

## Article

# Path Navigation for Multi-Agent Flight System Based on Distributed Model Predictive Control and Reinforcement Learning

Chao Kang , Jihui Xu, Yuan Bian  and Wenjie Tian

School of Equipment Management and Unmanned Aerial Vehicle Engineering, Air Force Engineering University, Xi'an 710043, China; skyline6018@126.com (J.X.); twj-safety@foxmail.com (W.T.)

\* Correspondence: kc1015584136@163.com

**Abstract:** The continuous change in formation configurations offers significant potential for formation flight. However, existing methods neglect the potential of autonomous navigation and learning ability based on environmental conditions, which would limit scalability for large formations to some extent. To address this, this paper presents an integrated framework that incorporates an innovative policy for configuration learning into multi-agent flight systems. Firstly, a decoupled trajectory optimization model is proposed, consisting of prediction and learning modules that ensure local adaptivity, obstacle avoidance and dynamic feasibility of formation configuration. Secondly, we design a stage cost function representing obstacle conditions, configuration variability and minimum distance constraints between agents. Then, to enhance adaptability in constrained scenarios, we introduce a reference value generation method that adjusts trajectory learning based on local observations, penalizing errors. Finally, we propose a configuration learning strategy called the local-level action planner and global cost, which coordinates local trajectory optimization with global maintenance. This method integrates distributed model predictive control (DMPC) with modified multi-agent proximal policy optimization (MAPPO), easing the challenge of tracking trajectories from varying rigid geometric shapes. Simulation results demonstrate that the proposed approach improves success rate compared to traditional methods while maintaining formation and avoiding obstacles.



Academic Editor: Rui Araújo

Received: 9 January 2025

Revised: 13 February 2025

Accepted: 18 February 2025

Published: 19 February 2025

**Citation:** Kang, C.; Xu, J.; Bian, Y.; Tian, W. Path Navigation for Multi-Agent Flight System Based on Distributed Model Predictive Control and Reinforcement Learning. *Appl. Sci.* **2025**, *15*, 2234. <https://doi.org/10.3390/app15042234>

**Copyright:** © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

**Keywords:** formation configurations; formation flight; decoupled trajectory optimization; configuration learning strategy

## 1. Introduction

Formation flight with various changing configurations has presented a significant challenge, demanding the algorithm design the control protocol to change the configuration continuously while effectively avoiding obstacles [1–4]. The traditional control methods have shown strong significant results in the area of maintaining and changing the geometry rigid configuration [3,4]. Meanwhile, the formation flight system is essential for the multi-agents in the constrained environment. However, a rigid configuration maintenance can lead to high system complexity and stability issues [3,5]. Traditional algorithms have mainly focused on position-, displacement- and distance-type studies, based on the types of sensed and controlled variables used thus far. Two key challenges limit the practical application of configurations: (a) The preset trajectory sequence of each agent within the formation restricts configuration changes to time, scale and orientation variations in the target configuration [6–8], especially for the rigid configuration control level [6–9]. (b) The

formation would lead to reconstruction failures from the undesirable situation by unknown obstacles and the agents' breakdowns with much communication consumption [9,10].

Various studies exist on the formation control protocol along with changing configurations in a multi-agent flight system, such as the rigid configuration control level [3–8]. While these methods are valued for their stable structure in formation flight, they need to preserve the relative positions (displacement/distance/bearing-based [1,3,11]) and facilitate transitions between various formation shapes. Preserving collinearity and distance ratios would strongly rely on the predefined change scale and nominal configuration. Alternatively, some methods focus on model predictive control (MPC) [12,13], utilized by constructing a model to meet safety and flexibility requirements only in a strong connected approach. While this policy can ensure unknown obstacle avoidance and dynamically adjusts geometry by modulating velocity values and directions based on current costs to some extent, it heavily depends on strong communication among agents, making it susceptible to reconstruction failures if one or more agents experience breakdowns [13,14]. However, the Multi-Agent Deep Reinforcement Learning (MADRL) technique has shown less research in the area of configuration control.

Based on the practical challenges, the objectives of continuous configuration control can be summarized as follows: an adaptable module for adjusting the local trajectory while maintaining configuration, a predictable module that smoothly responds to sudden changes in values and a versatile module for forming flexible configurations in dense environments. In our previous work [15], proposing a method named as Affine-based Maneuver Control on Optimal Flight System (AMCOFS), we focused solely on the first objective above, based on the local observation of each agent, addressing the conflict between the obstacle avoidance in the local frame and the rigid configuration maintenance in the global one. An integrated flight framework of the global and local control law was designed as a nonlinear optimization problem. However, the *fmincon* nonlinear optimization cannot be extended to the higher dimension caused by the large calculation [16], where the computation of Hessian matrix after sampling all the points' values in every iteration limits the extension capability.

In this paper, we address the prediction and flexibility challenges during the formation interaction with the environment by proposing a novel Deep Reinforcement Learning (DRL) integrated with the distributed model predictive control (DMPC) method. Our framework establishes an online learning policy of integrating local and global control. The problem of multi-agent flight system has been modeled as a fully cooperative DRL problem. Also, we focus on addressing the action selection problem introduced by the reality constraints (note that the system does not include the specific obstacle detection law). Our main contributions can be summarized as follows:

1. The interrelationships among agents are taken into account, with the problem being formulated as a fully cooperative multi-agent task. Following this, an improved multi-agent proximal policy optimization (I-MAPPO)- and DMPC-based integrated framework is employed to learn the optimal cooperative policies dynamically. This process allows for ongoing trial and error during the online learning phase, with the goal of developing a strategy that guarantees a collision-free trajectory.
2. To enhance the efficiency of online learning, the algorithm mainly consists of combined action selection and the reference baseline generation. The concept of the Artificial Potential Fields (APFs) method has been integrated with policy gradient techniques to refine the learning of linear and angular velocities within the action space. Additionally, the cost function has been designed as a stage reward function. Motivated by the local observations and the motion constraints of flight system in practical scenarios, the reference values are derived from the DMPC algorithm with

a two-layer cost function as a novel baseline to DRL, which is on the basis of the kinematic model of a point mass and the affine-based [9,15] trajectory navigation.

3. Extensive simulations are conducted to demonstrate that the proposed integrated algorithm significantly improves system performance compared to the benchmark algorithms.

## 2. Related Work

Traditional studies have been conducted for formation control protocol along with changing configuration in multi-agent flight systems, mainly in distributed control strategies [1,17]. However, these methods exhibit low accuracy and flexibility in dense environment. To address this, refs. [9,18] proposed an affine-based mapping method for target configuration involving translation, rotation, scaling, shear, or compositions of these transformations. Ref. [9] extended the affine-based method with the stress matrix for changing the configuration within the nominal configuration set. However, these methods show the low effectiveness in transforming the configuration geometry in an unknown environment without predefining the change scale and nominal configuration. Additionally, this policy does not account for sudden changes in an agent's velocity regarding both magnitude and direction. Thus, the velocity is specifically calculated at each step of the configuration transformation, relying on prior information about the environment. To overcome this problem, many studies have brought the RL methods to model the issue as a navigation task. Refs. [19,20] employed Q Learning (QL) to establish paths while avoiding static or dynamic obstacles in the plane. However, it hardly converges in complex and highly dynamic environments by extending to the high dimension.

To enhance the ability of trajectory planning of mobile agents for dynamic configuration in dense environments, ref. [21] utilized the Deep Q Network (DQN) algorithm combined with greedy policy to support trajectory planning and optimization in a large-scale dense environment. As for the value-based Double DQN (DDQN) [22] method, ref. [23] explored a high-precision navigation technique for multi-agent systems to enhance the efficiency of target tracking. However, these methods mainly utilized discrete actions with limited agent action space, which may reduce the efficiency in real-world constraints. Ref. [24] proposed a modified Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm. The primary improvement focuses on prioritized experience replay and dynamic delay updates, which reduce the impact of value estimation errors on training time. Ref. [14] introduces the Deep Deterministic Policy Gradient (DDPG) algorithm with continuous action space to address the problems of high dimension of continuous state space. Their experimental results demonstrated that the combined DDPG algorithm exhibited superior performance in generating effective path plans for mobile robots within continuous action spaces. However, the DDPG algorithm suffers from low training efficiency and the local minimum problem, which negatively impacts motion convergence due to a high proportion of illegal policies stemming from the lack of policy action filtering [25].

Therefore, recent research on proximal policy optimization (PPO) has shown great potential in multi-agent navigation. In [26], this algorithm was used to solve the trajectory tracking problem in manipulator and mobile robot areas. Ref. [27] utilized the PPO to ensure safety constraints through constrained neural networks evaluated by the cost function. In addition, ref. [28] developed a PPO-based MADRL framework in the handover and power allocation problem, which fully used the local observation to the lower bound of the objective function and policy gradient. Ref. [29] demonstrated the effectiveness of MAPPO in the Particle World environments, Starcraft II Micromanagement Tasks and the Hanabi Challenge with continuous action spaces. The possibility of applying DRL for path navigation of robots and UAVs has been confirmed in the literature. Compared with

other DRL-based methods, the policy-gradient-based algorithms have more advantages in continuous action generation.

In addition, the prediction module is a vital stage in motion navigation. Ref. [12] utilized a distributed nonlinear model predictive controller (DNMPC) combined with an estimator module to achieve an unrigid configuration. Ref. [13] explored the integration of APF into a nonlinear model predictive controller (NMPC) as a cost function, which conducted configuration changes through a modified A\* path planning approach. While this policy utilizes local observations and current costs to avoid collision and dynamically adjusts geometry, it greatly depends on a limited time interval. Meanwhile, for the local minimum problem of APF, some researchers proposed an APF algorithm combined with RL algorithms. In [30], a modified APF approach based on membrane evolution was proposed for the path navigation of mobile robots. In [31], the APF method was used to solve the problem of the six-axis robot's collision from workspace. In this paper, we carry out an improved centralized training and distributed execution framework in MADRL to achieve continuous actions for path navigation to support the configuration change. To address the flexibility problem, the APF method is utilized in the actor network. In addition, as for the prediction challenges during reinforcement online learning, we integrate our previous research [15] and the DMPC method with the critic network, ensuring environment constraints for each agent.

The paper is organized as follows. Section 3 describes the problems and preliminaries. The description and proof of main results for the proposed algorithm framework are given in Section 4. The results of simulation and discussions are drawn in Section 5. Section 6 concludes the entire paper and provides a glimpse into future prospects. And all the notations and symbols are shown in Appendix A.

### 3. Problem Description and Preliminaries

#### 3.1. Problem Definition

The multi-agent task is built and shown in Figure 1, and the environment contains  $N$  agents and  $M$  obstacles. It can be seen that the agent's position is denoted as  $(x_i, y_i)_t^n$ ,  $i = 1, \dots, n$ ,  $t = 1, \dots, T$ . The obstacle's position is denoted as  $(x_i, y_i)_{obs}^R$ ,  $i = 1, \dots, M$ , and each obstacle is calculated as a circular style with the radius  $R$ . In addition, the predefined desired position of each agent is located at  $(x_i^d, y_i^d)$ ,  $i = 1, \dots, n$ . The overall mission is considered successful when all agents pass through the static obstacles without any collisions and maintain the configuration described in our previous work [15].

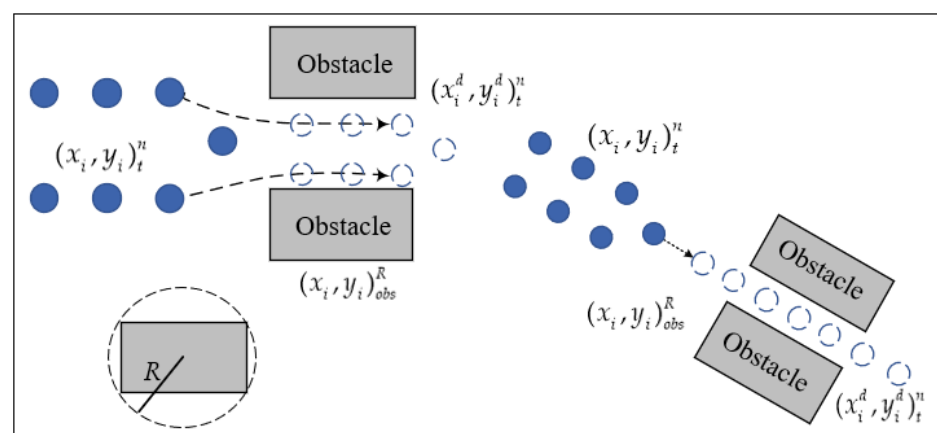


**Figure 1.** Illustration of formation motion with rigid configuration. Dotted line in black represents the desired trajectory. The agents' positions are blue, desired one blue-dotted and obstacles are gray.

However, within realistic constraints such as communication loss or sudden situations, the traditional algorithm may not successfully navigate. This is due to the unknown obstacles causing a loss of rigidity during the transition to a predefined configuration [15].

### 3.2. Kinematic and Dynamic Models of UAV

The multi-agent formation topology is denoted as  $G = \{V, A, \varepsilon\}$ , where the vertex set  $v_i \in V$  within graph  $G$  is mapped to point  $p_i$ . In addition, all agents share the same dynamic model. In addition, a mass-point model can be used to describe an UAV agent, and the fixed-wing agent model is described as follows:

As shown in Figure 2, by the use of the feedback linearization technology, the nonlinear UAV model can be simplified to a linear constant system. Thus, the double-integrator system is reduced as follows:

$$\begin{cases} \dot{\mathbf{p}}_i(t) = \mathbf{v}_i(t) \\ \dot{\mathbf{v}}_i(t) = \mathbf{u}_i(t) \end{cases} \quad (1)$$

where  $\mathbf{p}_i \in \mathbb{R}^n$  represents the position,  $\mathbf{v}_i \in \mathbb{R}^n$  the velocity and  $\mathbf{u}_i \in \mathbb{R}^n$  the acceleration.

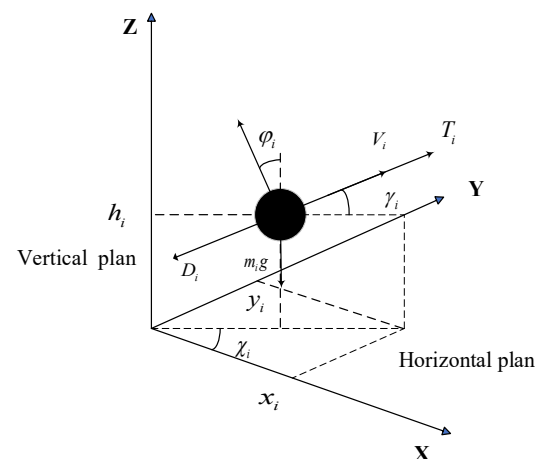


**Figure 2.** Fixed-wing agent model.

It is presupposed that all agents possess a restricted situational awareness, and their ability to detect fellow agents is limited by their equal sensing range. Thus, the neighbor set of an agent  $i$  can be depicted as follows:

$$N_i = \{j : \|\mathbf{p}_{ij}\| < d_{sen}, j = 1, \dots, n; j \neq i\}, \quad (2)$$

where  $\mathbf{p}_{ij} = \mathbf{p}_i - \mathbf{p}_j$  is the position vector from UAV  $i$  to UAV  $j$  (the Euclidean norm), and  $d_{sen}$  is the sensing range. Given the substantial difference in time constants between the horizontal and vertical trajectory controllers, configuration control can be concentrated primarily on achieving consensus in horizontal positional and velocity motion in this paper.

## 4. Proposed Algorithm for Multi-Agent Configuration Change

Given that the decisions of agents are fully interactive, we frame this challenge as a fully cooperative multi-agent task as  $\Gamma = \langle P, A, O, S, R, \theta \rangle$ .

### 4.1. Trajectory Reference Generation

To address the problem of responding to environmental changes and sudden events, this section generates a series of optimal formation sequences by the affine-based DMPC model in short-term future coordinates based on historical coordinates. The fundamental process is shown in Figure 3 as below.

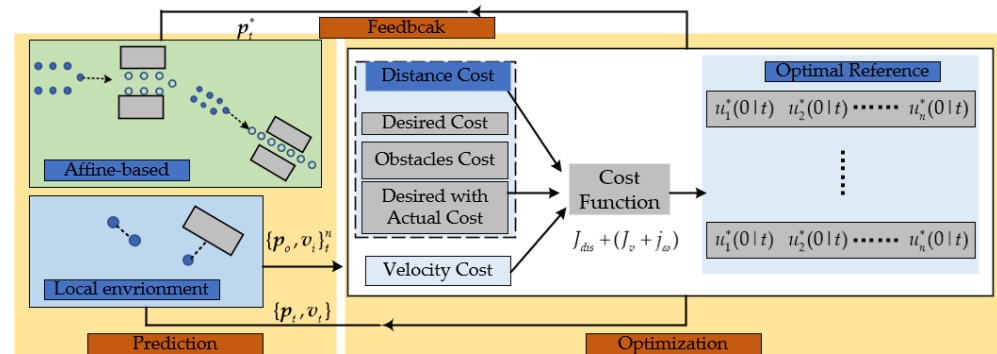


**Figure 3.** Illustration of trajectory reference generation.

#### 4.1.1. Affine-Based Trajectory Planning

Based on the undirected graph  $G$ , the set of nominal configurations is denoted as  $R = [r_1, r_2, \dots, r_N]^T$ , and  $I(r)$  is defined as the affine image which satisfies affinely span  $\mathbb{R}^2$ , shown in Figure 4 (the details of lemmas and definitions can be found in [15]).

$$I(r) = \left\{ P = [p_1^T, p_2^T, \dots, p_N^T]^T \in \mathbb{R}^{2N} \mid p_i = Ar_i + b \right\}, \quad (3)$$

where  $\{A(t) \in \mathbb{R}^{d \times d}, b(t) \in \mathbb{R}^d\}$  are the adjustable parameters for continuous shape transformation. Formulation (3) shows a linear changing style from a set of vectors to another sequences of vectors and contains the translation, rotation, scaling, shear, or compositions of these transformations in linear space. The aim of using the affine-based transfer method is mainly to maintain all the points collinear within the formation.

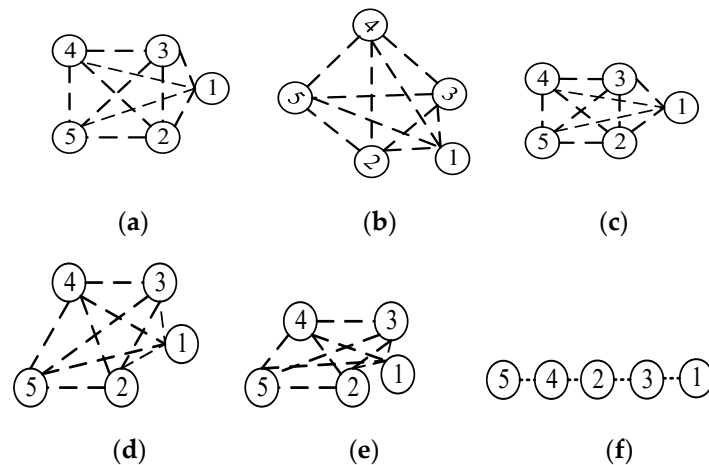


**Figure 4.** Affine-based predefined configuration patterns for illustration: (a) nominal; (b) rotation and scaling; (c) scaling; (d) shear; (e) shear and scaling; (f) linear style. The dotted line represents the outlines of the shape. Agents are represented as 1 to 5.

Based on our previous work [15], firstly, we predefined the desired geometric patterns  $A(r)$  and then generated the global trajectory sequences  $\{p_i^*(k)\}$  with predefined positions based on the knowledge of obstacle position and shape available, as shown in Figure 5 [15]. However, the affine-based trajectory planning method lacks the adaptability to respond proactively to environment change. Thus, the DMPC method is introduced in the paper.

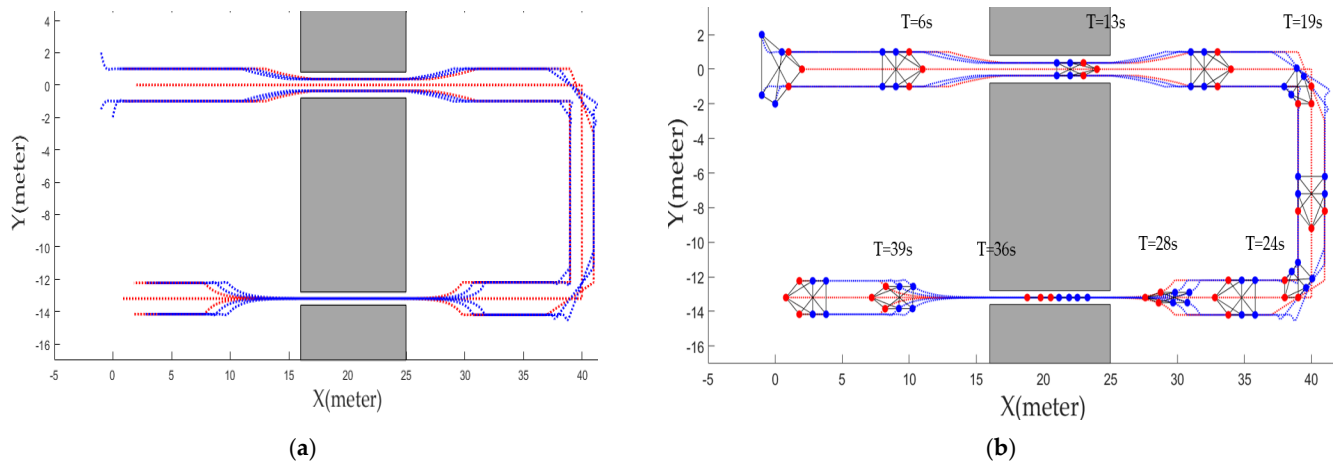


**Figure 5.** Affine-based predefined configuration patterns for illustrations: (a) predefined path; (b) predefined configuration.

#### 4.1.2. DMPC-Based Reference Generation

While each agent shares assumed state trajectories with neighbors in DMPC techniques (prediction/optimization/feedback module), there is a problem of finite horizon that limits the algorithm's prediction capability. Thus, the affine-based global trajectory sequences  $\{p_t^*(k)\}$  from Section 4.1.1 are designed as the input into the feedback module and local environment  $\{p_o, v_i\}_{i=1}^n$  in the prediction module. In addition, the feedback is carried out by the predefined state value  $\{p_d, v_d, u_d\}_t^n$ .

This paper proposes a two-layer cost function in the optimization module to generate the reference sequences, denoted as  $u_{i,t}^* = \underset{X^*}{\operatorname{argmin}} J(X^*)$ . The sets of constraints  $C_{input}$ ,  $C_{output}$  are designed as follows:

$$C_{input} = \begin{cases} v_{min} \leq \|v_i\| \leq v_{max} \\ u_{min} \leq u_i \leq u_{max} \end{cases}, \quad (4)$$

where the limits of  $v_{min}$  and  $v_{max}$  are given by the motion capabilities of the UAVs.  $R_a$  and  $R_s$  denote the safety distance of the agent and the obstacle.

$$C_{output} = \begin{cases} \|p_j(t) - p_i(t)\| \geq R_s \\ \|p_o(t) - p_i(t)\| \geq 2R_a \end{cases}. \quad (5)$$

The cost function is designed by penalizing on the errors between the predicted and the affine-based global trajectory sequences  $\{p_t^*(k)\}$ . The optimization module takes  $p_d^*$  as the optimization variable, obtained as  $u_d^* = \underset{p_d^*}{\operatorname{argmin}} J(X^*)$ , where a two-layer  $J(X^*)$  is designed consisting of two components as below:

$$J(X^*) = \lambda_1 \left( \min \left\{ 0, \frac{\operatorname{dist}(p_i^*, p_j^*) - R_s}{\operatorname{dist}(p_i^*, p_j^*) - R_a} \right\}^2 \right) + \lambda_2 \left( \min \left\{ 0, \frac{\operatorname{dist}(p_i^*, p_{obs}) - R_s}{\operatorname{dist}(p_i^*, p_{obs}) - R_a} \right\}^2 \right) + \sum_{j \in N^L, j \neq i} [(v_i - v_j)^2 + (v_i - v_d)^2] \quad (6)$$

where  $\lambda_1$  and  $\lambda_2$  represents the adjusting factor. From the above Formulation (6), it can be seen that this cost function is divided into two terms, which mainly describes the avoidance function in distance and the adjustment of average velocity. The purpose of utilizing these two terms is to reflect this method's ability to generate a more flexible trajectory from the collision among the agents and obstacles. The details of the description are as follows: the

first term uses the predefined trajectory  $p_i^*$ , agents' neighbor  $p_j$  (considered as dynamic obstacle) and the obstacles  $p_{obs}$  to evaluate the trajectory's feasibility in the environment at current time step  $t$ . The second term represents the agents' velocity  $v_i$  dynamically adjusted by utilizing the weighted average velocity of its neighbors  $v_j$  and the predefined velocity  $v_d$ . Then, a complete reference path is generated by the optimal output  $u_d^*(p_i^*(k) \rightarrow p_d^*)$  in double-integrator (1) form. Formulation (6) describes a two-layer cost function to obtain the optimal reference sequences through the minimum value (Algorithm 1).

---

**Algorithm 1:** Trajectory Reference Generation
 

---

```

1 Initialization  $C_{input}, C_{output}, N, T, A, B, p_0, u_{max}, v_{max}, u_{min}, v_{min}$ 
2 Defining the initial state of formation  $\{X_k^i | p_k^i, v_k^i, p_t^*(k)\}, i \in V$ , the affine image  $I(r)$ ,
the changing matrix  $[A, b]$  and trajectory sequences  $\{p^*(t)\} = [I_n \otimes A(t)]r + I_n b(t)$ 
3 while Iter  $K_{iter} \leq 4500$  and not ReachTargetPosition do
4   BroadcastLastPrediction()  $\rightarrow \Sigma_{k-1}$ 
5   for  $k = 0, 1, \dots, +\infty$  do
6     for all agent  $i \in N = 1, 2, \dots$  do
7        $\{u_i | k\} \leftarrow$  Reset Input ( $\{X_k^i | p_k^i, v_k^i, p_t^*(k)\}$ )
8       Input  $\{X_k^i | p_k^i, v_k^i, p_t^*(k)\}$ 
9       Calculate  $J(X^*)$  and store  $\{u_d^*, X_k^i\}$  (Broadcast)
10      Update  $X_k^i$ 
11      if  $C_{output} \notin \emptyset$  then
12        LocalControl  $\rightarrow$  BuildCollisionCost ( $J(X^*)$ )
13        NegativeFeedback  $\rightarrow \Delta(u_d, v_d, p_d)$ 
14        DMPCsolve  $\rightarrow \{p_d^* | k\}, \min J(X^*)$ 
15      else
16        Predict  $\{X_k, \Sigma_{k+1} | t = k, k+1, \dots, k+N\}$ 
17        DMPCsolve  $\rightarrow \{p_d^* | k\}$ 
18      end if
19       $\{u_{d,k}^*\} \leftarrow$  UpdateReferenceOutput
20    end
21  end for
22  ReachTargetPosition  $\rightarrow$  CheckReach ( $X_i, p_i^*$ )
23   $k \rightarrow k+1$ 
24 end while
  
```

---

**Remark 1.** The  $J(X^*)$  formulation requires only its neighboring nodes, making it adaptable to a variety of communication topologies. However, the algorithm demonstrates effectiveness when the necessary condition of strong interconnectivity within the formation fails in real-world scenarios.

**Remark 2.** Note that an accurate multi-agent model is necessary in problem  $J(X^*)$  to output future behavior, while in real-world scenarios, issues of asymptotic stability, robustness against model uncertainty, noise and coupling constraints could only be solved in this paper to some extent. Thus, the design of Equation (6) would mainly concentrate on the discrete dynamic points instead of the continuous ones.

The objective of reference generation is to serve as the reference baseline for the following critic network and reward function in the MADRL algorithm.

#### 4.2. Improved Multi-Agent Proximal Policy Optimization

The research task is modeled as decentralized partially observable Markov Decision Processes (DEC-POMDP) with shared rewards. A DEC-POMDP in multi-agent systems is defined as  $M = \langle S, \{\mathcal{A}\}, O, \{\Omega_i\}, r, \rho, N, \gamma \rangle$ , where  $s \in S$  is the state space of the environment,  $\mathcal{A}$  is the joint action space with each agent's individual action space as  $\mathcal{A} = \prod_{i \in N} a_i$ .  $O(o|s, a)$  are the observation probabilities for each agent with a given state  $p_i s$ .  $\rho(s'|s, \mathcal{A})$  ( $S \times \mathcal{A} \times S' \rightarrow [0, 1]$ ) represents the transition probabilities function when  $s(s_t \in S) \rightarrow s'(s_{t+1} \in S')$ .  $r_i(s, a_i)$  denotes the agent's reward function.  $\gamma \in [0, 1]$  is the discount factor for calculating the future rewards. The fundamental process is as follows: agents use joint policy  $\pi(a_t|o_t) \equiv \{\pi_1, \pi_2, \dots, \pi_N\}$  parameterized by  $\theta$  to execute the joint action  $\mathcal{A}$  to the local observation. Then, updating each agent's policy based on the received reward and the observed state transition  $s(s_t \in S) \rightarrow s'(s_{t+1} \in S')$ . The final objective is to maximize the expected cumulative return of each agent at each time step by optimizing their strategies during the iterated interactions with the environment, as  $J(\theta) = \mathbb{E}_{a,s}[\sum_t \gamma^t R(s_t, a_t)]$ . Algorithm which is described in the end of Section 4.

As for the MAPPO algorithm, it is a policy-based gradient-type method with two separate networks (policy/value) that extends the PPO algorithm from a single-agent to multi-agent system. It utilizes the centralized training with decentralized execution (CTDE) framework, shown in Figure 6. Thus, each agent owns its own actor/critic networks and updates optimal policy, which is typically parameterized with parameters  $\theta$  in the actor network, based on the commonly used formulation:

$$\Delta\theta = \hat{\mathbb{E}}_t[\nabla_{\theta} \log \pi_{\theta}(a_n^t | s_n^t) \hat{A}(s_n^t, a_n^t)], \quad (7)$$

where  $\hat{\mathbb{E}}_t[\dots] = 1/N \sum_{n=1}^N \sum_{t=1}^{T_n} (\dots)$  is the average sample of  $\mathbb{E}_{\tau}[\dots]$ .  $\hat{A}(s_n^t, a_n^t)$  is the estimation of advantage function  $A_{\pi}(s_n^t, a_n^t)$ , which can be estimated by the state value function  $V_{\pi}(s_t)$  in the critic network with parameters  $\omega$ , calculated as follows:

$$A(s_t, a_t) \approx r_t + \gamma * V_{\omega}(s_{t+1}) - V_{\omega}(s_t), \quad (8)$$

where  $V_{\omega}(s_t)$  is the parameterized critic of the  $V_{\pi}$  estimation, and the parameter  $\omega$  is updated by minimizing the loss function as follows:

$$\arg\min_{\omega} L(\omega) = \frac{1}{N} \sum_{n=1}^N \sum_{t=1}^{T_n} [V_{\omega}^{\pi}(s_n^t) - \delta_n^t]^2, \quad (9)$$

where  $\delta_n^t = r_t + \gamma * V_{\omega}^{\pi}(s_{t+1})$ , and  $\bar{\omega}$  is updated with the recent  $\omega$  to stabilize the learning process in target network. Thus, two parameters  $\theta$  and  $\omega$  are updated iteratively by interacting with the environment. Normally, the value function  $V_{\pi}(s_t)$  is often designed as the baseline to be used in estimating  $A_{\pi}(s_n^t, a_n^t)$  as follows:

$$\hat{A}(s_n^t, a_n^t) = A(s_n^t, a_n^t) - b(s_n^t), \quad (10)$$

where  $b(s_n^t) = V_{\omega}(s_t)$  is the baseline to reduce the variance of the gradient by providing constant values for the state, making the policy update smoother. However, using state value function directly as the baseline limits the policy improvement of ignoring state-action pair information. Thus, this paper proposes a novel way of utilizing the gradient of state error  $e_t$  as the baseline, named as State Error Metric (SEM) and denoted as follows:

$$\hat{A}(s_n^t, a_n^t) = -\beta e_t - \alpha \frac{de_t}{dt}, \quad (11)$$

where  $\beta$  and  $\alpha$  are the hyperparameters. The state error is denoted as follows:

$$e_t = \|s_t - \hat{s}_t\|, \quad (12)$$

where the current agents' state  $s_t = [p_t, v_t, a_t]^T$ , and  $\hat{s}_t$  represents the reference state offered by DMPC (6). Formulation (11) can reflect the level at which the current policy matches the state of the goal and adjust that directly in the tracking direction by measuring the two configurations' difference. Compared with the traditional baseline, this would mainly focus on the current state change on local observation rather than the historical data to estimate.

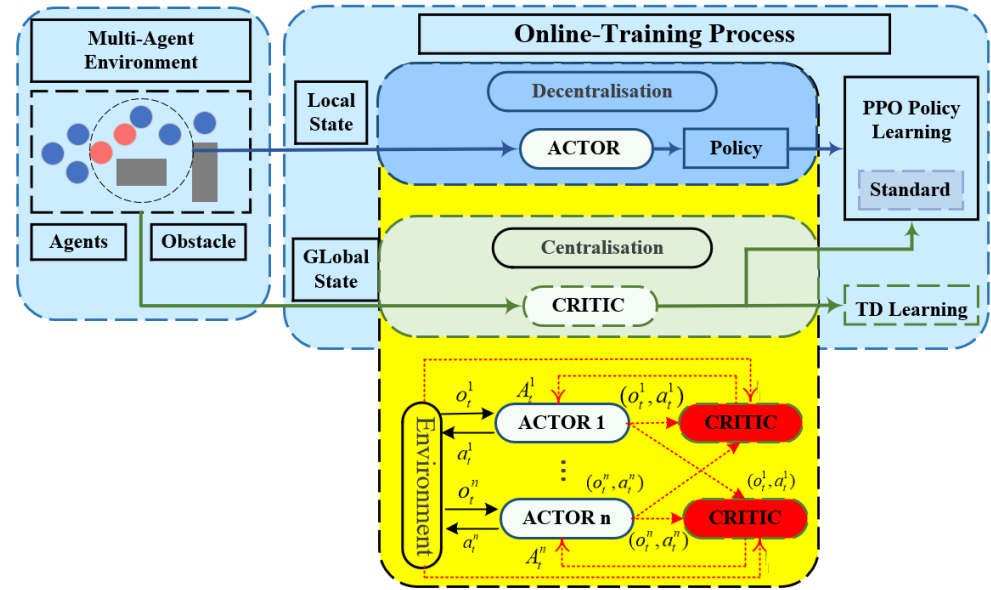


**Figure 6.** Illustration of the MAPPO-CTDE framework.

#### 4.2.1. Action Space

This paper considers the continuous action space of the agent, denoted as velocity vector ( $\hat{a}_t^n = (\hat{v}_t^n, \hat{\omega}_t^n) = [v_{x,t}^i, v_{y,t}^i, \omega_{x,t}^i, \omega_{y,t}^i]^T$ ).  $[v_{x,t}^i, v_{y,t}^i]^T$  represents the linear velocity obtained from the APF technique, policy gradient function for the angular velocity  $[\omega_{x,t}^i, \omega_{y,t}^i]^T$ . Furthermore, the output for each agent within the action space signifies the potential information, shown in Figure 7, and is conveyed to the reward function for evaluation in Section 4.4. Firstly, the APF technique is shown as below:

Let  $U_{ij}(\|p_{ij}\|)$  represent a non-negative smooth pairwise artificial potential function associated with the relative distance between agent  $(i, j)$ . It governs the collective and individual movement within the swarm, which satisfies the following:

$$\begin{cases} U_{ij}(\|p_{ij}\|) = \begin{cases} a \ln(\|p_{ij}\|) + \frac{b}{c(\|p_{ij}\|)^2} & \|p_{ij}\| \leq d_{sen} \\ a \ln(r) + \frac{b}{c(r)^2} & \|p_{ij}\| > d_{sen} \end{cases} \\ U_{ob} = \sum_{k=1}^m \frac{b}{c(\|p_{io}\|)^2}, (b, c)_{ob} \end{cases} \quad (13)$$

where  $i, j = 1, 2, \dots, N, i \neq j$ ,  $a$ ,  $b$  and  $c$  are positive constants,  $U_{ij}(\|p_{ij}\|)$  is bounded and continuously differentiable over interval  $\nabla[b/(\|p_{ij}\|)^2]$ .  $\nabla[b/(\|p_{ij}\|)^2]$  describes the gradient negative cube odd function, which is not defined at 0 point. Therefore,  $U_{ij}(\|p_{ij}\|)$

would obtain its boundary maximum value  $U_0$  when  $\|p_{ij}\| \rightarrow 0$ ; minimum  $U_d$  when  $(i, j)$  satisfies the desired position, that is,  $\|p_{ij}\| = d$ ; the constant  $U_r$  when  $\|p_{ij}\| > d_{sen}$ .  $U_{ob}$  represents the obstacles' repulsion, which could generate a repulsive force solely to prevent collisions.

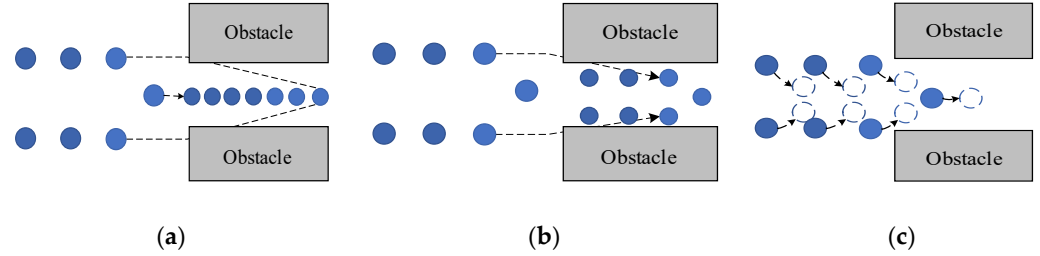


**Figure 7.** Illustration for the potential position generated by action space. (a) Obstacle collision in linear style. (b) Obstacle collision in scaling style. (c) Potential positions. The agent is blue, obstacles are gray, and potential positions are dotted lines.

Equation (13) effectively captures the essential forces of attraction and repulsion that exist between agents and between agents and obstacles.  $a$ ,  $b$  and  $c$  are the positive constants. Thus, the output from the action space is denoted as  $\hat{u}_{local}^i$ :

$$\begin{aligned} \hat{u}_{local}^i = & \underbrace{-\sum_{j=1, j \neq i}^N \nabla_i [\lambda_{ij} U_{ij}(\|\tilde{p}_{ij}\|) + \lambda_{io} U_{io}(\|\tilde{p}_{io}\|)]}_{f_i^p} \\ & + \underbrace{C_0 \sum_{j=1, j \neq i}^N (\tilde{v}_{ij} + \tilde{v}_{id})}_{f_i^v} + \underbrace{C_1 \sum_{j=1, j \neq i}^N (\tilde{\omega}_{ij} + \tilde{\omega}_{id})}_{f_i^\omega}, \end{aligned} \quad (14)$$

where  $C_0$  and  $C_1$  are the constant value, and  $\lambda_{ij}$  and  $\lambda_{io}$  are different spring constants.  $f_i^p$  is the gradient-based term, which regulates position consensus by employing local repulsive and attractive forces, ensuring collision avoidance and cohesion maintenance, and  $f_i^v$  means the agent's velocity is adjusted by utilizing the weighted average velocity of its neighbors and desired velocity as a reference, same as  $f_i^\omega$ .

Each agent's potential position is determined by following the steepest gradient toward the geometric point where the sum of potential forces is zero. Agents then adapt and locally track the potential trajectory, effectively integrating trajectory planning with formation control. Subsequently, the calculated potential position is discretized to facilitate updates to the local trajectory, as shown in Figure 8.

It is depicted as Figure 8, where the agent interacts with other agents and obstacles (with d-react), which generate the two kinds of artificial potential forces  $U_{ij}$  and  $U_{io}$  to obtain the fast gradient descent direction  $\nabla U_t^i$  to the potential position. In addition, the policy gradient determines the angular velocity  $\omega_\theta$  to adjust the agent's trajectory towards the potential position from its current direction. Based on Equation (7), the angular velocity  $\omega_\theta = f(s_t) = f(p, \theta, v)^n$  and the updated parameter of the policy gradient is  $\theta' = \theta + \alpha \Delta \theta$ , where  $\Delta \theta$  is obtained from Equation (9). Thus, the updated angular velocity is  $\hat{\omega}_t^{n'} = \pi_{\theta'}(a | s)$ .

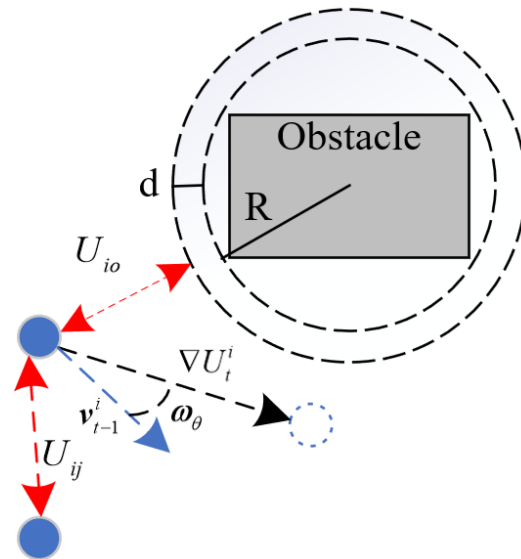


**Figure 8.** Illustration for potential position (red dotted circle) of one agent on local level. The forces in red dotted line among agents and obstacle. Potential gradient in black dotted line. The velocity direction in blue dotted line.

#### 4.2.2. Action Selection

The main problem would be the selection of the potential action vector  $\hat{a}$ ; the components within action space  $\hat{a}_t^n = (\hat{v}_t^n, \hat{\omega}_t^n)$  are all defined as continuous variables with a certain range. At each time step  $T_i$ , the action of an agent is expressed as follows:

$$\hat{a}_t^n = (\hat{v}_t^n, \hat{\omega}_t^n), (\hat{a}_t^n \in \mathcal{A}, \forall n \in \mathcal{N}), \quad (15)$$

$$\hat{a}_t^n = (a_t^1, a_t^2, \dots, a_t^N), \quad (16)$$

$$\begin{cases} v_{\min} \leq \|\hat{v}_t^n\| \leq v_{\max} \\ u_{\min} \leq \hat{u}_t^n \leq u_{\max} \\ |\hat{\omega}_t| \leq \hat{v}_t / R_{\min} \end{cases}, \quad (17)$$

- Action of linear velocity  $\hat{v}_t^n$ : determined by the APF function from action space  $\hat{u}_{local}^i$ ;
- Action of angular velocity  $\hat{\omega}_t^n$ : obtained from the policy gradient function  $\nabla \pi_\theta$  in Actor module.

Supposing a disordered swarm of  $N$  agents is steered by protocol (14) in linear velocity, if the interaction between agents within the graph  $(G, p)$  is undirected and connected for the whole time,  $Z_0$  is denoted as the sum of total artificial potential; the lemma and proof are as follows.

**Lemma 1.** The graph  $(G, p)$  is connected mutually if

$$Z_0 < U_r \text{ for all } t \geq 0. \quad (18)$$

**Proof.** It is assumed that  $M(1 \leq M \leq N)$  agents function as leaders. Thus, the kinematic error of the leaders in the system is defined as follows:

$$\begin{cases} \tilde{p}_i = p_i - p_d, i = 1, 2, \dots, M, \\ \tilde{v}_i = v_i - v_d \end{cases} \quad (19)$$

based on the fact that the global trajectory is twice differentiable, it satisfies  $\dot{\mathbf{p}}_d = \mathbf{v}_d$ ,  $\dot{\mathbf{v}}_d = \mathbf{u}_d$  and we have  $\tilde{\mathbf{p}}_{ij} = \tilde{\mathbf{p}}_i - \tilde{\mathbf{p}}_{j'}$ ,  $\dot{\tilde{\mathbf{p}}} = \tilde{\mathbf{v}}_i$ ,  $\dot{\tilde{\mathbf{v}}} = \tilde{\mathbf{u}}_i$ . Given that  $Z_t$  represents the summation of the total relative kinetic energy of an agent at time  $t$ , it can be deduced that

$$Z(t) = 1/2 \sum_{i=1, i \in O}^N \left\{ [U_i(\|\tilde{\mathbf{p}}_{ij'}\|)] + C_2(\mathbf{p}_i - \mathbf{p}_d)^T (\mathbf{p}_i - \mathbf{p}_d) + C_3(\mathbf{v}_i - \mathbf{v}_d)^T (\mathbf{v}_i - \mathbf{v}_d) \right\}, \quad (20)$$

let  $V_i = \sum_{i=1, i \in O}^N \left\{ \tilde{\mathbf{p}}_i^T \tilde{\mathbf{p}}_i + U(\|\tilde{\mathbf{p}}_i\|) \right\}$  be the total artificial of agent  $i$ . Then, in the error system,  $Z(t)$  can be rewritten as follows:

$$Z(t) = \frac{1}{2} \sum_{i=1, i \in O}^N [V_i + \tilde{\mathbf{v}}_i^T \tilde{\mathbf{v}}_i], \quad (21)$$

it can be easily demonstrated that  $Z(t)$  is positive semi-definite, making it a suitable candidate function for Lyapunov analysis. Considering the symmetry of  $U_{ij}$  with respect to  $\tilde{\mathbf{p}}_{ij}$ , we have the following:

$$\frac{\partial U_i}{\partial \tilde{\mathbf{p}}_{ij'}} = \frac{\partial U_i}{\partial \tilde{\mathbf{p}}_i} = -\frac{\partial U_i}{\partial \tilde{\mathbf{p}}_{j'}}, \quad (22)$$

thus, we have

$$\frac{d}{dt} \left( \frac{1}{2} \sum_{i=1, i \in O}^N V_i \right) = \sum_{i=1, i \in O}^N \left\{ \tilde{\mathbf{v}}_i^T (\nabla_{\mathbf{p}_i} U_i) + C_2 \tilde{\mathbf{v}}_i^T \tilde{\mathbf{p}}_i \right\}, \quad (23)$$

where  $U_i = \sum_{i \in O} U_{ij'}(\|\tilde{\mathbf{p}}_{ij'}\|)$ , and  $j'$  means other agents and obstacles. Then, the derivative to time  $t$  for  $Z(t)$  is obtained as follows:

$$\dot{Z}(t) = \sum_{i=1, i \in O}^N (\nabla_{\mathbf{p}_i} U_i) + C_2 \tilde{\mathbf{v}}_i^T \tilde{\mathbf{p}}_i + \tilde{\mathbf{v}}_i^T \dot{\tilde{\mathbf{v}}}_i, \quad (24)$$

Thus, via substituting (27) to (32), we might have the following:

$$\dot{Z}(t) = \sum_{i=1}^N \tilde{\mathbf{v}}_i^T \sum_{j=1, j \neq i}^N C_1 a_{ij} (\mathbf{v}_{ij} - \mathbf{v}_{iv}) + \sum_{i=1}^N C_3 \tilde{\mathbf{v}}_i^T \dot{\tilde{\mathbf{v}}}_i, \quad (25)$$

and Equation (31) can be expressed in matrix form as follows:

$$\dot{Z}(t) = -\tilde{\mathbf{v}}^T [(C_1 Y(t) + C_3 \Omega(t)) \otimes I_n] \tilde{\mathbf{v}}, \quad (26)$$

where  $\Omega(t)$  is the stress matrix,  $Y(t) = \text{diag}[\gamma_1, \gamma_2, \dots, \gamma_N]$ ,  $\tilde{\mathbf{v}} = (\tilde{\mathbf{v}}_1^T, \tilde{\mathbf{v}}_2^T, \dots, \tilde{\mathbf{v}}_N^T)^T$ . Apparently,  $\Omega(t)$  and  $Y(t)$  are positive semi-definite, and these two factors would explain that  $Z(t \geq 0)$  is a non-increasing function with  $\dot{Z}(t) \leq 0$ .

Thus, the level set  $\psi = \left\{ [\tilde{\mathbf{p}}^T, \tilde{\mathbf{v}}^T]^T \in \mathbb{R}^{2nN} : Z(t) \leq Z(0), t \geq 0 \right\}$  can be seen as a positively invariant set. Then, it is supposed that  $Z(0) < U_r$ , and the equation can possibly meet  $U_i(\|\tilde{\mathbf{p}}_{ij'}\|) \leq Z(t) \leq Z(0) < U_r$  ( $i \in O, t \geq 0$ ) in each edge  $[e_{ij}]$ . Therefore, the connectivity of graph  $G$  of all agents  $\mathbf{p}_i$  for swarm is shown within  $t \geq 0$  and  $i \in O$ ; Thus, the Lemma 1 has been proved.  $\square$

**Remark 3.** Despite the similarities between certain terms in Equation (14) and existing protocols, the linear speed of the formation guided by Equation (14) would be distinctively unique. This is due to the presence of the affine-based trajectory of each agent that can serve as an evaluation index against the local trajectory.

**Remark 4.** Although the paper does not specifically analyze the impact of acceleration and turning rate as inputs, it is important to note that these two factors are influenced by the agents themselves. Therefore, assuming that the initial distance and interaction range between agents meet the required

range,  $Z(0) < U_r$  and  $Z(0) < U_0$  can effectively avoid collisions within the agent formation and ensure overall stability.

#### 4.3. Observation and State Space

The local observations  $O_i$  of each agent primarily include the distances and velocity of itself  $i$  and the neighbors  $N_i$  rather than the global information. Thus, each agent's policy decision is informed by the current observation as  $O_i^t$  or the hidden state (RNN) at first in the time step  $T_i$ , denoted as  $O_i^t = \{(p_i^t, v_i^t, \omega_i^t), (p_j^t, v_j^t, \omega_j^t)\}$ , where  $(p_i^t, v_i^t, \omega_i^t)$  is the state itself and  $(p_j^t, v_j^t, \omega_j^t)$  is the neighbors'  $N_i$  observation. The local observation information  $O_i^t$  is sent to each agent at the start of each time step  $T_i$ .

To enhance the agent's comprehension of the dynamic environment and to acquire the appropriate formation configuration, we delineate the state space into distinct components: the current environmental state and the predefined configuration state. The state space  $S$  can be illustrated as below:

$$S = \{s, s_d(t)\} = \begin{cases} s = \{(p_i, v_i, \omega_i), (p_{ij}, p_{id}, p_{io}, v_{ij}, v_{id})\} \\ s_d(t) = \{[p_d^*], [(p_d, v_d)] | p_i^*(t) = A(t)r_i + b(t)\} \end{cases}, \quad (27)$$

where  $(p_i, v_i, \omega_i)$  signifies the agent's position and velocity, while  $(p_{ij}, p_{id}, p_{io}, v_{ij}, v_{id})$  denotes the distances between agents, the agents and their desired trajectory, and the agents and any obstacles. Furthermore,  $(p_{ij}, p_{id}, p_{io}, v_{ij}, v_{id})$  also encompasses the differences in velocity between the agent and its neighbors, as well as the discrepancies between the agent's velocity and its desired value.

Additionally,  $s_d(t)$  represents the trajectory reference. One of the desired pieces of information is  $\{p_d^*\}$ , which denotes a fixed and predefined formation shape sequence. The other reference information  $(p_d, v_d)$  is derived from the DMPC module.

#### 4.4. Reward Function

In this paper, the reward function serves as a feedback mechanism to guide the agent's learning process and assess the efficacy of its actions, mainly for its generated potential position. The purpose of this process is to settle the local replanning strategy for each agent, which evaluates the generated potential state based on the environment.

This article utilizes a segmented cost function to formulate the reward function, a choice motivated by its potential to accelerate the convergence of network updates, thereby enhancing the agents' learning efficiency in acquiring the optimal strategy. The reward function is designed as  $R_t = -(J_i(\hat{X}_i))$ , whose aim is to use the four-layer cost function to reflect the potential state  $(\hat{X}_i)$  at the time step  $t$ . Thus, the reward of each potential state  $(\hat{X}_i)$  would be transferred by the opposite of the cost function below, and the learning process can be explicitly guided so that the multi-agent system learns in the direction of cost reduction  $-J_i(\hat{X}_i)$ .

The potential motion estimation of the agents is solved as minimization of the cost function  $J_i(\hat{X}_i)$ , subject to sets of constraints  $C_{inputs}$ ,  $C_{agent}$ . The input constraints are as follows:

$$C_{input} = \begin{cases} v_{min} \leq \|\hat{v}_t\| \leq v_{max} \\ |\hat{\omega}_t| \leq v_t / R_{min} \end{cases}, \quad (28)$$

where the limits of  $v_{min}$  and  $v_{max}$  are given by the motion capabilities of the UAVs in the formation. The angular velocity  $\omega_k$  is limited by the minimal allowed turning radius  $R_{min}$

of the formation that ensures smooth movement of all followers in the desired shape.  $R_s$  and  $R_a$  denote the safety distance of the agent and the obstacle.

$$C_{agent} = \begin{cases} \|\hat{p}_j(t) - \hat{p}_i(t)\| \geq R_s \\ \|\hat{p}_o(t) - \hat{p}_i(t)\| \geq 2R_a \end{cases}, \quad (29)$$

The cost function for the agents' potential state can provide feedback continuously during the maneuver. Thus, the four-layer  $J_i(\hat{X}_i)$  is designed as below:

$$\begin{aligned} J_i(\hat{X}_i) = & \sum_{k=1, k \in N} \|\hat{p}_i(k) - p_d^*(k)\|^2 \\ & + \sigma_1 \left( \min \left\{ 0, \frac{\text{dist}(\hat{X}_i, \hat{X}_j) - R_s}{\text{dist}(\hat{X}_i, \hat{X}_j) - R_a} \right\}^2 \right) + \sigma_2 \left( \min \left\{ 0, \frac{\text{dist}_{obs}(\hat{X}_i) - R_s}{\text{dist}_{obs}(\hat{X}_i) - R_a} \right\}^2 \right) \\ & + \sum_{j \in N^L, j \neq i} [(\hat{v}_i - \hat{v}_j)^2 + (\hat{\omega}_i - \hat{\omega}_j)^2 + (\hat{v}_i - v_d)^2] \\ & + \sum_{i=1}^N \text{pen}_i \end{aligned}, \quad (30)$$

where  $\sigma_1$  and  $\sigma_2$  represents the adjusting factor. Equation (30) mainly evaluates four aspects: the position distance to the reference value and the obstacles among agents in formation, the distance among the agents in reference sequences, the velocity distance in each linear and angular way and rotation maneuver penalty. This four-layer cost function totally considers displacement, distance and bearing direction of the agent in each time step, especially the rotation maneuver penalty to judge the turning maneuver during the flight. The details of Equation (30) are as follows.

The first layer penalizes deviations between the reference  $p_d^*(k)$  and the current potential position  $\hat{p}_i(k)$  through the Euclidean distance  $\|\hat{p}_i(k) - p_d^*(k)\|^2$ , which would show the basic relevant position at the current moment for all the agents  $k \in [1, N]$ .

The second layer represents the avoidance function between the potential state  $\hat{X}_i$  and its neighbors  $\hat{X}_j$ , treating other agents as dynamic obstacles  $\text{dist}(\hat{X}_i, \hat{X}_j)$ , while the other component indicates the distance to the nearest obstacle  $\text{dist}_{obs}(\hat{X}_i)$ .

The third layer represents the potential agents' velocity  $\hat{v}_i$  and  $\hat{\omega}_i$ , adjusted by utilizing the weighted average velocity of its neighbors  $(\hat{v}_j, \hat{\omega}_j)$  and the desired velocity  $v_d$  as a reference to adapt the flight direction. In addition, the angular velocity is mainly set to enhance the flight smoothness; thus, the angular velocity of the agent can only be determined by that of neighbors  $(\hat{\omega}_i - \hat{\omega}_j)^2$ .

The final layer, known as the rotation maneuver penalty  $\text{pen}(i = 1, \dots, N)$  for the formation, imposes a cost on turning maneuvers to adjust the turning rate of the agent in the event of a sudden change in velocity. This term can be utilized in scenarios where trajectories devoid of turning maneuvers are favored, even though it results in a longer time to reach the target. In this paper, a moment estimation method is assigned to this penalty term, and defined as follows:

$$\begin{aligned} \text{pen} &= E[U^2] - E[U]^2 \\ &= \left( \frac{1}{N_s} \sum_{i=1}^N U_i^2 \right) - \left( \frac{1}{N_s} \sum_{i=1}^N U_i \right)^2, \\ &= \frac{1}{N_s} \|U\|_2^2 - \frac{1}{N_s^2} \|U\|_1^2 \end{aligned}, \quad (31)$$

where  $E[\cdot]$  is mathematic expectation, and  $N_s$  is the number of sampling points. The squared distance vector  $U \in \mathbb{R}^N$  is defined as follows:

$$U = (\|\hat{\omega}_{i,2} - \hat{\omega}_{i,1}\|_2^2, \dots, \|\hat{\omega}_{i,N_s} - \hat{\omega}_{i,N_s-1}\|_2^2), \quad (32)$$

In addition, based on the assumption that the graph information of each sampling point is independent, the gradient of  $pen$  concerning  $\hat{\omega}_{i,j}$  is calculated as follows:

$$\begin{aligned} \frac{\partial pen}{\partial \hat{\omega}_{i,j}} &= \frac{4}{N_s} (U_{n-1} - \frac{\|U\|_1}{N_s}) (\hat{\omega}_{i,j} - \hat{\omega}_{i,j-1}) \\ &\quad - \frac{4}{N_s} (U_n - \frac{\|U\|_1}{N_s}) (\hat{\omega}_{i,j+1} - \hat{\omega}_{i,j}) \end{aligned} \quad (33)$$

where  $U_n \in U$ , and the quasi-Newton method [32] is employed to solve the proposed optimization problem and would generate an optimal formation sequence to feedback the agent.

Thus, the reward function (30) is designed to evaluate the current experience between the state and the action, where the agent can improve the strategies' learning effect  $\pi_\theta$  to instruct the next state  $s_{t+1}^i = (p, v, \omega)_t^i$  (Algorithm 2).

---

**Algorithm 2:** Improved Multi-Agent Proximal Policy

---

```

1 Initialize:  $\theta$  and parameters for policy  $\pi_\theta$ , and  $\phi$  for critic  $V(s)$ ,  $p_d^* \forall n$ .
2 Initialize: current policies  $\pi_{t_0}^n$  ( $\theta_{t_0}^n \leftarrow \theta^n$ ), target critic  $Q_{\bar{w}^n}$  ( $\bar{w}^n \leftarrow w^n$ )
3 Initialize: learning rate  $l_\alpha$ 
4 while Iter  $K_{iter} \leq K_{itermax}$  and not ReachTargetPosition do
5   SetCircularBuffer  $D = \{\}$  and SetEmptyList  $\eta = []$ 
6   Initialize ActorRNN  $h_{0,\pi}^1, h_{0,\pi}^2 \dots h_{0,\pi}^n$ 
7   Initialize CriticRNN  $h_{0,V}^1, h_{0,V}^2 \dots h_{0,V}^n$ 
8   for an episode  $t = 1$  to  $T$  do
9     each agent  $i(1, 2, \dots, n)$ 
10    ProbabilityObtain  $(p_t^i, h_{t,\pi}^i) \leftarrow \pi_t^i(o_t^i | h_{t-1,\pi}^i; \theta)$ 
11    ActionSelection  $u_t \leftarrow p_t^i$ 
12    ValueFunctionObtain  $(v_t^i, h_{t,V}^i) \leftarrow V(s_t^i, h_{t-1,V}^i; \phi)$ 
13  end for
14  ActionExecution  $u_t \leftarrow \pi_t^n(a_t^n | s_t^n)$ , ObservationObtain  $\rightarrow r_t, s_{t+1}, o_{t+1}$ 
15  TrajectoryObtain  $\tau = \{s_t, o_t, h_{t,\pi}, h_{t,V}, u_t, r_t\}$ 
16  AdvantageCompute (SEM)  $\leftarrow A^n(s^n, a^n)$ , RewardCompute  $\leftarrow \hat{Q}^n(s^n, a^n)$ 
17  TrajectorySplit  $\rightarrow \tau/L$ , DataStore  $\{\hat{Q}^n(s^n, a^n), A^n(s^n, a^n), a^n, o^n\}_{n=1}^n \rightarrow D$ 
18  Randomize  $\rightarrow D'$ 
19  for  $k = 1$  to  $M$  do
20     $B \leftarrow D'$  with all agent data
21    GradientCompute  $\rightarrow \Delta\theta^n = \{\nabla_{\theta^n} f(r_k(\theta^n), A_k^n)\}_{k=1}^{M/B}$ 
22     $\rightarrow \Delta w^n = \{\nabla_{w^n} (Q_k^n - V_{w^n}(o_k^n))^2\}_{k=1}^{M/B}$ 
23    for  $b \leftarrow$  recurrent data block do
24      Update RNN  $\rightarrow h_{\pi}^i, h_V^i$ 
25    end for
26  end for
27  AdAMUpdate  $\theta_{t_0}^n \leftarrow \theta^n \leftarrow$  GradientAscent
28   $\bar{w}^n \leftarrow w^n \leftarrow$  GradientDescent
29  Empty  $\rightarrow D'$ 
30 end while

```

---

**Remark 5.** The way of using cost functions is technically to measure undesirable system behavior or penalize errors; however, the non-zero values would be produced under certain conditions, especially when the sudden stall occurs, which can lead to sparse reward signals. Furthermore, this

approach would not resolve the nonlinear problems associated with the multi-agent system and could compromise robustness by excessively minimizing the cost function.

## 5. Simulation and Analysis

### 5.1. Experimental Setup

The experiment was performed on a desktop machine with 256 GB RAM, one 64-core CPU and one GeForce RTX 3090 GPU (ThinkStation P-, Hefei, China), and the whole framework of the proposed algorithm is shown in Figure 9. The whole experiment was on a two-dimensional plane as  $[5000 \text{ m} \times 5000 \text{ m}]$ . There were  $N = 4 - 7$  agents in the plane as a comparison experiment, each with a radius of sensing range  $d_{sen} = 30 \text{ m}$ . The initial formation configurations were set with different numbers of agents  $N = 4 - 7$  as Figure 9 shows;  $N_{obs} = 4, 6, 8$  obstacles were set in Pygame library in PYTHON (3.12.2 AMD64), with the constant velocity ( $v_{obs-x} = 5 \text{ m/s}$ ,  $v_{obs-y} = 3 \text{ m/s}$ ) motives in four random directions (up, down, left and right) in the two-dimensional plane. In addition, the direction of any obstacles was reset randomly with the detection of the experimental boundaries for obstacles' positions in each time step, which can ensure the obstacles stay within the experimental field. The agents' positions were randomly selected from the box  $[-50 \text{ m}, 50 \text{ m}] \times [-50 \text{ m}, 50 \text{ m}]$  at the initial stage. The success of configuration change was set as the distance between the center of agent and the obstacle and other agent being less than  $10 \text{ m}$ , when the formation still remains the complete structure. Table 1 lists the main parameters for the environment and agents.

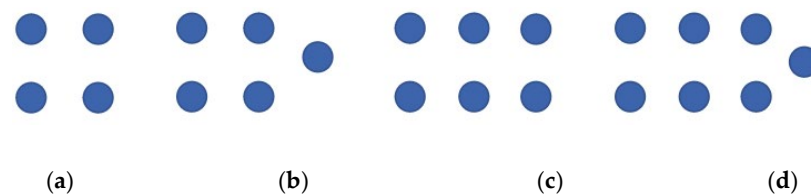


**Figure 9.** Illustration of the different numbers of agents 4–7. (a) 4 agents (b) 5 agents (c) 6 agents (d) 7 agents.

**Table 1.** The parameter settings of multi-agent system.

| Agent                      | Symbol               | Value                               |
|----------------------------|----------------------|-------------------------------------|
| Initial linear velocity    | $v_0$                | 2 m/s                               |
| Desired linear velocity    | $v_d$                | 45 m/s                              |
| Linear velocity            | $v_{\max}, v_{\min}$ | 80 m/s, 5 m/s                       |
| Acceleration               | $u_{\max}, u_{\min}$ | $3 \text{ m/s}^2, -3 \text{ m/s}^2$ |
| Maximum angular velocity   | $\omega_{\max}$      | 10 rad/s                            |
| Predictive horizon         | $N_p$                | 10                                  |
| Minimum distance           | $R_s, R_a$           | 15 m, 20 m                          |
| Obstacle sensing range     | $R_{obs}$            | 20 m                                |
| Obstacle constant velocity | $v_x, v_y$           | 5 m/s, 3 m/s                        |

Other parameters were set as  $a = 1.5$ ,  $b = 2$ ,  $c = 0.5$ ,  $\delta = 0.6$ ,  $\omega_{ij} = 1$ .

It can be seen from Figure 10 that the whole experimental process contains three major parts, the initial multi-agent formation in the partially known environment, the DMPC module and the MARL process.

In the MARL process, the actor network utilizes an RNN and target two-layer network. To observe configuration change more effectively, the environment is set not to terminate upon collision. Instead, it only resets when the agent exceeds the maximum number of steps allowed. The Adam optimizer was used to update the neural network parameters

**Figure 11.** Illustration for the average tracking and acceleration error for the formation trajectory navigation. (a) average tracking error (b) average acceleration. The numbers of 1–7 represent each agent.

It can be seen from Figure 11 that the DMPC method effectively tracks the affine trajectory planning based on the local environment. The algorithm tracks each agent's predefined trajectory, transforming disordered sequences of formation into an ordered one while minimizing error during configuration changes. Figure 11a illustrates that a closer alignment with the predefined trajectory results in smaller errors. The DMPC algorithm practically integrates local environment data to provide a more accurate reference, which would replace the traditional baseline in MARL process. Thus, the I-MAPPO algorithm is tested in the following along with the compared MADDPG.

Then, we evaluate the performance of the proposed algorithms using two key measurements—the average reward and the average number of collisions (note that it contains the numbers reaching the obstacles' radius)—during the learning process under the proposed reference baseline using Equation (11). The two distinct DRL strategies, MADDPG and I-MAPPO, which can both reduce the variance of the policy gradient and enhance continuous action selection. Thus, the experiment analyzes the algorithm's performance with varying numbers of agents and obstacles. The average reward is calculated as the mean reward of all agents from 4 to 7 and obstacles 4, 6 and 8. In all the experiments below, the various numbers of agents and obstacles are designed to analyze the proposed algorithm in different topology (4–7) based on Figure 7. All the experimental values are directly obtained from TensorBoard.

Thus, it is divided into four parts: firstly, using these two algorithms to test the small formation with agents  $N = 4$ , obstacles 4, 6 and 8; then, these two algorithms are designed to test in the normal size formation with  $N = 4$ ,  $N = 5$  agents, obstacles 4, 6 and 8, respectively; finally, the experiment tests the formation with  $N = 7$  agents, obstacles 4, 6 and 8.

These two MARL methods share the similar policy-based (MADDPG-off policy / MAPPO-on policy) learning structure. In small formation, compared to MADDPG and I-MAPPO algorithm, both MADDPG and I-MAPPO achieve higher rewards at the final stage of convergence, indicating that the addition of the reference baseline enhances reward acquisition. In addition, we can know from Figure 12a–c, the I-MAPPO can achieve the convergence faster than MADDPG. Moreover, the I-MAPPO has a better ability of trajectory replanning and configuration reconstruction from oscillations above, especially from Figure 12c, and the formation could replan the local trajectory of each agent to form the new unrigid configuration after encountering the collision at the descent point (Learning step  $T = 800,000$ ).

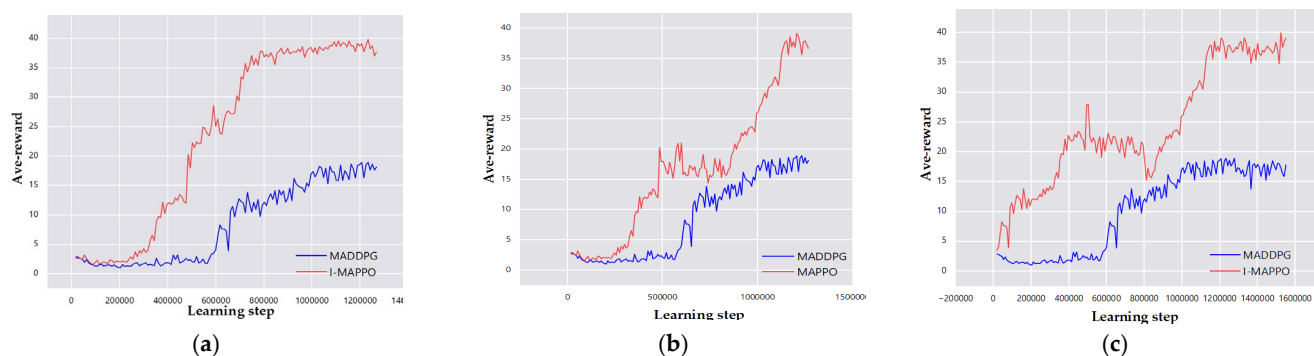


**Figure 12.** An illustration of the average reward per formation learning at agents  $n = 4$ . (a) obstacles 4 (b) obstacles 6 (c) obstacles 8. It is noted that higher rewards are obtained by I-MAPPO in red.

It is demonstrated from Figure 13c that the efficiency of the algorithm could help the formation configuration reconstruct in a very short time, and this algorithm leads the configuration start to form an adaptable one after reaching the collision conditions. Then, the formation would pass through the obstacles with the new unrigid configuration

shape. Figure 13b shows the low speed of online learning of the I-MAPPO algorithm. Figure 13a–c show that although the formation may encounter the obstacle or reach the collision condition, the flight system can still change the configuration to adapt to the current environment. In addition, the I-MAPPO converges faster than the MADDPG.

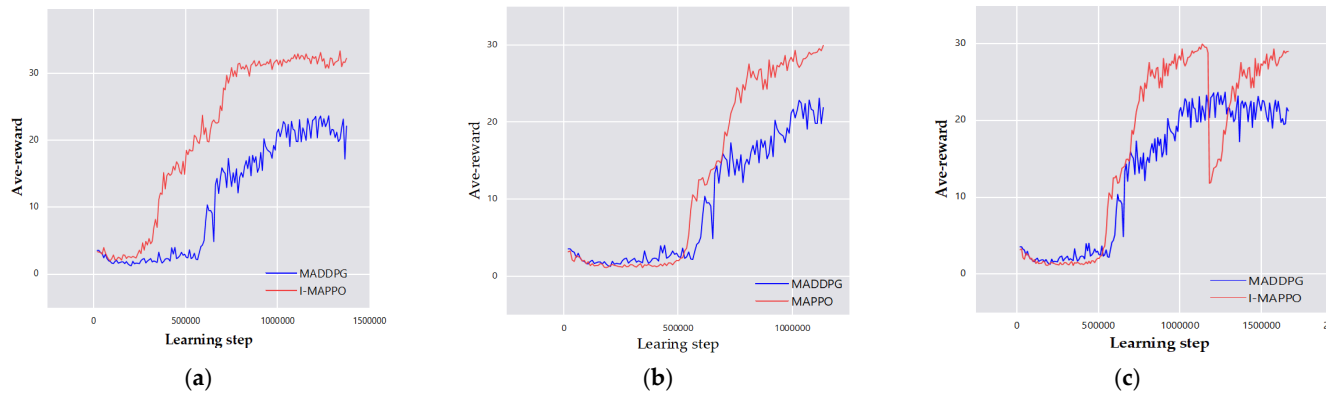


**Figure 13.** An illustration of the average reward per formation learning at agents  $n = 5$ . (a) obstacle 4 (b) obstacle 6 (c) obstacle 8. It is noted that higher rewards are obtained by I-MAPPO in red.

It is shown in Figure 14c that the oscillations increase rapidly along with the growth of the numbers of agents and obstacles. In addition, even though the MADDPG would achieve the high average reward, this algorithm fails to converge in our final step. Instead, the I-MAPPO could lead the formation to the convergence stage. Figure 14b shows a relatively rapid rate of convergence.

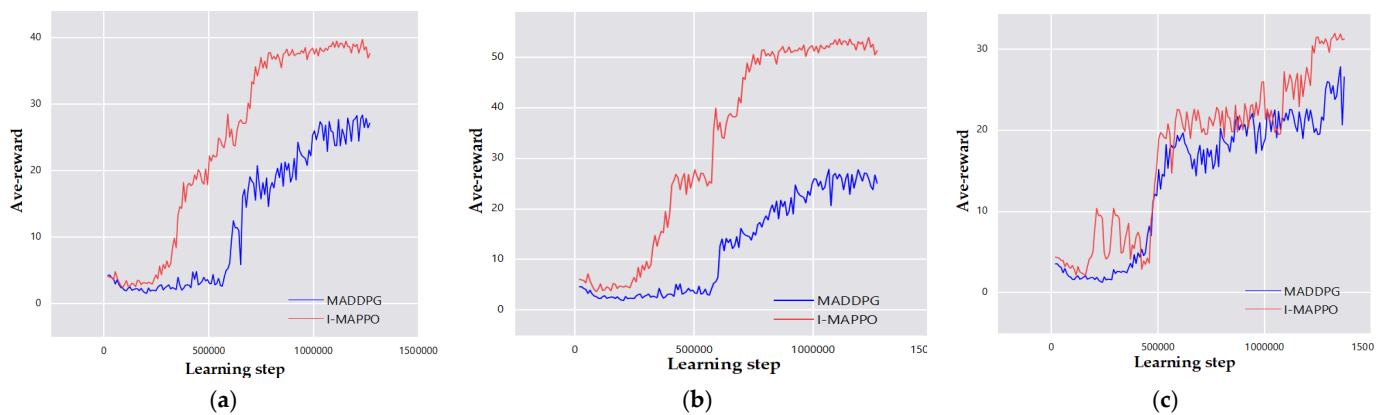


**Figure 14.** An illustration of the average reward per formation learning at agents  $n = 6$ . (a) obstacle 4 (b) obstacle 6 (c) obstacle 8. It is noted that higher rewards are obtained by I-MAPPO in red.

In the experiment from Figure 15, the experimental results show that the two algorithms are affected by the numbers of agents and the obstacles, and the oscillations stay with the flight all along. Thus, this shows that the dense environment (other agents are also considered as the obstacles) would limit the capability of fully autonomous formation flight. From Figure 15a, the effectiveness of two algorithms almost shares the same result, and the result from Figure 15b oscillates the most in this experimental stage. In addition, the reward value is not strongly relevant to the numbers of the agents and obstacles. The reason for I-MAPPO outweighing MADDPG is that the on-policy execution strategy (Equation (15)) enables all agents to choose more optimal continuous actions from a local perspective and the guidance under the reference baseline (Equation (11)), resulting in greater reward feedback.

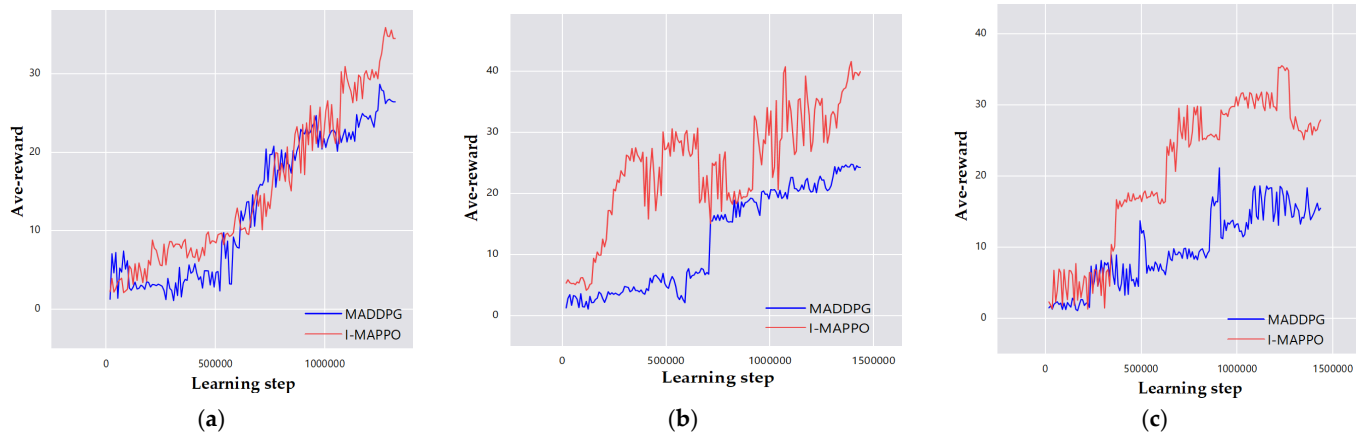


**Figure 15.** An illustration of the average reward per formation learning  $n = 7$ . (a) obstacle 4 (b) obstacle 6 (c) obstacle 8. It is noted that higher rewards are obtained by I-MAPPO in red.

It can be seen from Figure 16 that the average collision numbers show that three algorithms predict the obstacle collision in all learning steps based on the experimental condition above (agents 4–7, obstacles 4, 6, 8). The average trends of each algorithm demonstrate the same collision situation during the learning and training process. In previous work [15], the reason for collision numbers increasing is that the traditional algorithm can perform well in the static environment with the predefined trajectory. In addition, the MADDPG algorithm produces undesirable results due to the offline training based on multiple parameters and the historical experience in memory buffer. The proposed I-MAPPO can achieve better results based on the current adjustable policy.

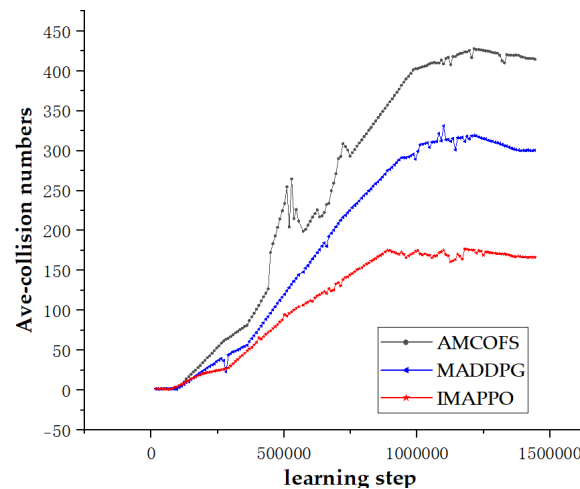


**Figure 16.** Illustration of the average collision numbers (note that they contain the numbers of those reaching the obstacles' radius). It is noted that I-MAPPO obtained lower numbers in learning process.

Then, we compared the traditional baseline (TB) and the proposed baseline-SEM during the learning steps for the two MARL algorithms. As shown in Table 3, the I-MAPPO-SEM used the gradient of state error, which can greatly enhance the ability of the adjusting the local trajectory of each agent. The advantage of this technique (Equation (11)) can be concluded as follows. Using gradient information fits the state error to guide the direction rapidly; meanwhile, the current change in the state error can replace the historical one from the traditional baseline. I-MAPPO-SEM showed the better of the three measurements than I-MAPPO-TB.

**Table 3.** Algorithm comparison in Success rate, Ave-collisions rate and Ave-smooth rate.

| Algorithm   | Success Rate (%) | Ave-Collisions Rate (%) | Ave-Smooth Rate (rad) |
|-------------|------------------|-------------------------|-----------------------|
| I-MAPPO-TB  | 33 ± 5.8         | 49.6                    | 0.4037                |
| I-MAPPO-SEM | 86 ± 5.8         | 21.2                    | 1.0301                |

Among all the measurements, the smooth rate  $sr$  is denoted as follows:

$$sr = \frac{\sum_{t=1}^T \sum_{i=2}^{n-1} |\theta_{i+1} - \theta_i|}{nt}, \quad (34)$$

where  $\theta_i = \arctan(\|p_{i+1,y} - p_{i,y}\| / \|p_{i+1,x} - p_{i,x}\|)$ .

Thus, the traditional reward function (TRF) and proposed reward function–cost function (CF) are also tested as the comparison. It is noted that the TRF basically uses the constant value in each predefined condition. In this case, based on Equations (28)–(30), we divide the TRF into three parts: (1) the reward of obstacle collision  $r_{obs}$ ; (2) the reward of neighbors of each agent collision  $r_{ij}$ ; (3) the reward for tracking the reference trajectory  $r_d$ . More precisely, these rewards are designed as  $R = \lambda_1 r_{obs} + \lambda_2 r_{ij} + \lambda_3 r_d$ , shown as follows:

$$\begin{cases} \lambda_1 r_{obs} = -50 \\ s.t. \text{dist}(\hat{X}_i, X_{obs}) = \|p_i - p_o\|^2 \leq 2R_a \\ \lambda_2 r_{ij} = -30 \\ s.t. \text{dist}(\hat{X}_i, \hat{X}_j) = \|p_i - p_j\| \leq R_s \\ \lambda_3 r_d = 200 \\ s.t. \text{dist}(\hat{X}_i, X_d^*) = \|p_i - p_d^*\| \leq E_{min} \end{cases}, \quad (35)$$

where  $\lambda_1, \lambda_2, \lambda_3$  represents the weight of the three kinds of reward values, respectively. Thus, the comparison test result is shown in Table 4. It can be seen that the I-MAPPO-TRF demonstrated that the obtained result cannot reach the high success rate due to the constant reward value, which limits the flexibility to the current changeable state.

**Table 4.** Algorithm comparison in Success rate, Ave-collisions rate and Ave-smooth rate.

| Algorithm   | Success Rate (%) | Ave-Collisions Rate (%) | Ave-Smooth Rate (rad) |
|-------------|------------------|-------------------------|-----------------------|
| I-MAPPO-TRF | 36 ± 5.8         | 51.0                    | 0.5019                |
| I-MAPPO-CF  | 86 ± 5.8         | 21.2                    | 1.0301                |

The total Success rate, Ave-collisions rate and Ave-smooth rate are compared for the different algorithms during learning steps. As shown in Table 5, the success rate of our algorithm was almost 100% within three algorithms, which made great improvement by 47% and 41% compared with the previous work and MADDPG, respectively. Due to the difficulty of convergence of the previous work, the success rate was the most unstable. Secondly, although MADDPG can solve the problems in the continuous action space, it still has disadvantages of the policy selected from the historical experience buffer without completely considering the current state. Our algorithm combines the prediction module on the basis of I-MAPPO, so that the algorithm can achieve a higher success rate in different environments. In addition, based on the combined action selection mechanism, the proposed flight framework would enhance the angular velocity learning ability to guarantee the flight practicability in unknown environments with smoother trajectory. The configuration changes can basically depend on the environment rather than the rigid

information. These results demonstrate that the proposed method significantly improves success rates and collision avoidance rate over the baseline and traditional methods.

**Table 5.** Algorithm comparison in Success rate, Ave-collisions rate and Ave-smooth rate.

| Algorithm | Success Rate (%) | Ave-Collisions Rate (%) | Ave-Smooth Rate (rad) |
|-----------|------------------|-------------------------|-----------------------|
| AMCOFS    | $39 \pm 5.8$     | 53.3                    | 0.2237                |
| MADDPG    | $45 \pm 5.8$     | 42.5                    | 0.6041                |
| I-MAPPO   | $86 \pm 5.8$     | 21.2                    | 1.0301                |

## 6. Conclusions and Future Work

This paper proposed an integrated flight system framework for the trajectory navigation of formation in a continuous differentiable and combined action space. Then, an affine-based DMPC module reference layer was developed to enhance the fast convergence of the proposed algorithm. To demonstrate the effectiveness of the proposed method, we conducted a series of simulations involving a formation with varying numbers of agents and obstacles. The goal was to navigate through square areas characterized by random obstacle motion while adapting configurations and avoiding collisions. The results show that our algorithm notably reduces the collisions' number during online learning, with the average number of collisions being approximately 1/500 of AMCOFS and MADDPG. Moreover, the success rate of the task operated in the learning phase is 41% higher compared to MADDPG, 47% higher compared to AMCOFS. These results indicate that the proposed method effectively facilitates formation navigation and obstacle avoidance, which can ensure the unrigid configuration change continuously.

In future work, we will extend this approach to the three-dimensional environment to enhance practical applicability. Moreover, there are several important interests that require attention, which include variable topology structures and reconstruction response mechanisms to fit unknown environments.

**Author Contributions:** Conceptualization, C.K. and J.X.; methodology and writing—original draft, C.K.; writing—review and editing, Y.B. and W.T.; visualization C.K.; supervision J.X. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The raw data supporting the conclusions of this article will be made available by the authors on request.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## Appendix A

**Table A1.** Notations and symbols.

| Notations and Symbols       | Meaning                            |
|-----------------------------|------------------------------------|
| $i, j = 1, \dots, n$        | Nodes as agents in formation       |
| $G = \{V, A, \varepsilon\}$ | Undirected graph                   |
| $t = 1, \dots, T$           | Time step                          |
| $N, M$                      | Numbers of neighbors and obstacles |
| $p_{ij}$                    | The vector distance between agents |

Table A1. Cont.

| Notations and Symbols                                         | Meaning                                       |
|---------------------------------------------------------------|-----------------------------------------------|
| $R = [r_1, r_2, \dots, r_N]^T$                                | The set of nominal configurations             |
| $\{A(t) \in \mathbb{R}^{d \times d}, b(t) \in \mathbb{R}^d\}$ | Adjustable parameters                         |
| $\ \cdot\ $                                                   | Euclidean norm                                |
| $R_a, R_s$                                                    | Safety distance of the agent and the obstacle |
| $u$                                                           | Input vector                                  |
| $J(\cdot)$                                                    | Cost function                                 |
| $\lambda, \sigma$                                             | Adjusting factor                              |
| $A_\pi(s_n^t, a_n^t)$                                         | Advantage function                            |
| $V_\omega(s_t)$                                               | State value function                          |
| $b(s_n^t)$                                                    | Baseline                                      |
| $\alpha, \beta$                                               | Hyperparameter                                |

## References

- Oh, K.K.; Park, M.C.; Ahn, H.S. A survey of multi-agent formation control. *Automatica* **2015**, *53*, 424–440. [\[CrossRef\]](#)
- Zhao, S.; Zelazo, D. Bearing Rigidity and Almost Global Bearing-Only Formation Stabilization (Vol.). *IEEE Trans. Autom. Control* **2016**, *61*, 1255–1268. [\[CrossRef\]](#)
- Zheng, C.; Mi, Y.; Guo, H. Optimal Spatial-Temporal Triangulation for Bearing-Only Cooperative Motion Estimation. *arXiv* **2023**, arXiv:2310.15846.
- Lin, Z.; Wang, L.; Chen, Z.; Fu, M. Necessary and sufficient graphical conditions for affine formation control. *IEEE Trans. Autom. Control* **2016**, *61*, 2877–2891. [\[CrossRef\]](#)
- Ai, X.L.; Yu, J.Q.; Chen, Y.B.; Chen, F.Z.; Shen, Y.C. Optimal Formation Control with Limited Communication for Multi-Unmanned Aerial Vehicle in an Obstacle-Laden Environment. *Proc. Inst. Mech. Eng. Part G J. Aerosp. Eng.* **2016**, *231*, 979–997. [\[CrossRef\]](#)
- Dong, W. Robust formation control of multiple wheeled mobile robots. *J. Intell. Robot. Syst.* **2011**, *62*, 547–565. [\[CrossRef\]](#)
- Hengster-Movrić, K.; Bogdan, S.; Draganjac, I. Multi-agent formation control based on bell-shaped potential functions. *J. Intell. Robot. Syst.* **2010**, *58*, 165–189. [\[CrossRef\]](#)
- Liu, Y.; Jia, Y. An iterative learning approach to formation control of multi-agent systems. *Syst. Control Lett.* **2012**, *61*, 148–154. [\[CrossRef\]](#)
- Zhao, S. Affine Formation Maneuver Control of Multiagent Systems. *IEEE Trans. Autom. Control* **2018**, *63*, 4140–4155. [\[CrossRef\]](#)
- Mahdoui, N.; Mahdoui, V.; Natalizio, E. Communicating multi-uav system for cooperative slam-based exploration. *J. Intell. Robot. Syst.* **2020**, *98*, 325–343. [\[CrossRef\]](#)
- Zhao, S.; Zelazo, D. Translational and scaling formation maneuver control via a bearing-based approach. *IEEE Trans. Control Netw. Syst.* **2017**, *4*, 429–438. [\[CrossRef\]](#)
- Monteriu, A. Nonlinear decentralized model predictive control for unmanned vehicles moving in formation. *Inf. Technol. Control* **2015**, *44*, 89–97. [\[CrossRef\]](#)
- Kuriki, Y.; Namerikawa, T. Formation control with collision avoidance for a multi-uav system using decentralized MPC and consensus-based control. *SICE J. Control Meas. Syst. Integr.* **2015**, *8*, 285–294. [\[CrossRef\]](#)
- Xue, J.; Kong, X.; Wang, G.; Dong, B.; Guan, H. Path planning algorithm in complex environment based on DDPG and MPC. *J. Intell. Fuzzy Syst.* **2023**, *45*, 1817–1831. [\[CrossRef\]](#)
- Kang, C.; Xu, J.; Bian, Y. Affine Formation Maneuver Control for Multi-Agent Based on Optimal Flight System. *Appl. Sci.* **2024**, *14*, 2292. [\[CrossRef\]](#)
- Chuan, W.; Lei, Y.; Jianguo, Z. Study on optimization of radiological worker allocation problem based on nonlinear programming function-fmincon. In Proceedings of the 2014 IEEE International Conference on Mechatronics and Automation, Tianjin, China, 3–6 August 2014.
- Quan, L.; Yin, L.; Xu, C.; Gao, F. Distributed Swarm Trajectory Optimization for Formation Flight in Dense Environments. In Proceedings of the 2022 International Conference on Robotics and Automation, Philadelphia, PA, USA, 23–27 May 2022.
- Xu, Y.; Lin, Z.; Zhao, S. Distributed Affine Formation Tracking Control of Multiple Fixed-Wing UAVs. In Proceedings of the 2020 39th Chinese Control Conference, Shenyang, China, 27–29 July 2020.
- Bhopale, P.; Kazi, F.; Singh, N. Reinforcement learning based obstacle avoidance for autonomous underwater vehicle. *J. Mar. Sci. Appl.* **2019**, *18*, 228–238. [\[CrossRef\]](#)

20. Chen, C.; Chen, X.; Ma, F.; Wang, J. A knowledge-free path planning approach for smart ships based on reinforcement learning. *Ocean Eng.* **2019**, *189*, 106299. [[CrossRef](#)]
21. Liu, Q.; Shi, L.; Sun, L. Path planning for uav-mounted mobile edge computing with deep reinforcement learning. *IEEE Trans. Veh. Technol.* **2020**, *69*, 5723–5728. [[CrossRef](#)]
22. Sewak, M. Deep Q Network (DQN), Double DQN, and Dueling DQN. In *Deep Reinforcement Learning*, 8th ed.; Springer: Singapore, 2007; pp. 95–108.
23. Zeng, Y.; Xu, X.; Jin, S. Simultaneous navigation and radio mapping for cellular-connected uav with deep reinforcement learning. *arXiv* **2020**, arXiv:2003.07574. [[CrossRef](#)]
24. Peng, L.; Donghui, C.; Yuchen, W.; Lanyong, Z.; Shiquan, Z. Path Planning of Mobile Robot Based on Improved TD3 Algorithm in Dynamic Environment. *Heliyon* **2024**, *10*, e32167.
25. Wang, Y.; He, Z.; Cao, D.; Ma, L.; Li, K.; Jia, L.; Cui, Y. Coverage Path Planning for Kiwifruit Picking Robots Based on Deep Reinforcement Learning. *Comput. Electron. Agric.* **2023**, *205*, 107593. [[CrossRef](#)]
26. Zhang, S.; Pang, Y.; Hu, G. Trajectory-tracking control of robotic system via proximal policy optimization. In Proceedings of the 2019 IEEE International Conference on Cybernetics and Intelligent Systems and IEEE Conference on Robotics, Automation and Mechatronics, Bangkok, Thailand, 18–20 November 2019.
27. Gu, S.; Kuba, J.G.; Chen, Y.; Du, Y.; Yang, L.; Knoll, A.; Yang, Y. Safe Multi-Agent Reinforcement Learning for Multi-Robot Control. *Artif. Intell.* **2023**, *319*, 103905. [[CrossRef](#)]
28. Guo, D.; Tang, L.; Zhang, X. Joint Optimization of Handover Control and Power Allocation Based on Multi-Agent Deep Reinforcement Learning. *IEEE Trans. Veh. Technol.* **2020**, *69*, 13124–13138. [[CrossRef](#)]
29. Yu, C.; Velu, A.; Vinitzky, E.; Wang, Y.; Bayen, A.M. The Surprising Effectiveness of MAPPO in Cooperative, Multi-Agent Games. *arXiv* **2021**, arXiv:2103.01955.
30. Orozco-Rosas, U.; Montiel, O.; Sepúlveda, R. Mobile robot path planning using membrane evolutionary artificial potential field. *Appl. Soft Comput.* **2019**, *77*, 236–251. [[CrossRef](#)]
31. Jiang, S.; Fang, H.; He, K.; Yan, C. Research on obstacle avoidance path planning algorithm for six-axis robot. In Proceedings of the IEEE International Conference on Information and Automation, Fujian, China, 11–13 August 2018.
32. Lewis, S.; Overton, L. Nonsmooth optimization via quasi-Newton methods. *Math. Program* **2013**, *141*, 135163. [[CrossRef](#)]

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.