nian, X. Dai, H. Wang, H. Xiong, in *Int. Conf. Autonomous Unmanned Systems* (Ed: W. Fu, M. Gu, Y. Niu, Springer, Singapore **2022**, p. 1149.
[21] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, P. Abbeel, I. Mordatch, in *Advances in Neural Information Processing Systems*, Vol.*30* (Ed: U. von Luxburg, I. Guyon, S. Bengio, H. Wallach, R. Fergus), Curran Associates, Long Beach, USA **2017**, p. 1.
[22] B. Li, S. Li, C. Wang, R. Fan, J. Shao, G. Xie, in *China Automation Congress (CAC)*, IEEE, Beijing, China **2021**, p. 4750.
[23] Y. Shi, J. Song, Y. Hua, J. Yu, X. Dong, Z. Ren, in *Chinese Control Conf. (CCC)*, IEEE, Hefei, China **2022**, p. 3456.
[24] K. Wang, R. Xing, W. Feng, B. Huang, in *Int. Con. Wireless Communications, Networking and Applications* (Ed: Z. Qian, M. Jabbar, X. Li), Springer, Singapore **2021**, p. 187.
[25] D. Xu, Y. Guo, Z. Yu, Z. Wang, R. Lan, R. Zhao, X. Xie, H. Long, *Drones* **2022**,*7*, 28.
[26] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov (Preprint), arXiv:1707.06347, v1, Submitted: Jul. **2017**.
[27] N. Yu, J. Feng, H. Zhao, *Alex. Eng. J.* **2024**, *100*, 268.
[28] T. Schaul, J. Quan, I. Antonoglou, D. Silver, in *Int. Conf. Learning Representations*, OpenReview.net, Puerto Rico, USA **2016**, p. 322.
[29] X. Luo, Q. Wang, H. Gong, C. Tang, *IEEE Access* **2024**, *12*, 38017.
[30] Z. Hu, X. Gao, K. Wan, Q. Wang, Y. Zhai, *IEEE Trans. Veh. Technol.* **2023**, *72*, 13985.
[31] J. Zhong, T. Long, J. Sun, J. Li, Y. Cao, in *Int. Conf. Control Science and Systems Engineering (ICCSSE)*, IEEE, Shenzhen, China **2023**, p. 172.
[32] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, in *Int. Conf. Machine Learning* (Ed: D. Jennifer, K. Andreas), PMLR, Stockholm, Sweden **2018**, p. 1861.
[33] B. Ma, Z. Liu, F. Jiang, W. Zhao, Q. Dang, X. Wang, J. Zhang, L. Wang, *Chin. J. Aeronaut.* **2023**, *36*, 281.
[34] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, in *Int. Conf. Machine Learning* (Ed: P. Eric, J. Tony), PMLR, Beijing, China **2014**, p. 387.
[35] J. J. H. Ackermann, V. Gabler, T. Osa, M. Sugiyama (Preprint), arXiv:1910.01465, v1, Submitted: Oct. **2019**.
[36] Z. Sui, Z. P, J. Yi, S. Wu, *IEEE Trans. Neur. Net. Lear. Syst.* **2021**, *32*, 2358.
[37] Z. Xia, J. Du, J. Wang, C. Jiang, Y. Ren, G. Li, Z. Han, *IEEE Trans. Veh. Technol.* **2022**, *71*, 931.
[38] G. Shen, L. Lei, Z. Li, S. Cai, L. Zhang, P. Cao, X. Liu, *IEEE Internet Things J.* **2022**, *9*, 11141.
[39] Y. Sun, C. Yan, X. Xiang, H. Zhou, D. Tang, Y. Zhu, *Ocean Eng.* **2023**, *271*, 113811.
[40] Y. Yin, Z. Chen, G. Liu, J. Yin, J. Guo, *Expert Syst. Appl.* **2024**, *247*, 123202.
[41] Q. Zhang, M. Liu, H. Wang, W. Qian, X. Zhang, *IET Cyber-Syst. Robot.* **2023**, *5*, 1.
[42] J. Lu, M. Henchion, B. Mac Namee, in *Proc. 12th Conf. on Language Resources and Evaluation (LREC)* (Ed: N. Calzolari, F. Béchet, P. Blache, K. Choukri, C. Cieri, T. Declerck, S. Goggi, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, S. Piperidis), European Language Resources Association, Marseille, France **2020**, p. 6740.
[43] S. Fujimoto, H. Hoof, D. Meger, in *Int. Conf. Machine Learning*, (Ed: D. Jennifer, K. Andreas), PMLR, Stockholm, Sweden **2018**, p. 1587.
[44] W. Ren, R. W. Beard, *IEEE Trans. Autom. Control.* **2005**, *50*, 655.
[45] H. Van Hasselt, A. Guez, D. Silver, in *Proc. AAAI Conf. Artificial Intelligence*, AAAI Press, Phoenix, AZ **2016**, p. 30.
[46] Z. Pan, C. Zhang, Y. Xia, H. Xiong, X. Shao, *IEEE Trans. Circuits Syst. II, Exp. Briefs* **2021**, *69*, 1129.
[47] D. Van Vu, M. H. Trinh, P. D. Nguyen, H.-S. Ahn, *IEEE Contr. Syst. Let.* **2020**, *5*, 451.
[48] K. Xiao, L. Ma, S. Tan, Y. Cong, X. Wang, in *Advances in Guidance, Navigation and Control: Proc. 2020 Int. Conf. Guidance, Navigation and Control* (Ed: L. Yan, H. Duan, X. Yu), Springer, Singapore **2022**, p. 5131.