Distributed Cooperative Control of Multi-Agent System: Path Planning and Formation-Containment Control

Jinyuan Zhang

For multi-agent systems, swarm navigation in a predefined formation with obstacle avoidance has gained significant interest among robotics and control researchers due to its widespread applicability. Considering the limited visibility radius of the agents, this work presents a distributed approach to control a team of self-organized agents moving safely in formation in a 3D environment filled with cluttered obstacles. The formation is determined by the shape vector, which defines the relative distance and bearing. To maintain the desired shape, each agent in our system plans its trajectory independently based on its local information and relative information from other networked agents. According to the proposed method, both the Artificial Potential Field (APF) and leader-follower methods are used to achieve our control objective. In addition, the containment control objective has also been considered. To enhance system robustness, our scheme also incorporates flexible strategies: a) quickly reforming into a new topology if an agent is disconnected, and b) rescaling the formation when a narrow corridor is detected. Finally, a rescue mission simulation with a group of agents is provided to validate the approach's feasibility.

Contents

1 Introduction

1.1 Background and Motivation

Autonomous intelligent systems are complex systems that operate independently without human intervention. With the impressive achievements in areas such as AI, control, mechanics, and communication, autonomous intelligent systems have rapidly developed as a convergence point of these technologies. These systems have garnered attention from both academia and industry. Recent industry highlights include NVIDIA's humanoid robot [1] and Boston Dynamics' legged robots [2], whose applications span a variety of fields, ranging from smart grids [3] to autonomous shepherding systems [4]. Among the various topics on intelligent systems, swarm intelligence stands out as a unique area. Inspired by the fascinating collective behavior observed in nature, particularly from social biological entities such as ants and bird flocks, we attempt to explore a question like this: Is it possible to integrate a large number of individual agent into a cooperative "swarm" that can collectively carry out a prescribed mission and respond as a whole to commands from higher-level planning systems? Therefore, we are interested in developing a framework that offers a potential solution to this challenge. This project specifically focuses on the cooperative control of a multi-agent system (MAS), which representing a practical application of swarm intelligence principles.

Compared to the limitations of single-agent system operations, multi-agent systems (MASs) offer enhanced fault tolerance, increased scalability, and improved adaptability to complex environments, making them better suited to address a wide range of tasks [5]. The effectiveness of MASs fundamentally depends on cooperative control, which serves as the cornerstone of their operational advantage. Hence, the research goal of cooperative control is to design a state-of-the-art control protocol that coordinates agents in a team to pursue a common objective via a communication network. This common goal is exemplified in several remarkable applications, including security surveillance [6] (e.g., fire monitoring), search and rescue operation in dangerous zones [5] [7] (e.g., earthquake-affected areas), cooperative transportation [8], exploration in unknown area [9] (e.g., cluttered bamboo forests), and autonomous vehicle platoons on urban highways [10]. Based on the characteristics of these tasks, agents can be categorized into three heterogeneous types: Unmanned Aerial Vehicles (UAVs), Unmanned Ground Vehicles (UGVs), and Unmanned Surface Vehicles (USVs).

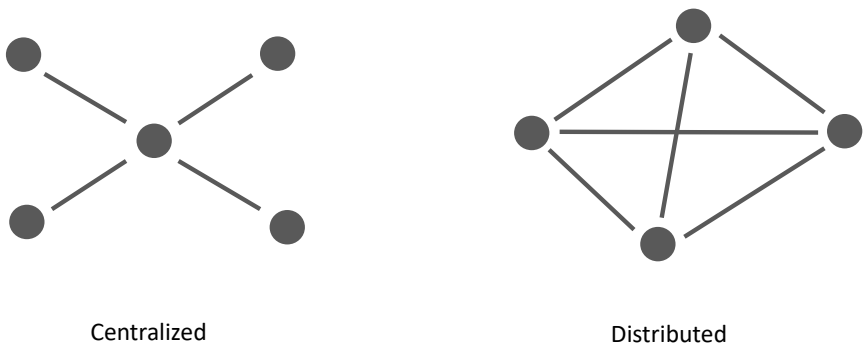


Figure 1: Network Topologies in MASs Based on Centralized and Distributed Implementation.

So far, among the emerging the techniques in cooperative control, extensive effort has been dedicated

to exploring various coordination strategies. Generally, these strategies can be classed into two implementations: (a) *centralized* and (b) *distributed*. The centralized approach relies on a single controller to manage all decision-making processes, meaning that this entity makes all the decisions, which are then executed by subordinate agents. In [11], a centralized sequential convex optimization method has been developed to calculate optimal parameters for formation. This method requires a high-bandwidth communication system and has poor fault tolerance. Consequently, if the central node fails, the entire system becomes inoperative. However, considering the facts discussed previously, the concept of distributed system has seen wider use. In such systems, decision-making depends on the information flow between agents and their immediate neighbors through a connected graph. Each agent can independently process information, make decisions, and take appropriate actions. This implementation significantly enhances the system's robustness to failures. In [12], a decentralized spatial-temporal trajectory optimization framework was designed for an aerial robot team. This framework considers both formation similarity and dynamic feasibility to simultaneously achieve formation flying and inter-agent collision avoidance. However, a weakness in this approach is that each agent is required to perform very heavy local optimization calculations, potentially leading to a high onboard computational time cost.

Further comparisons are detailed in [13] , which demonstrate the pros and cons of both centralized and distributed control solutions through practical implementations. Some other implementation, such as the decentralized method discussed in [5], is a specific subset of distributed systems.

1.2 Formation Control in Free Space

Among the emerging research branches in cooperative control, formation control and containment control stand out as two representative techniques. Formation control aims to generate appropriate control signals that drive multiple agents to achieve a predefined geometric pattern by controlling the relative distance and bearing between agents. Beyond the optimization-based method discussed above, extensive strategies related to formation, such as leader-follower [14], behavior-based [15], and virtual structure [16] approaches, have been developed. Among the methods mentioned, the behavior-based method is not widely used since it decomposes the agent's motion into several specific behaviors, such as obstacle avoidance and following behavior. The convergence process of the system cannot be expressed with mathematical formulas, and as a result, the stability of the formation cannot be guaranteed. On the other hand, the virtual structure method performs well in maintaining formation stability because it treats a Multi-Agent System (MAS) as a single, rigid entity. However, it is unsuitable for systems with a large number of agents [17].
Thanks to the simplicity, stability, flexibility and of the leader-follower approach, it is the most widely used in practical applications. The leader-follower approach operates on the principle that one or more designated "leaders" guide the path, while the "followers" obey to specific rules to maintain their positions and orientation relative to the leader. This leader has full access to the overall navigation information and serves as the reference point in the formation. In some scenarios, such as [14], [18], a virtual leader is designated to replace the actual leader in the formation. Thus, the leader role can be either virtual or actual, tailored to the specific needs of the application.

In this setting, the computational demand on the followers is significantly reduced. The widespread adoption of the leader-follower approach is primarily due to the success of consensus theory [19] in 2005 regarding formation control, as the follower adjusts its independent controller to seek a consensus state (e.g., position, velocity) with its leader and other neighbors. In [20], Zhao proposed a distributed leader-follower based control scheme that allows MAS to perform time-varying affine transformations of their formation, such as rotation, shearing, and scaling. However, this approach (leader-follower-based) also has its weaknesses; the failure of the leader can destroy the entire team. To address this problem, a leader re-selection mechanism can be designed.

1.3 Containment Control in Free Space

Beyond formation control, containment control represents another active area in cooperative technique [21]. Complex tasks often need the collaboration of multiple leaders to ensure effective completion. In such multi-leader systems, containment control plays a crucial role. For instance, in a multiple leader system, containment control ensures that followers are "driven" to converge within the convex hull (a closed geometric space) formed by the leaders, as described in the study [5],and [21]. It's worth mentioning that the convex hull formed does not need to be a regular shape. Thus, the containment problem can be viewed as a special case of the traditional leader-follower scenario. It's indeed useful in some practical scenarios, such as rescue missions, where containment control enables lost agents to move into a convex hull formed by rescuer agents, who then guide them back to safety.

The containment control problem has been studied for groups of first-order [21], second-order [22], and higher-order integrator agents [23]. The issue also extends to heterogeneous agent systems; as presented in [24], a reinforcement learning based model-free control algorithm is proposed, which can tackle scenarios with both static and stationary leaders. Additionally, another study [18] explores containment control within a network of quadrotors.

1.4 Formation-Containment Control Problem

Based on formation control and containment control, a more fascinating topic, formation-containment control, has arisen, which was first investigated in [25] in 2015 and further extended in [26]. This technique is the most recent development that combines the objectives of both the formation and containment problems, namely, formation-containment control. Specifically, this approach requires not only that several leaders form a formation, but also that followers converge within that formation at the same time. Hence, two-layer tasks execute simultaneously.

This technique offers advantages in certain applications. For example, consider a team of robots exploring a narrow, unknown valley, only a few robots are equipped with sensors to detect narrow spaces. They become the "leaders" of the group, while the rest serve as "followers". When leaders encounter a narrow valley, they create a safe region, a convex hull. Followers can safely reach their destination by entering and staying within this safe region. A novel study [5] introduced a two-layer decentralized formation-containment framework designed for large-scale multi-robot systems, aimed at achieving the desired goal. Specifically, this framework addresses issues such as leader clustering, formation tracking, and the containment problem. However, other robust strategies require further research, including formation scaling, navigation in very dense, dynamic obstacle environments (such as areas with non-convex obstacles and narrow corridors).

1.5 Path/Motion Planning in Dense Environment

Now, we have discussed two cooperative techniques: formation and containment control. Most research on cooperative control assumes obstacle-free environments. However, due to real-world application requirements, there is a lack of studies on robust MAS navigation and obstacle avoidance in cluttered environments, since agents need to keep safe. In such situations, apart from the cooperative control, MAS motion/path planning also plays an important role. These techniques need to be employed to solve and facilitate the collision problem. Some pioneering research in obstacle avoidance has been conducted. These include artificial potential field (APF) [14], Model Predictive Control (MPC) [27], and strategies based on reinforcement learning (RL) [28], which all aim at providing a collision-avoidance component to agents' controllers for planning safe, collision-free paths. Specifically, APF creates virtual repulsive potential fields around each agent and obstacle, which enables agents to compute a path that always moves towards areas where the potential field's gradient decreases [29]. []

The MPC method employs a more dynamic path planning approach by introducing a collision avoidance term in the agent's optimization function. It utilizes a rolling optimization strategy for real-time path adjustments to handle obstacles [27]. RL-based strategies guide agents to learn through trial and error during training by designing navigation and collision avoidance rewards in their reward function, achieving self-optimized path planning in cluttered environments [28]. Due to the complex calculations, substantial onboard computing resources, and long training time required by MPC and RL methods, the APF method is more commonly employed for its simpler controller, fewer required parameters, and the ability to be mathematically modeled. Literature [30] proposed an obstacle avoidance control algorithm based on an improved APF path planning algorithm suitable for UAV swarm systems, which also addresses the issue of local minima (a common issue for APF). More recently, other optimization-based methods are wildly used to solve reciprocal avoidance in [9] for a real-time, large-scale formation flight. However, there is still limited research that solves the problem of cooperative control (e.g., formation-containment problem) and path/motion planning problem simultaneously.

1.6 Summary on Literature Survey

Formation Control			
Leader-follower [14] [20]	Virtual structure [16]	Behavior-based [15]	Reinforcement learning [28]
Optimization based [9] [12] [13] [8]	Time-varying formation [31] [32]	Fixed-time [33] [34]	Switching topology [31]
Containment Control			
First-order [21]	Second-order [22]	Higher-order [23]	Heterogeneous [24]
Formation-Containment Control			
First/Second-order [5] [18] [26]	High-order [25]	Heterogeneous [24]	Fixed-time [35]
Path Planning			
Dijkstra [36]	A* [37]	RTT [38]	Genetic algorithms [39]
Reinforcement learning [28]	MPC (other optimization methods) [9] [12] [27]		Potential Field (APF) [29] [5] [30] [14] [40]

Table 1: Summary (some content may not be mentioned in the text).

1.7 Contribution and Content

Motivated by the previously mentioned challenges and the growing demand for the design of advanced cooperative control techniques that deliver greater flexibility, autonomy, and fault robustness in complex tasks, this project aims to propose a distributed cooperative control framework for a MAS. This framework addresses formation control, containment control, and path planning (e.g., achieving a safe path) objectives. It focuses on a practical scenario to demonstrate swarm navigation with full autonomy in a wild, dense environment. Through cooperation between ground-air agents and intra-group network communication, a rescue mission is performed to save some lost robots. In this setup, all agents operate without prior knowledge of the environment, with only the leader agent (rescuer) equipped with obstacle detection sensors, while the follower (lost robot) is not. To mitigate unexpected situations, such as agent malfunctioning or the detection of narrow corridors, we have embedded redundant/fault-tolerant mechanisms to enrich our control strategy.

We detailed our contents and contribution as follows:

1. We propose a distributed cooperative control scheme for multiple networked agents to ensure that they achieve both formation and containment objectives during task execution. Both the leader and the follower controller will be presented in this project.

2. To address the need for safe navigation, we designed a path planning algorithm based on APF that handles both static and dynamic obstacles, ensuring that the swarm computes a safe trajectory without any internal collisions.

3. We present several mechanisms that enhance system resilience and fault tolerance, such as the reconfigurability of formation, which enables agents to rapidly reform into a new formation in the event of an agent malfunction. Another mechanism, such as the ability to rescale, allows agents to adjust the size of their formation to safely navigate through narrow valleys.

4. All three modules are integrated into a hierarchical cooperative control framework to create a high-level planning system. For each module, a series of simulation experiments have been carried out to validate the practicality and implementation capability of the respective technologies.

5. Basing on and extending the work [5], we expand the rescue simulation into three-dimensional space with a denser obstacle environment, incorporating agents such as UAVs and UGVs. This simulation validates the effectiveness and feasibility of the proposed hierarchical framework.

1.8 Symbol and Notation

Symbol	Notation
G	Interaction graph of agents
a_{ij}	Adjacency weight from node j to node i
p_i	Position of agent i
r_i	Radius of agent i
u	Control input for an agent
N	Total number of agents in the system
M	Total number of leaders in the system
$\mathrm{h_i}$	Formation shape vector for agent i
δ_{ij}	Formation offset between agents i and j
α_{ik}	Non-negative constants satisfying $\sum_{k=M+1}^{N} \alpha_{ik} = 1$
s	Formation scaling Vector
k_f	Formation control gain
k_c	Containment control gain
γ	Formation control coefficient
μ	Containment control coefficient
$\mathcal{T}$	Virtual target
$f(t)$	Predefined path
U	Virtual potential field
d_{safe}	Predefined safe distance for agent
$d_{\max}$	Threshold distance for repulsion activation
F	Virtual force
K_{rep}	Repulsive potential field constant
$\mathrm{sgn}(x)$	Signum function defined as $\frac{\lvert x\rvert}{x}$, if $x \neq 0; 0$, if $x = 0$

Table 2: Symbols and their notations

2 Technical Background and Problem Formulations

Establishing a communication network among agents is essential for cooperative tasks. In this section, we introduce the basic concepts of graph theory, which serve as a mathematical representation of the information flow topology between agents. Following this, a detailed technical problem description is presented.

2.1 Communication Network

In this project, the communication topology between agents (referred to as nodes in the following) is described by a directed graph. By representing agents as nodes and their communication links as edges, graph theory allows for the mathematical analysis of the system's structure and dynamics [[26]]. Let $G = (V, \varepsilon, A)$ be a directed graph containing N nodes, where $V = \{v_1, v_2, v_3 \ldots, v_N\}$ is a set of nodes, each one corresponding to an agent in MAS. ε denote as a set of edges and it satisfy $\varepsilon \subseteq V \times V$, such that $\varepsilon_{ij} = (v_j, v_i) \in \varepsilon$ means the edge between the pair of nodes i and node j. It illustrates the information flowing from node j to node i. Note that in graph G, if there exists a directed path from any node to another distinct node, then the graph is considered to be connected. A is the $N \times N$ adjacency matrix with element $a_{ij}, A = [a_{ij}]$. If there is a direct communication (edge) from node j to node i, then the value of a_{ij} is 1 ; if there is no direct communication relationship between them, then, otherwise a_{ij} is 0 . In addition, since each node does not form an independent communication link with itself, then elements

a_{ii} on the diagonal of the A are all 0 . Hence $a_{ij} > 0 \Leftrightarrow \varepsilon_{ij} \in \varepsilon$, otherwise $a_{ij} = 0$ and $a_{ii} = 0$. Also, we should mention that $a_{ij} \neq a_{ji}$ in direct graph.

In our MASs scenario, adjacency matrix A is important as it specifies the exact communication relationship among agents, and it revels how agents interact with each other. Define an in-degree matrix for a graph G as $D = \text{diag}\{d_i\}$ where D is an $N \times N$ diagonal matrix with $d_{ii} = \sum_j^N a_{ij}$, which signifies that the in-degree of each node is obtained by summing the elements of the corresponding column in the adjacency matrix A.

The Laplacian matrix $L \in \mathbb{R}^{N\times N}$ of G is defined as $L = D - A$. By analyzing it, we can understand the network's connectivity and properties, such as the potential for consensus among agents. A directed graph has a directed spanning tree if its Laplacian matrix has one zero eigenvalue and all other eigenvalues with positive real parts, which enables a single root node (leader) to influence the entire network, which is a condition for controllability and consensus.

In our case, to ensure all follower agents track the leader, the communication topology must guarantee at least one directed path from the leader to every follower.This allows each follower to access the leader's state information directly or indirectly.

2.2 Agent Motions

Considering a single integrator MAS consisting of N agents in two-dimension space, in which the position of the i^{th} agent is defined as $x_i(t)$, and the dynamic of each agent is characterized as follows:

$$\dot{p}_i(t) = u_i(t) \quad \forall i = 1, 2, \ldots, N \tag{1}$$

Where $u_i(t) \in \mathbb{R}^2$ represents the control input, such as velocity, applied to the i^{th} agent. It is a state vector that is decomposed into its components along the x and y axes in 2D space, given by $u_i(t) = [u_{ix}(t), u_{iy}(t)]^T$. Similarly, the position state vector $p_i(t)$, is represented in $\mathbb{R}^2$ as $p_i(t) = [x_i(t), y_i(t)]^T$. The control inputs u_i are bounded, $|u_i(t)| \leq u_{\text{max}}$, meaning there exists a predefined limit to the velocity that each agent can attain. This constraint ensures that the agents operate within safe and efficient operational parameters.

We model each agent as a point mass as shown in equation (1), but in practical implementation, the robot agent's kinematics are highly coupled and nonlinear. To address this, the input-output feedback linearization technique [41] is applied to transform the nonlinear and high coupled MAS into a controllable linear form at each operating point.

2.3 Problem Statements

This section shows a precise and technical problem statement. The primary goal of this project is to formulate a hierarchical control framework suitable for real-world scenarios where no agent knows prior information about the environment, and a few agents are designated as leaders, capable of guiding the followers along the path to execute the relevant task. There are three objectives that need to be achieved within this framework: a) Formation objective, b) Containment objective, and c) Path/motion planning objective. Assumption 1 outlines the MAS configuration for this project.

Assumption 1. *For a multi-agent system comprising N agent, the interaction topology is abstracted as a directed graph G. Assuming M followers in this MAS, denoted by $F = \{1, 2, 3, \ldots, M\}$, the remaining $N - M$ leaders are denoted by $L = \{M + 1, M + 2, M + 3, \ldots, N\}$.*

1) Formation Control Objective:

The Formation control problem is multiple agents independently coordinating their positions to collectively exhibit a macroscopic pattern or configuration. To achieve the formation control objective using a

leader-follower approach, a reference point or leader is established at position p_k, with each agent maintaining a predetermined distance from this leader to form the desired formation. Consider the i^{th} and j^{th} neighboring agents, where $i \in F, j \in \mathcal{N}_i$, with $\mathcal{N}_i$ representing the neighbor set of i^{th} agent. Their positions, P_i and P_j belong to $\mathbb{R}^2$. The formation geometry vectors for these two agents with respect to the leader/ are defined as $\mathbf{h_i}$ and $\mathbf{h_j}$. $\mathbf{h_i} \in \mathbb{R}^2$, as $\mathbf{h_i} = \left[\mathbf{h}_{\text{ix}}, \mathbf{h}_{\text{iy}}\right]^T$. These values are preset by the user and represent the desired Euclidean distances to the leader/target. Then, the formation offset between agent i and j is given by $\delta_{ij} = \mathbf{h_i} - \mathbf{h_j}$, with property $\delta_{ij} = -\delta_{ji}$.

A formation objective in a MAS is considered achieved if the following conditions met:

$$\lim_{t\to\infty} (p_i(t) - p_k(t) - \mathbf{h_i}) = 0 \tag{2}$$

$$\lim_{t\to\infty} (p_i(t) - p_j(t) - \delta_{ij}) = 0 \tag{3}$$

Where p_k represents the position of the target or leader, which the followers use as a reference point to maintain a specified distance and establish a formation. j is an adjacent/neighboring agent of i , as defined by $i \in F, j \in \mathcal{N}_i$.

2) Containment Control Objective

The containment control requires the followers to converge into the convex hull created by the leader. We note that the creation process of the convex hull can be seen as the process of the leader forming a formation via the formation control mentioned in the first control objective. Once the formation is attained by the leader, the followers, according to the containment objective, converge into the convex hull formed by the leader. It should be noted that at least two leaders are required to form a convex hull. Containment is achieved by the followers if there exist non-negative constants α_{ik} satisfying $\sum_{k=M+1}^{N} \alpha_{ik} = 1$, such that for follower i, the following convergence condition is met.

$$\lim_{t\to\infty} \left(p_i(t) - \sum_{k=M+1}^{N} \alpha_{ik} p_k(t) \right) = 0 \tag{4}$$

Where $i \in F$ (follower set) and $k \in L$ (leader set). Objective (2) implies that as time progresses, follower i 's position will converge to a specific position, which is the weighted average position of all i 's networked leaders due to the presence of α_{ik} term. Thus, when this discrepancy reduces to zero, the containment objective is fulfilled, meaning that the follower has contained itself within the convex hull formed by the leaders.

3) Path Planning Objective

To ensure the safety of agents, the path planning objective needs to be considered in our framework. Let agent i have a physical radius r_i and a sensor/visible range d_{sensor} . Upon detecting an obstacle with centroid p_o and encompassing radius r_o, the agent needs to maintain a safety distance d_{safe} to provide a buffer zone to prevent potential collisions. Similarly, the collision avoidance between agents must also be ensured. Therefore, agents are said to achieve path planning objective if the following expressions always hold for all time.

$$\|p_i(t) - p_o\| \geq r_i + r_o + d_{\text{safe}} , \tag{5}$$

$$\|p_i(t) - p_j(t)\| \geq r_i + r_j + 2d_{\text{safe}} , \tag{6}$$

3 Main Result

This section presents the detailed development of a cooperative control framework for multiple agent applications. Following the same description sequence as in Section 2.3 (Problem Statements), four objectives-oriented different controllers will be fully introduced, along with an explanation of why they were designed. We will then proceed with three subsections to formally describe the content and technology in our framework of these schemes, as shown below:

- *Part 1:* Distributed formation controller and some redundant/fault-tolerant algorithms are introduced that allow formation tracking, formation scaling and formation reconfiguration.
- *Part 2 and 3*: Distributed containment controller is designed while combined with formation controller to form a whole formation-containment protocol that can steer the leader to attain a predefined formation first and then guide followers into the formation shape.
- *Part 4*: APF-based path planning algorithm is designed to be embedded in each individual agent as a high-level planning system to prevent collisions.

3.1 Distributed Formation Controller Design for MASs

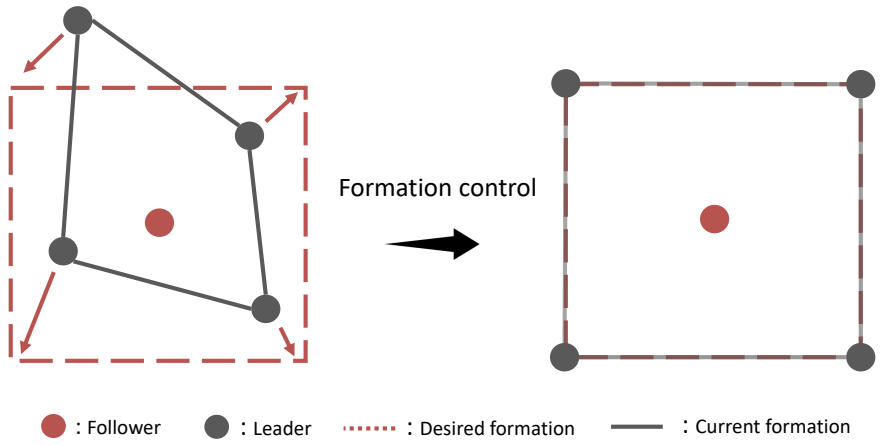


Figure 2: Graphical representation of the formation process for four agents (initially placed at random positions, then forming a square formation shape)

In this section, we develop a formation control law based on the leader-follower approach to achieve the desired pattern for a MAS. For a given desired shape, the objective is to enable each agent to converge to a position relative to the leader by regulating the control inputs (i.e., the speed of the follower). Figure 2 shows the follower forming a square formation from a random initial position. Note that we would like to achieve such objective in a distributed manner, where each agent has its own controller.This controller only relies on the follower's local information. This local information includes two scenarios: one is the leader's position information directly obtained from the leader, and the other is only from the position information of neighboring agents with whom they directly communicate. To facilitate this approach, the following assumption needs to hold true:

Assumption 2. *The interaction topology must include a directed spanning tree, with at least one leader serving as the root node. Additionally, it is assumed to be connected, to ensure that all followers can receive the leader's state information directly or indirectly.*

Through this communication method, we only require some agents to receive information from the leader, whilst other agents are only dependent on their neighbor's positional information so that all agents can successfully converge to the leader's relative position $\boldsymbol{h_i}$, thereby achieving the formation objective.

Motivated by literatures [5] [14], [21], [26], and following the formation objective (2), the formation tracking error for a follower agent i is mathematically defined as:

$$\xi_i = \sum_{j=1}^{M} a_{ij} \left((p_i - \mathbf{h_i}) - \left(p_j - \mathbf{h_j}\right) \right) + \sum_{k=M+1}^{N} a_{ik} \left((p_i - \mathbf{h_i}) - p_k \right) \forall i \in F \tag{7}$$

Here, a_{ij} and a_{ik} are the elements from corresponding adjacent matrices which indicate the communication links. $a_{ij} = 1$ if follower j communicates with follower i , and $a_{ik} = 1$ if leader k communicates with follower i , otherwise these elements are zero.

For example, consider a team with $N = 4$ agents, where the follower set is $F = \{1, 2, 3\}$, and the leader set is $L = \{4\}$. The formation configuration matrix is preset as $\mathbf{h}_{\text{formation}} = [\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3]^T$. In this specific scenario, the tracking error for follower agent 1 is specially:

$$\xi_1 = \sum_{j=1}^{3} a_{1j} \left((p_1 - \mathbf{h_1}) - \left(p_j - \mathbf{h_j}\right) \right) + a_{14} \left((p_1 - \mathbf{h_1}) - p_4 \right) \quad \forall i \in F \tag{8}$$

This representation captures the differences between the current positions of the agent 1 and their desired positions relative to both other followers and the leader. The weights are determined by the adjacency matrix elements, which reflect the communication topology. Noting that element like a_{11} is zero in corresponding adjacent matrix.

***Controller 1*:** The distributed formation controller for each follower agent is defined by the equation:

$$u_{i,\text{ formation}} = -k_f \xi_i - \gamma \operatorname{sgn}(\xi_i) \quad \forall i \in F \tag{9}$$

Where $k_f > 0$ is a scaling gain factor that prompts the agent to move counter to the formation tracking error ξ_i, with the objective to rapid minimize this error. The signum function $\operatorname{sgn}(\xi_i)$ serves as a corrective term to balance dynamic variations due to leader/target movement and to compensate for potential inaccuracies such as sensors. With the constant $\gamma > 0$, this also facilitates rapid error reduction to guide the follower in achieving the desired system state. Constant k_f and γ required proper design.

It's worth to remark that in the formation control problem discussed, the "leader" can be either a physical entity or a virtual construct. If it's virtual, we refer to it as the "target". This implies that formation control does not necessarily rely on an actual leader entity; instead, a virtual target position or state can be established as a reference, guiding follower agents to adjust their positions and achieve the predefined formation pattern.

To enrich the formation control strategy, two resilient / fault-tolerant formation related algorithms are proposed: a) *formation scaling* b) *formation reconfiguration*.

a) Formation Scaling Algorithm

Inspired by [18], to scale the formation, particularly necessary when agents encounter restrictive environments like narrow corridors, a scaling factor vector $s = [s_x, s_y]^T$ is defined. s_x and s_y represent the scaling variables along the X and Y axes, respectively. This scaling factor is important to modulate the

formation size and enable the group to uniformly scale down and successfully navigate through narrow spaces. When a scaling command is required or we say a narrow corridor detected , the leader computes an appropriate scaling factor for the followers. Inspired by[18], the scaled formation offset vectors between the leader and each follower are express as:

$$\hat{\mathbf{h}}_i = \left[s_x \mathbf{h_{ix}}, s_y \mathbf{h_{iy}}\right]^T \tag{10}$$

Similarly, the scaled formation offset between two followers such as i and j can be derived from its definition, shown below,

$$\widehat{\boldsymbol{\delta}}_{ij} = \left[s_x \delta_{\mathrm{ij,x}}, s_y \boldsymbol{\delta}_{\mathrm{ij,y}}\right]^T \tag{11}$$

b) Formation Reconfiguration Algorithm

Since agent faults are a common occurrence in multi-agent systems, incorporating redundancy strategies is essential for robust operation.

Let $F = \{1, 2, 3, \ldots, M\}$ denote the set of all follower agents in a multi-agent system. Suppose there exists a subset of faulty agents, denoted as F_{fault} , containing n agents. The objective of our algorithm is to reconfigure the remaining $M - n$ operational agents into a new predefined formation. Then, we introduce a set H_{r} to denote a predefined redundancy set, which contains M different user-defined formation configuration where $H_{\mathrm{r}} = [h_{\text{sub 1}}, h_{\text{sub 2}}, h_{\text{sub 3}} \ldots h_{\text{subM}}]$, In the event of a fault, the remaining $M - n$ agents will be mapped to the $(M - n)^{th}$ index within the H_{r} set, to regenerate corresponding positions according to that configuration.

For example, if $F = \{1, 2, 3, 4, 5\}$ and F_{fault} is 2 and 4 , resulting $M - N = 3$, then the operation agent is $1, 3, 5$. In this case, h_{sub3} will be mapped these three agents 1,3 and 5 which will be aligned to the $h_{sub3} = [h_1, h_2, h_3]$ that exactly contains 3 positions.

Simulations on formation scaling and formation reconfiguration algorithms are presented in the Appendices.

We will now present a control algorithm designed for follower formation tracking, scaling, and reconfiguration within multi-agent systems.The Algorithm 1 is shown below.

Algorithm 1 Control algorithm for follower formation tracking, scaling, and reconfiguration

Initialization: Define formation configuration $\mathrm{h_f} = [\mathrm{h}_1,\ \mathrm{h}_2, \ldots, h_M]$, interaction topology G, follower and leader set, etc.
for each agent $i \in \{1, \ldots, N\}$ **do**
    if the agent i is a follower **then**
        Receive the leader's position p_k directly or via neighbor's information.
        Compute the tracking error ξ_i using (7).
        if formation scaling/reconfiguration is required **then**
            Compute new h_i and δ_{ij} using the scaling factor s or $h_{\mathrm{sub}(M-N)}$ provided by the leader.
            Update tracking error ξ_i with the scaled positions.
        end if
        Compute the control input u_i using the distributed formation control law (9).
    end if
    Apply the control input u_i to move the agent towards their desired position.
end for
The leader monitors the system formation and fault, adjusts scaling factors s and n when necessary.
Repeat steps 2-14 until the formation is maintained as per the desired configuration.

Simulation 1:*Formation control via leader-follower strategy.*

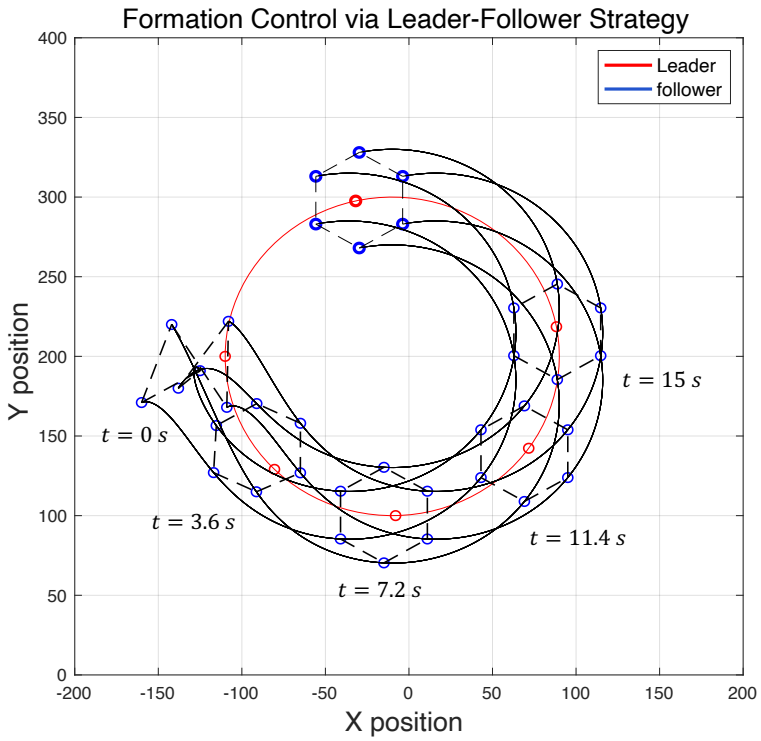


Figure 3: Trajectories of the leaders and followers, with snapshots taken at 0s, 3.6s, 7.2s, 11.4s, and 15s.

In the simulation results shown in Figure 2, we applied the control strategy defined in equation (10), with a focus on formation and tracking behaviours, but without considering inter-agent collision avoidance. Figure 2 shows the trajectories of the agents at selected snapshots taken at 0s, 3.6s, 7.2s, 11.4s, and 15s, providing a comprehensive view of the formation process and subsequent tracking of the leader. Initially, at t=0s, the six agents occupy random positions. During the interval from t=0s to t=3.6s, the agents are observed to transition from these initial states towards a predefined geometric formation (a hexagon). At 3.6s, the formation starts its tracking phase where the collective aligns its movement with that of the leader, progressively reducing individual positional errors until consensus is achieved. The adherence to the prescribed formation is increasingly improved over time, as reflected by the diminishing deviations from target positions within the formation. The "conformity" to the prescribed formation progressively enhances over time, as indicated by the decreasing deviations from target positions within the formation.

3.2 Distributed Containment Controller Design for MASs

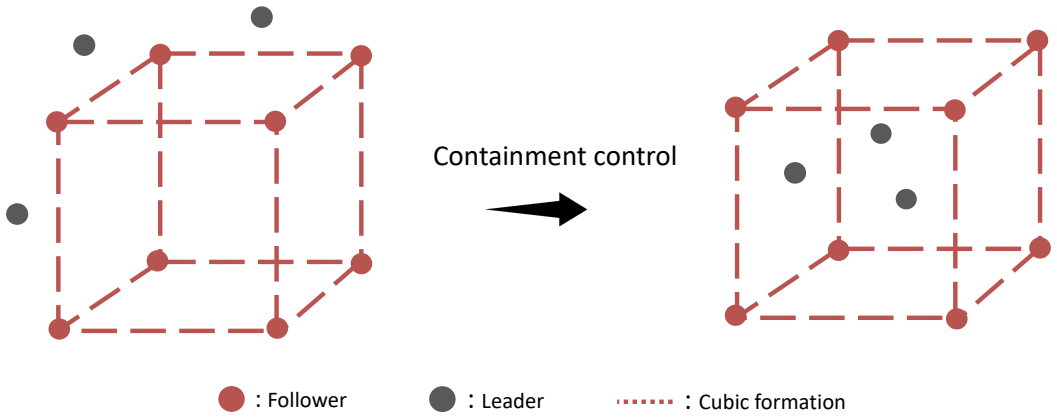


Figure 4: Graphical illustration of containment control (Initially, followers are positioned outside of the cubic formation. They then converge within the cubic convex hull spanned by leader

In continuation of our discussion on the formation control problem, under the motivation of [5] [21] [26], we will now introduce containment control as the second component of our hierarchical framework. To fulfill the containment objective in a leader-follower system, two assumptions must be met:

Assumption 3. *At least two leaders are needed to predefined a convex shape (e.g., circle or triangle), which must be established prior to the implementation of containment control.*

Assumption 4. *For each follower, there must be at least one leader with which it can directly communicate.*

The term "containment" is reflected in the core objective of the control strategy: through appropriate control inputs, it ensures that follower agents not only follow the leader agents but also remain within the convex hull formed by the leaders, namely, containment control. It's worth mentioning that this convex hull can have an arbitrary irregular shape, there is no need for it to have a specific shape. This approach is highly applicable in real-world scenarios such as rescue missions, where followers may lack obstacle

avoidance capabilities and are unable to return to a safe position on their own. Containment control offers an effective solution to this problem by simply requiring followers to maintain their position inside the shape defined by the leaders, such as within a circle. The control objective for a follower i, $i \in F$, is said to achieve the containment objective if the following condition holds true:

$$\lim_{t\to\infty}\left(p_i(t) - \sum_{k\in L}\alpha_{ik}p_k(t)\right) = 0. \quad \alpha_{ik} \geq 0 \tag{12}$$

This control objective was discussed in the previous sections. In the containment control problem, the condition $\sum_{k\in L}\alpha_{ik} = 1$ must be satisfied, meaning that the sum of the influence weights from each leader on a follower must equal 1. Through this normalization, we ensure that followers can eventually reach a point within the convex hull.For the follower agent, we define the containment error ζ_i for all $i \in F$, with respect to its neighbors as

$$\zeta_i = \sum_{k\in\mathcal{N}_i} a_{ik}\left(p_i(t) - p_k(t)\right) \tag{13}$$

Where $\mathcal{N}_i$ denotes the set of neighbors for followers i, which includes both leaders and followers with whom communication is established. a_{ik} signifies the adjacency matrix element that reflects whether there is communication between agent i and k, Since we have already defined $k \in \mathcal{N}_i$, there is no case $a_{ik} = 0$ here.

Controller 2: For a first-order MAS (1) with both dynamic and static leaders, we have designed a distributed containment controller for followers as expressed below,

$$u_{i,\text{ containmet}} = -k_c\zeta_i - \mu\,\text{sgn}\left(\zeta_i\right) \quad \forall i \in F \tag{14}$$

Where k_c and μ are positive scaling factor, that require proper design. ζ_i is the containment error defined earlier in equation (13). The design of containment controller follows same principles as controller 2 which was previously discussed, with the only the changes being the k_c and μ, as well as containment error ζ_i.

Simulation 2: *Containment Control (Static leader form irregular formation)*

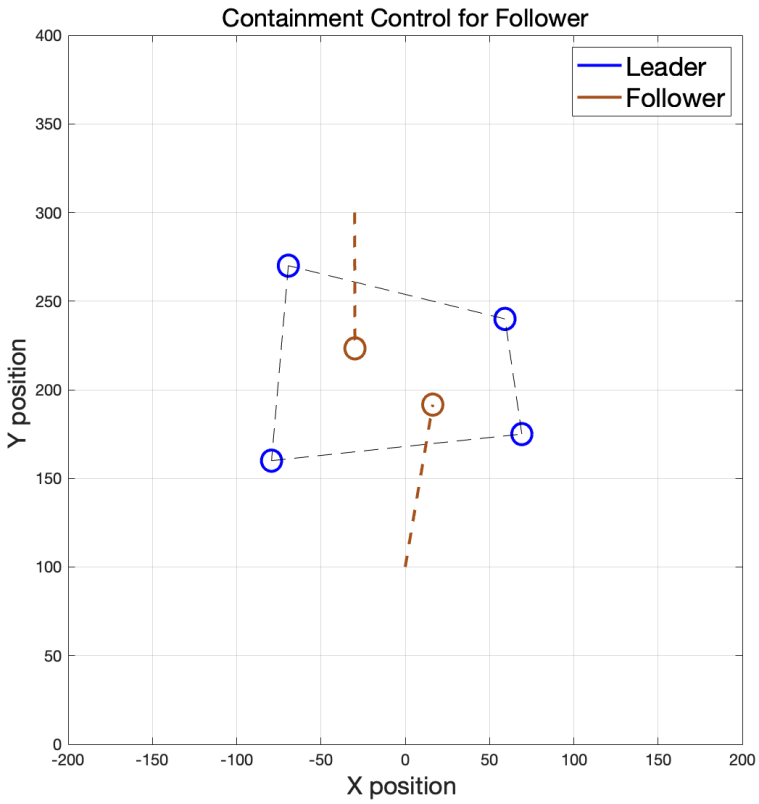


Figure 5: The trajectory of followers converges within the irregular convex hull formed by static leaders.

In this simulation, we applied the control strategy outlined in Equation (14), which focuses on whether the followers can converge into the formation shape formed by the leader. Figure 5 presents the followers' trajectories, and as observed, all followers eventually converge into a static irregular convex hull formation defined by the leader. Their final positions settle at the centroid of the positions of their three nearest leaders. This outcome is due to our configuration that these two followers communicate only with their adjacent three leaders. We assume that there is no communication between these two follower agents. These results are consistent with our analysis of the containment control protocol described earlier.

3.3 Distributed Formation-Containment Control Protocol for MASs

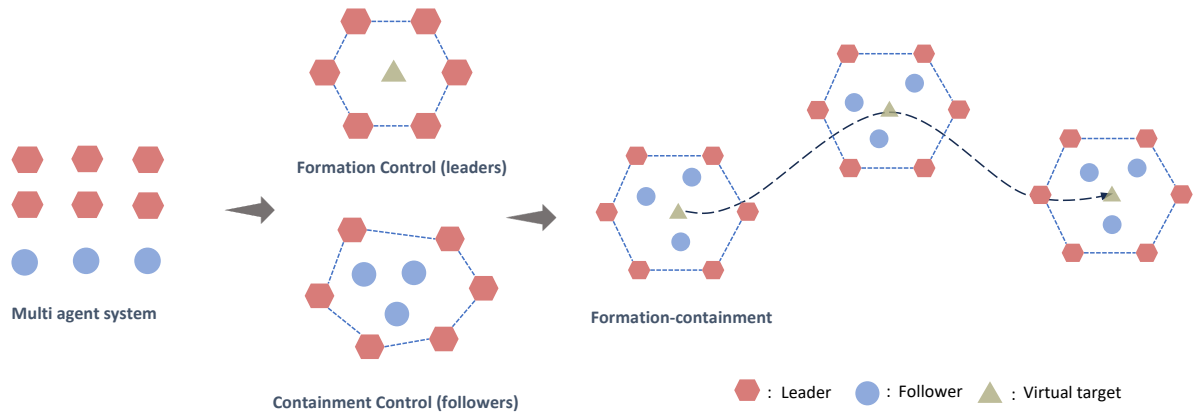


Figure 6: Graphical illustration of formation-containment framework for a MAS

We begin by clarifying formation-containment control, a technique that combines properties of both *formation* and *containment* control. In the context of formation-containment scenarios, three main components are identified: Virtual Target, Leader and Follower. The following assumptions must be met prior to formation-containment control.

Assumption 5. *A virtual target (also known as virtual leader) is introduced to serve as a reference trajectory for the leader agents to follow.*

The complete process of formation-containment control is described as follows, and these two layer tasks occur simultaneously:

1. Via the formation controller, the leader agent aims for a virtual target, forming a specific shape and continuously tracking this virtual target.
2. Via the containment controller, follower agents move into and remain within the convex hull maintained by the leader. When the leader begins to move, the followers continuously track and ensure they remain within the convex hull.

It's essential to recognize that, within the formation-containment control, the objectives of formation control and containment control are specifically distinct, as described below:

Objective of Formation Control in the Formation-Containment Problem: This technique is specifically oriented towards the leader. This differs slightly from the definition provided in Section 3.3, where the terms "leader" and "virtual target" mentioned here correspond directly to the "follower" and "leader" roles as previously described in Section 3.3, illustrating a terminological shift.

Objective of Containment Control in the Formation-Containment Problem: This technique is specifically oriented towards the follower agents. Unlike the general containment control described in 3.2, which allows for arbitrary, irregular shapes for containment, the containment control discussed here specifically demands that follower agents move into a convex hull that predefined by leader agents as a regular geometric shape (e.g., hexagon, triangular).

Therefore, in the realm formation-containment problem : For leader agents, their error is defined as "formation tracking error" in relation to the virtual target. For follower agents, the error is defined as "containment error" with respect to the leader agents.

Given more technical form: The sets of followers and leaders are denoted as $F = \{1, 2, 3, \ldots, M\}$ and $L = \{M + 1, M + 2, M + 3, \ldots, N\}$ respectively. Additionally, a virtual target symbolized as $\mathcal{T}$ and is labeled as the $(N + 1)^{\text{th}}$ node. Here, we assume that the formation-containment problem requires only one virtual target to guide all the networked leading agents.

-For leader agent: the formation tracking error ξ_k for all $k \in L$ in relation to the virtual target $\mathcal{T}$ and $j(j \in L)$ is defined as:

$$\xi_k = \sum_{j=M+1}^{N} a_{kj} \left((p_k - \mathbf{h_k}) - \left(p_j - \mathbf{h_j} \right) \right) + a_{k\mathcal{T}} \left((p_k - \mathbf{h_k}) - p_{\mathcal{T}} \right) \quad \forall k \in L \tag{15}$$

Where a_{kj} and $a_{k\tau}$ are elements in corresponding adjacency matrix which represents the communication weight from j to k or $\mathcal{T}$ to k. The terms $\mathbf{h_k}$ and $\mathbf{h_j}$ represents predefined desired positional offset of agent k and j relative to the virtual target.

-For a follower agent: the containment error ζ_i for all $i \in L$, in relation to both leader and agent j (where $j \in F \cup L$), is defined as:

$$\zeta_i = \sum_{j=1}^{N} a_{ij} \left(p_i - p_j \right) \quad \forall i \in F \tag{16}$$

Where, a_{ij} are elements in corresponding adjacency matrix, defined as $a_{ij} = 1$ for communication between agent j to agent i, and 0 otherwise. The containment error ζ_i reflects the positional discrepancy between a follower and all other agents within the network, weighted by their direct communication links, which are quantified as 0 or 1 .

Control Protocol (controller 1 + controller 2): Consider N single-integrator dynamic agents, interconnected through a directed graph G that satisfies Assumption (1) to (4). In this case, the MAS can achieve formation-containment tracking under the guidance of the following distributed control protocol:

$$\begin{cases} u_{k,\,\text{formation}} = -k_f \xi_k - \gamma \,\text{sgn}\left(\xi_k\right) & \forall k \in L \\ u_{i,\,\text{containmet}} = -k_c \zeta_i - \mu \,\text{sgn}\left(\zeta_i\right) & \forall i \in F \end{cases} \tag{17}$$

Where k_f, k_c, γ, μ are all positive constants (> 0) , $u_{k,\,\text{formation}}$ is controller for leader and $u_{i,\,\text{containmet}}$ is controller for follower.

After discussing the controllers for both followers and leaders, we now turn to the control input (or controller) for the virtual target. As outlined in several studies [14], a common and straightforward approach is to define virtual target's control input be a time-dependent function $f(t)$ that predetermines its trajectory at various time points. In other words, this virtual target provides a reference signal for the leader, which consists of the position information of the target denoted as $p_{\mathcal{T}}$, The control input, as well as the dynamics for the virtual target, are given as follows:

$$\dot{p}_{\mathcal{T}}(t) = u_{\mathcal{T}}(t), \quad u_{\mathcal{T}}(t) = f(t) \tag{18}$$

For example, the path of the virtual target could be predefined by the user as a dotted line, as shown in Figure 6.

Additionally, if agent encounter obstacles along their path or face potential risks of collision, it is necessary to incorporate an avoidance mechanism within their individual controllers. To address this need, we will detail the design of an APF-based collision avoidance controller in the subsequent part.

***Simulation 3**: Formation-Containment Control (with dynamic leaders being guided by a path predefined by a virtual target.)*

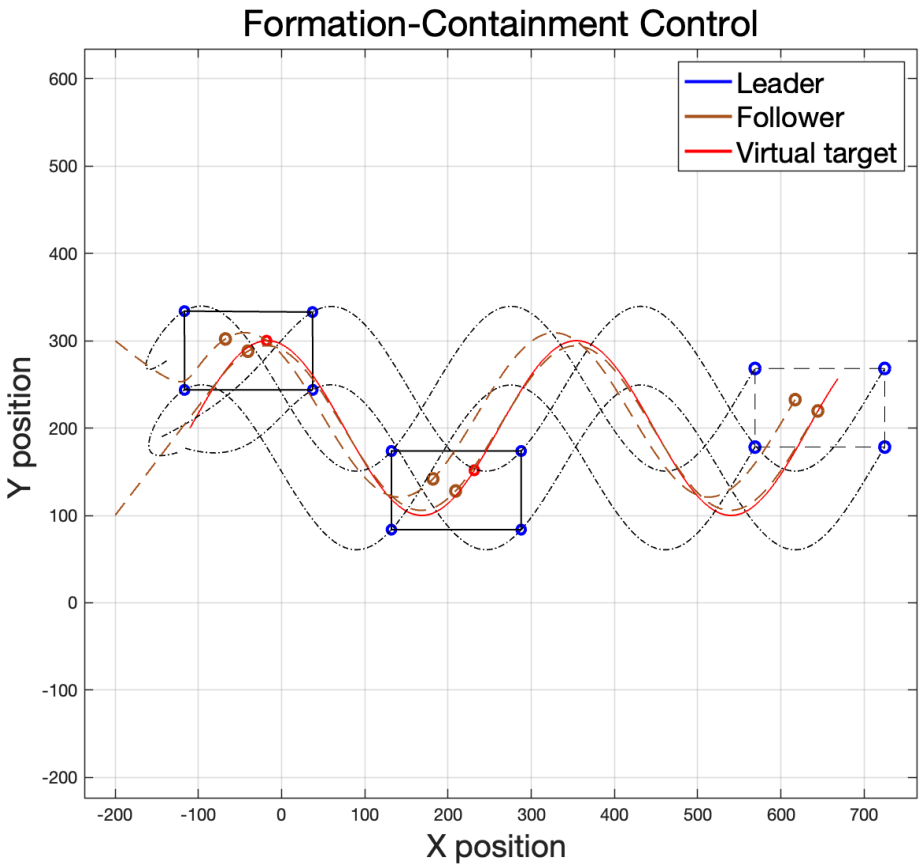


Figure 7: Using protocol (17) within a fixed network topology, the agents' trajectories are shown. Snapshots at 3.6s, 12s, and 30s present followers continuously stay within the moving convex hull shaped by the leaders, who maintain their formation while following a virtual target.

Fig.6 presents the trajectories of the leader, followers, and a virtual target during the execution of formation-containment control. Initially, the leader and followers are placed in random positions, while the virtual target follows a predefined time-varying sinusoidal path $p_{\mathcal{T}}$, given by,

$$p_{\mathcal{T}} = \begin{bmatrix} \text{initial } x + vt \\ \text{initial } y + A\sin(2\pi ft) \end{bmatrix} \tag{18}$$

Here, v,A,f represent the speed in the x-direction, amplitude, and frequency, respectively.

The simulation demonstrates two outcomes: 1) initially, the leader forms a regular formation, and the followers move into formation simultaneously; 2) the leader persistently tracks the virtual target, whereas the followers maintain tracking of the leader. The agent's interaction topology/ communication graph G shown below.

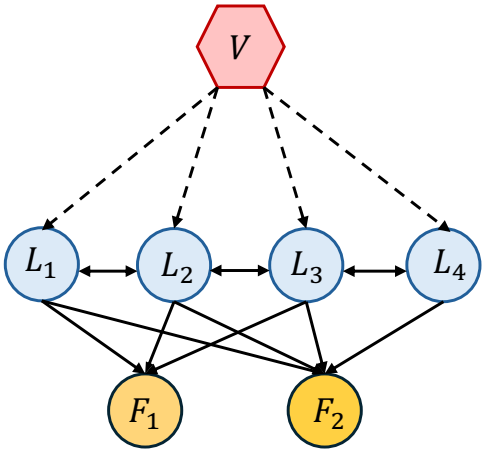


Figure 8: Interaction topology for the simulation 3. V is virtual target, L_1 - L_4 are leaders,F_1 - F_2 are followers.

3.4 Distributed Path Planning Controller Design

In the cooperative framework design for our MAS, it is essential to keep the safety of agents which means we need to address the path/motion planning problem. By employing the APF method, we have implemented a path planning controller that enables both types of agents to navigate safely.

In this section, inspired by [29] [14] and [40], we consider address the obstacle and collision avoidance problems by using APF method, where this algorithm was first proposed by Khatib in 1985 [29], which is abstracted from the interaction between electric charges. The general idea for APF is a virtual force approach, where agents navigate by interacting with potential fields. These fields are of two types: attractive, guiding agents towards the goal, and repulsive, preventing collisions with obstacles or other agents. In our scenario, two distinct types of APF are designed and we only consider repulsive as follows.

- APF approach for obstacles avoidance

Firstly, regarding obstacle avoidance, each obstacle in the environment is treated as a source of high potential. As an agent approaches an obstacle, the field generates repulsive forces, steering the agent away from the obstacle. This repulsive potential field is designed to react to both static and dynamic obstacles, ensuring that agents can move safely and efficiently.

Specifically, the APF method establishes a repulsive potential field, $U_{i,rep}^{\text{obstacle}}\ (p_i(t))$, generated by an obstacle. This field exerts a repulsive force on agent i when it is at the position $p_i(t)$.The expression of this repulsive potential field is dependent on an effective distance $d_{io,eff}$ between the agent and the obstacle (as shown on Fig. 1) which can be calculated as:

$$d_{io,eff} = \|p_i - p_o\| - (r_i + r_o + d_{\text{safe}}) \tag{19}$$

Where $d_{io,eff}$ is the effective distance between the agent at position $p_i(t)$ and the obstacle at position p_o, which various over time as the agent moves. It represents the distance for generating repulsive forces, ensuring that the agent keeps a safe distance from the obstacle. When $d_{io,eff}$ is less than or equal to a predefined maximum activate distance d_{max} (a constant), the repulsive potential field is activated and can be expressed as:

$$U_{i,\text{ rep}}^{\text{obstacle}}\ (p_i(t)) = \begin{cases} \frac{1}{2} k_{\text{rep}} \left(\frac{1}{d_{io,eff}} - \frac{1}{d_{\text{max}}} \right)^2 & \text{if } d_{io,\text{ eff}} \leq d_{\text{max}} \\ 0 & \text{if } d_{io,\text{ eff}} > d_{\text{max}} \end{cases} \tag{20}$$

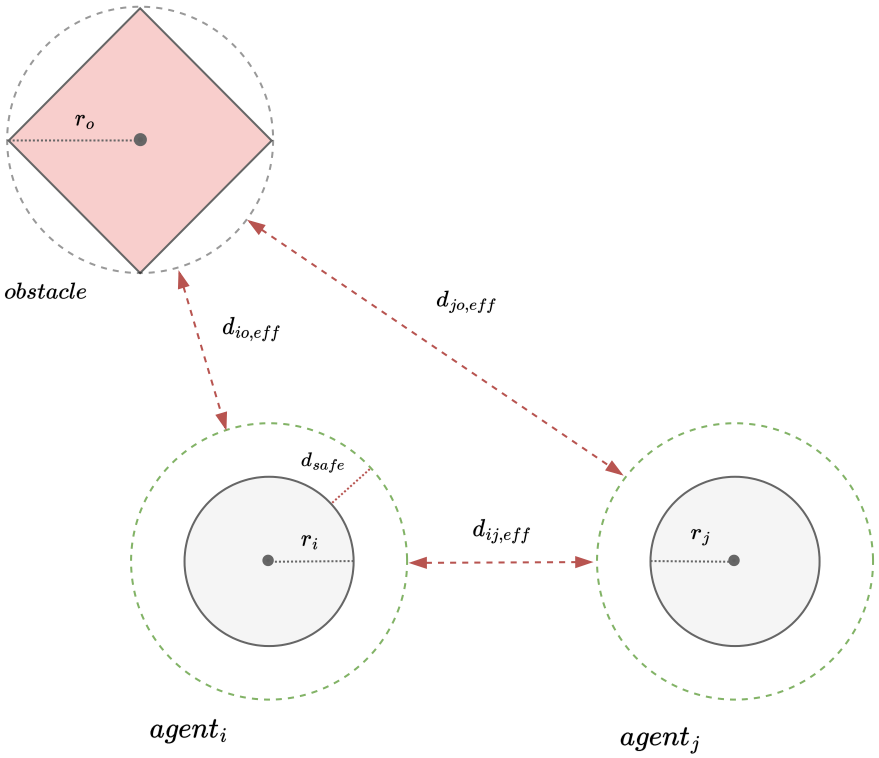


Figure 9: A Schematic representation of effective distances $(d_{io,eff}, d_{jo,eff})$, radii (r_i, r_o, r_j), and safe distance (r_{safe}).Note that the dimensions of the agents and obstacle are not to scale.

Here, k_{rep} is the gain factor of the repulsive force potential field, $k_{rep} > 0$. This formula indicates that when an agent i enters the repulsion zone of an obstacle(within range in $d_{\max}$), it experiences a force that increases as the distance between the agent and the obstacle decreases, thereby compelling the agent to alter its path to avoid a collision. Noted the selection of $d_{\max}$ is crucial as it determines maximum distance at which an agent begins to feel the repulsive force caused by obstacle and needs to change its path. By choosing it properly, we can ensure that the agent can predict and respond to collision risks at a safe enough distance. In some of other literature, the $d_{\max}$ also known as the limits or range of influence of an obstacle.

Having talked the repulsive potential field, we now turn our attention to the repulsive force $F_{i,\text{ rep}}^{\text{obstacle}}\ (p_i(t))$, which is defined as the negative gradient of the $U_{i,\text{ rep}}^{\text{obstacle}}\ (p_i(t))$ and is the driving influence that modulates the agent's motion in real-time, which is given by:

$$F_{i,\text{ rep}}^{\text{obstacle}}\ (p_i(t)) = -\nabla U_{i,\text{ rep}}\ (p_i(t)) \tag{21}$$

Then we just apply the derivation of $U_{i,rep}\,(p_i(t))$ to find the exact representation of repulsive force applied on agent i by one of the obstacles, which is shown as follows,

$$F_{i,\text{ rep}}^{\text{obstacle}}\ (p_i(t)) = \begin{cases} k_{\text{rep}} \left(\frac{1}{d_{io,eff}} - \frac{1}{d_{\max}}\right)^2 \frac{1}{(d_{io,eff})^2} \nabla_{p_i(t)} d_{io,eff} & \text{if } d_{io,\text{ eff}} \leq d_{\max} \\ 0 & \text{if } d_{io,\text{ eff}} > d_{\max} \end{cases} \tag{22}$$

The gradient $\nabla_{p_i(t)} d_{io,\text{ eff}}$ shows the direction in which the effective distance between agent i and the obstacle increases most rapidly. It also determines the direction of the repulsive force to ensure that the agent moves in a way that maximizes this "agent-obstacle" distance.According to Objective (19), noted $\nabla_{p_i(t)} d_{io,eff} = \nabla_{p_i(t)} \left(\|p_i - p_o\| - (r_i + r_o + d_{\text{safe}}) \right) = \frac{p_i(t) - p_o(t)}{\|p_i(t) - p_o(t)\|}$, since all r_i, r_o and d_{safe} are constant.

- APF approach for collision avoidance

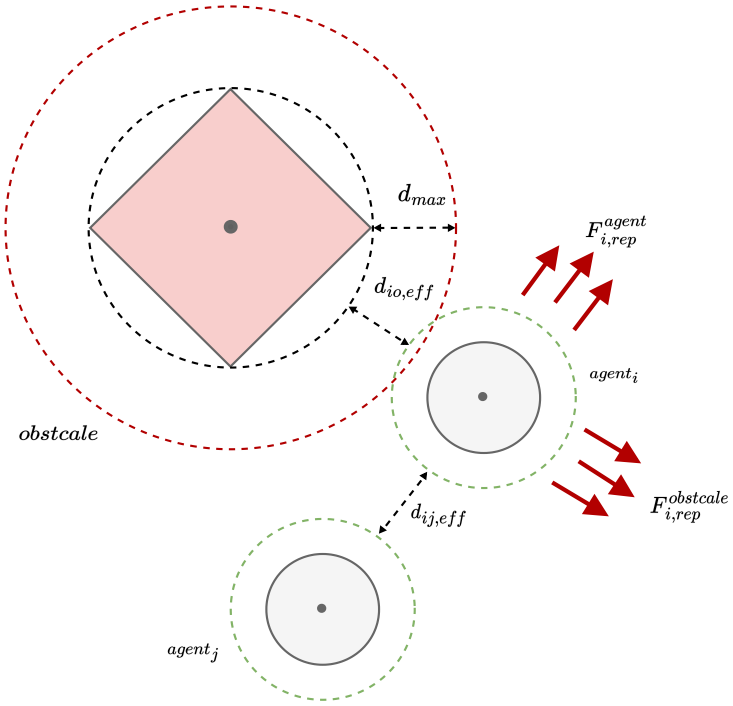


Figure 10: Scenario shows the obstacle and collision avoidance via the APF method. Two repulsive forces are generated to push the agent i towards to the safe zone when both $d_{ij,eff}$ and $d_{io,eff} \leq d_{\max}$.

For collision avoidance among agents, we treat each agent as a high potential point in its own field. If another agent approaches too closely, the APF generates repulsive forces, effectively driving the other agents away to prevent collisions. This ensures that each agent maintains a safe distance from its peers. Considering there are two adjacent agents i and j each with radii r_i and r_j respectively, and a predefined safety distance for each agent d_{safe} is given. Then the effective repulsive distance range between two agents will be similar as objective (19):

$$d_{ij,eff} = \|p_i - p_j\| - (r_i + r_j + 2d_{safe}) \tag{23}$$

Thus, for the repulsive potential field $U^{\text{agent}}_{i,\text{ rep}}\ (p_i(t))$, induced by agent j on agent i at position $p_i(t)$ can be defined as follows:

$$U^{\text{agent}}_{i,\text{ rep}}\ (p_i(t)) = \begin{cases} \frac{1}{2} k_{\text{rep}} \left(\frac{1}{d_{ij,eff}} - \frac{1}{d_{\max}} \right)^2 & \text{if } d_{ij,eff} \leq d_{\max} \\ 0 & \text{if } d_{ij,eff} > d_{\max} \end{cases} \tag{24}$$

Where $d_{\max}$ can be regard as a threshold distance at which the repulsive force between an agent and another agent is activated. Similar to (8), the repulsive force exerted on agent i by another agent j is defined as the negative gradient of $U_{i,\text{ rep}}^{\text{agent}}\ (p_i(t))$,where,

$$F_{i,\text{ rep}}^{\text{agent}}\ (p_i(t)) = \begin{cases} k_{\text{rep}}\ \left(\frac{1}{d_{ij,eff}} - \frac{1}{d_{\max}}\right)^2 \frac{1}{(d_{ij,eff})^2} \nabla_{p_i(t)} d_{ij,\text{ eff}} & \text{if } d_{ij,eff} \leq d_{\max} \\ 0 & \text{if } d_{ij,\text{ eff}} > d_{\max} \end{cases} \tag{25}$$

A simple demonstration for obstacle and collision avoidance are given shown on Fig. 7.It's worth to mention that the force $F_{i,rep}^{\text{agent}}\ (p_i(t))$ is vector pointing for agent j to agent i.Similarly, since the repulsive forces between agents are mutual, agent j is also experienced a force of the same magnitude from agent i.

- In summary

When considering obstacle and collision avoidance in a MAS with Q obstacles and N agents, if there is an obstacle or a neighboring agent approach to agent i a repulsive force is generated to push that agent into a safer location that away from potential collisions with either the obstacle or other agents. The total repulsive force $F_{i,\text{ total}}$ acting on agent i at position $p_i(t)$ can be expressed as :

$$F_{i,\text{ total}}\ (p_i(t)) = \sum_{j,j\neq i}^{N} F_{i,\text{ rep}}^{\text{agent}}\ (p_i(t)) + \sum_{q=1}^{Q} F_{i,\text{ rep}}^{\text{obstcale}}\ (p_i(t)) \tag{26}$$

The formula noted as (26) holds valid only if $d_{ij,eff} \leq d_{\max}$ and $d_{io,eff} \leq d_{\max}$ satisfied. Otherwise, the force experienced on agent i is zero. Therefore, in the distributed path planning controller, the control input contributed by the obstacle/collision avoidance component can be expressed as follow,

$$u_{i,\text{ obs / collision}} = F_{i,\text{ total}}\ (p_i(t)) \tag{27}$$

***Simulation 4**: Demo on path planning and reciprocal avoidance(agent-agent)*

- path planning(repulsive field and attractive field)

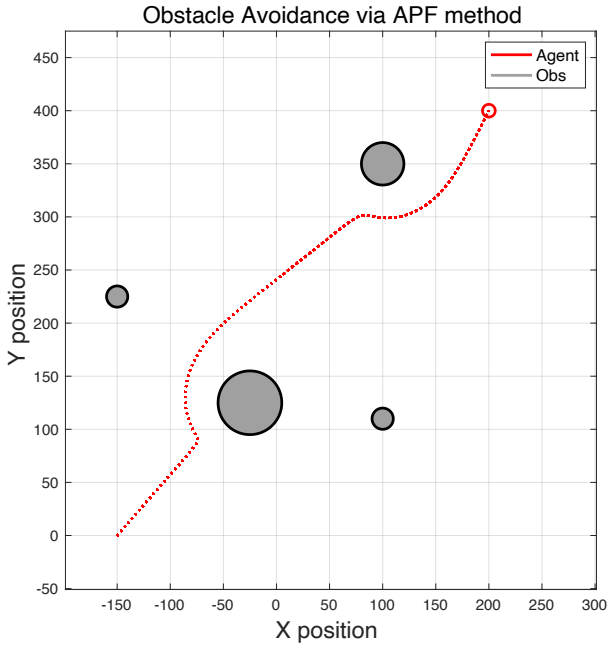


Figure 11: Real time path planing for a single agent without known prior environment information

In Fig 8, the graph shows how obstacle avoidance is managed for the agent among static obstacles using the APF method. When the agent detects an obstacle and the condition $d_{io,eff} \leq d_{max}$is true, a virtual repulsive force is calculated as a control input. These inputs change the direction of the agent, steering it towards areas where the repulsive potential decreases. At the same time, an attractive field remains to pull the agent towards its target. This graph demonstrates the agent's real-time path planning objectives.

-Reciprocal avoidance(agent-agent collision avoidance)

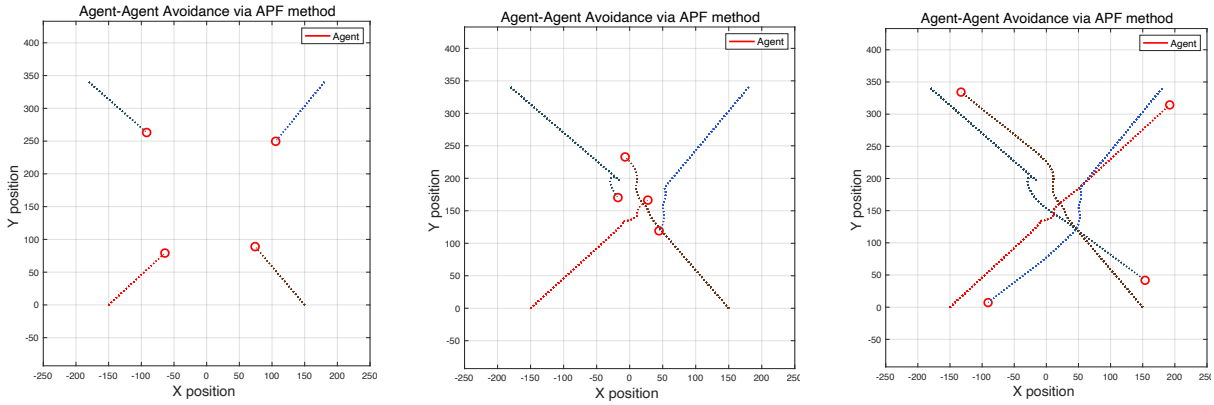


Figure 12: Three snapshots at different moments, showing that the agents can reciprocal avoidance.

3.5 Summary

Consider a MAS comprising $N+1$ nodes, include a virtual Target $\mathcal{T}$, followers i and leaders k with N single-integrator dynamic agents as described by equation (1). These agents are interconnected through a directed graph G that meets Assumptions 1 to 5. Within this system, followers $i \in F$ and leader $k \in L$, where $F = \{1, 2, 3, \ldots, M\}$ and $L = \{M+1, M+2, M+3, \ldots, N\}$ navigating in an environment contain total Q obstacles. To achieve two primary objectives: 1) MAS accomplishes formation-containment tracking relative to a virtual target 2) ensures safe motion (obstacle/collision avoidance), the following distributed control protocol is proposed based on equations (17) and (27).

$$\begin{cases} u_k = u_{k,\text{ formation}} + u_{k,\text{ obs / collision}} = -k_f \xi_k - \gamma \operatorname{sgn}(\xi_k) + F_{k,\text{ total}} & \forall k \in L \\ u_i, = u_{i,\text{ containment}} + u_{i,\text{ obs / collision}} = -k_c \zeta_i - \mu \operatorname{sgn}(\zeta_i) + F_{i,\text{ total}} & \forall i \in F \end{cases} \tag{28}$$

This protocol ensures that the MAS not only follows the virtual target in a predefined formation but also navigates safely through the environment, addressing both formation-containment tracking and obstacle/collision avoidance. Protocol (28) forms the cornerstone of our hierarchical cooperative control framework, while some resilience strategies such as formation scaling and reconfiguration acting as redundancy policies to increase the robustness of our framework.

Algorithm 2 Distributed formation-containment control approach consider path planing.

Initialization: $G, h_f, k_f, k_c, \gamma, \mu, d_{\text{max}}, d_{\text{safe}}, k_{\text{rep}}, F, L, \mathcal{T}$ etc.
for each agent $i \in \{1, \ldots, N\}$ **do**
  if i is a leader $(i \in L)$ **then**
    Receive virtual target position $p_{\mathcal{T}}$ and neighbors' leader positions $p_{j \in \mathcal{N}_i}$ via checking $a_{i\mathcal{T}}$ and a_{ij} in G.
    Update ξ_k and $u_{i,\text{formation}}$.
    if $d_{io,\text{eff}}$ or $d_{ij,\text{eff}} \leq d_{\text{max}}$ or formation scaling/reconfiguration is required **then**
      Update $u_{i,\text{obs/collision}}$ or ξ_k and $u_{i,\text{formation}}$.
    end if
    Compute the total control input $u_i = u_{i,\text{formation}} + u_{i,\text{obs/collision}}$.
  else
    if i is a follower $(i \in F)$ **then**
      Receive $p_{j \in L \cup F}$ for both leader positions and neighbor followers' positions via checking a_{ij} in G.
      Update $u_{i,\text{containment}}$.
      if $d_{io,\text{eff}}$ or $d_{ij,\text{eff}} \leq d_{\text{max}}$ **then**
        Update $u_{i,\text{obs/collision}}$.
      end if
      Compute the total control input $u_i = u_{i,\text{containment}} + u_{i,\text{obs/collision}}$.
    end if
  end if
  Apply the control input u_i (if $i \in L$) or u_i (if $i \in F$) to all agents at the same time.
end for
Repeat steps 2-21 until the formation-containment objective is achieved and all agents are navigating safely.

4 Case Study: A Real Time Rescue Mission

4.1 Simulation Result

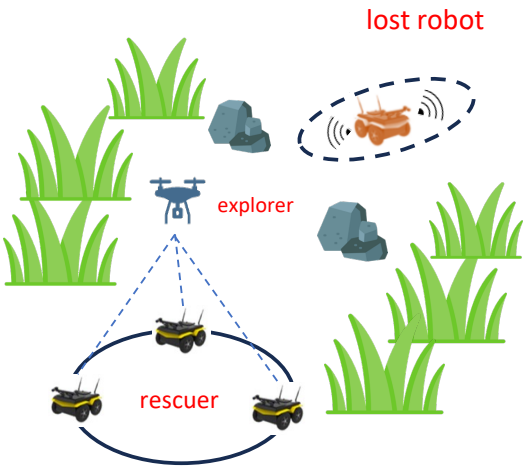


Figure 13: Graphical demonstration of rescue mission in the wild.

In this section, we demonstrate the effectiveness of the proposed hierarchical cooperative control framework through a MATLAB simulation of a rescue mission case study. Specifically, we evaluate the performance of the framework by examining whether the error of an agent (the difference between the target and current positions) can converge to near zero. We noted that this work is based and extended on study [5]. Additionally, features such as formation-containment, obstacle avoidance, and collision avoidance among agents are demonstrated in our simulation. This simulation highlights the potential application of our proposed framework in real-world robotic cooperative experiments, and also confirms the viability of our control algorithm.

-Rescue mission simulation problem statement and objective:

The rescue mission involves deploying a team of robots, including explorers and rescuers, on a search mission in a wild environment filled with unknown obstacle s(no prior map information given), as shown in Fig 13. The objective is to locate and safely bring the lost robot back to a secure location through our coordinated control framework. We assign 3 different components in our whole system: virtual target, leaders, and followers. They are specifically as the following roles:

1. **Follower (lost robot)**: The follower is designated as the robot that has lost contact in the wild due to a malfunction in its obstacle detection sensors, its be assumed with a very limited communication range that cannot return independently, only when rescue team approaching to the lost robot, short distance communication link build, then back to the secure position.

2. **Virtual Target (explorer's projection)**: The virtual target is conceptualized as the two-dimensional projection of an unmanned aerial vehicle (UAV). This projection is tasked with exploring unknown environments to locate the lost robot and utilizing this projection to guide rescuers to the vicinity of the lost robot.

3. **Leader (Rescuer)**: The leader is identified as the rescuer, specifically an unmanned ground vehicle (UGV), whose role is to navigate to the lost robot's location under the guidance of the explorer's projection. The leader then facilitates the movement of the lost robot into a safe area, enclosed by the rescuers' formation (a convex hull), and leads them (lost robot) back to a secure location. Obstacles detection sensors are equipped on each rescuer agents.

It's important to note that UAV are not only tasked with exploring and locating lost robots on a two dimensional plane but also with accounting for ground obstacles. In other words, UAVs are required to detect ground obstacles and ensure that their projection onto the two-dimensional plane can navigate around these obstacles. This is done to provide a group guidance system for obstacle avoidance to the rescuers (rescuers) on the ground, enabling them to move safely in formation during rescue missions. Additionally, each rescuer must also possess the capability to autonomously avoid obstacles when necessary.

In our simulation verification, the path of the virtual target is not predetermined. Instead, we treat the projection as an agent that implements obstacle avoidance through the APF algorithm.

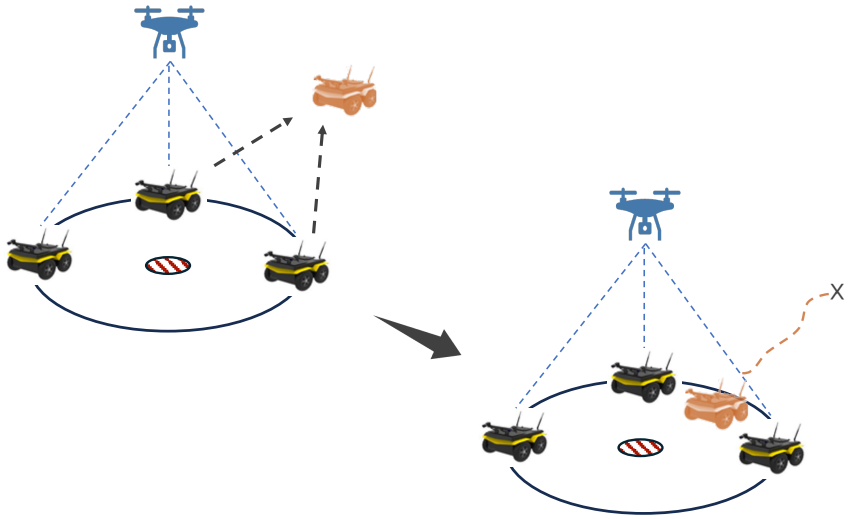


Figure 14: The lead robots form a triangular formation, continuously tracking the virtual target, and when they come near the Lost Robots, the two lead robots start communicating with the lost robots (black dashed lines). Finally, the lost robots to converge into the convex hull via containment control.

-Simulation result

The simulation was conducted using the MATLAB platform, and the simulation code and relevant videos are available at the GitHub link provided in Appendix. Aside from MATLAB's basic functions, we did not adopt any external libraries or toolboxs to achieve the relevant control objectives.

Before presenting the simulation results, it is worth introducing a key concept essential for enhancing the automation of MASs: the high-level task allocation system. This level aims to design rational algorithms that ensure tasks are completed efficiently despite limited resources. For instance, consider a scenario involving 2 UAVs, 6 UGVs, and 3 lost robots scattered across different locations. The algorithm needs to decide the number of swarms, design the rescue sequence, and configure the number of rescuers to complete the mission effectively under constrained conditions. Research on multi-agent system task

allocation can be found in [42]. In our simulation, problems such as the number of swarms, swarm size, and rescue sequence are predefined.

Now, we present our simulation results. The simulation outcomes are illustrated in Figures 15 (a) to (g). In this simulation, a group of 15 robots, which includes 2 explore UAVs, 10 rescue UGVs, and 3 lost robots in placed in a 3D environment. The action sequence shown below.

1. Task Allocation: Depending on the positions and priorities of the explorer, the high-level task allocation system decides to split the 10 rescue robots into 2 clusters, with each cluster comprising 5 rescue robots and 1 explorer, forming a swarm team. Among the 3 lost robots, the first swarm is assigned to rescue one lost robot, while another swarm targets the other two lost robots.

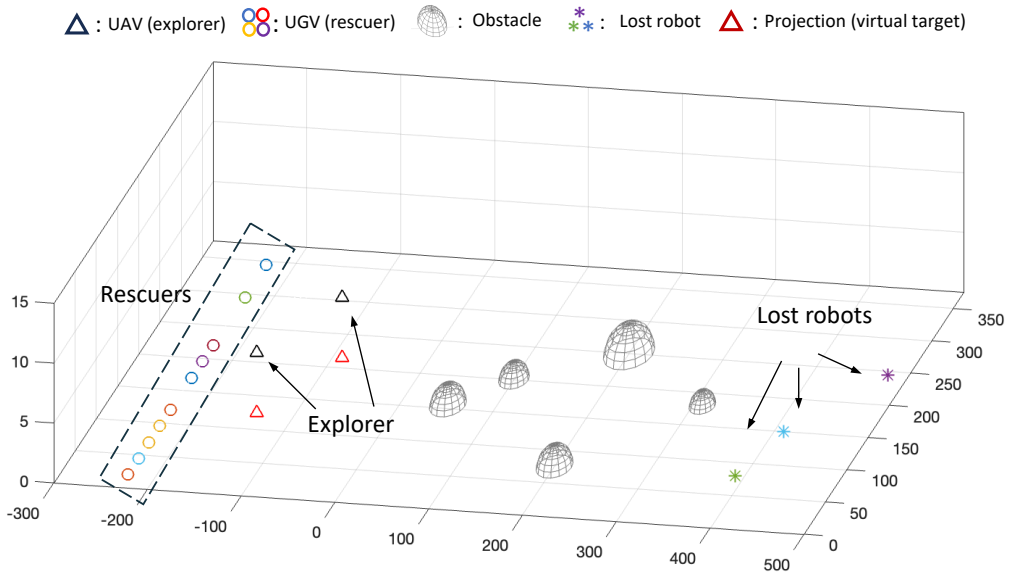
2. Formation and Tracking: Rescue robots in each cluster/swarm form their specific shape called formation and continue tracking the virtual target provided by the UAVs.

3. Search by Explorer UAV: Exploiting the advantage of a wide aerial view while flying, the explorer UAV searches for lost robots. Upon locating one, the UAV begins safely guiding the ground rescuers to the vicinity of the lost robot.

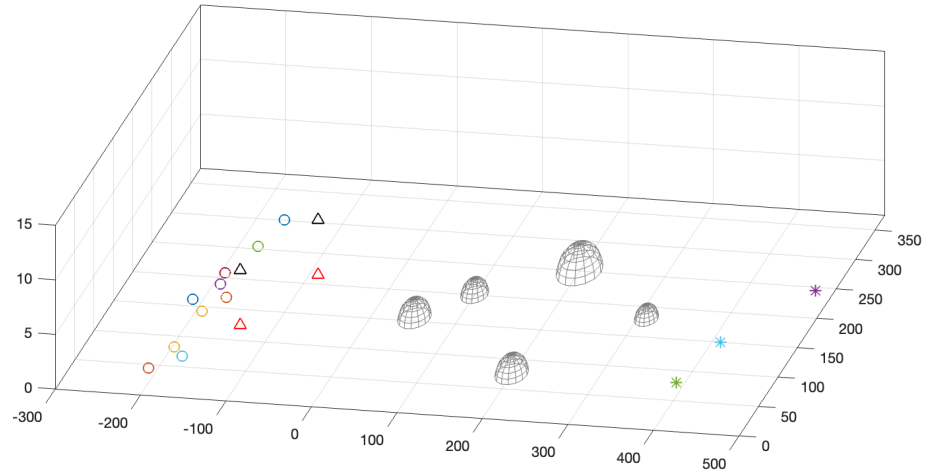
4. Rescue Operation via containment: When the rescuer cluster reaches the nearby of the lost robot, "some" or "all" of the rescuers start communicating with it and, eventually, make it converge inside the formation. This process is governed by the containment controller within the lost robot.

5. Return Phase: Safely guide the rescued robot back to a predetermined safe place.

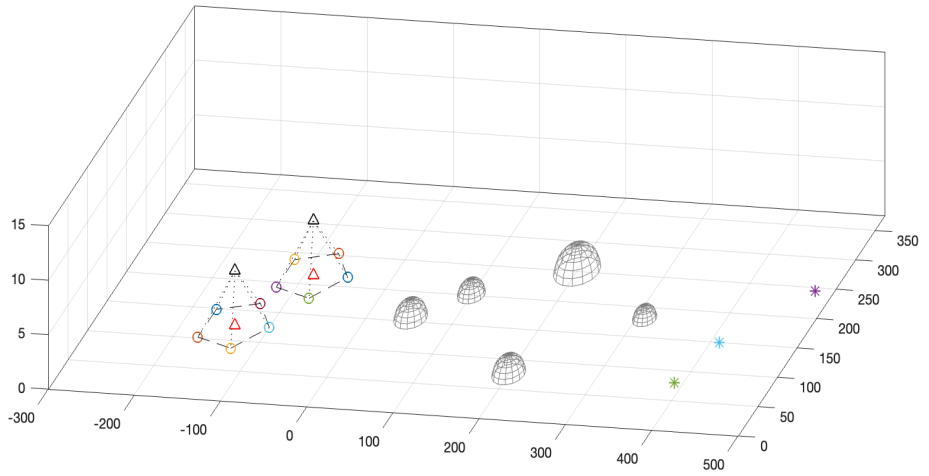
Note that throughout the entire experiment, the obstacle and collision avoidance mechanism must be implemented.



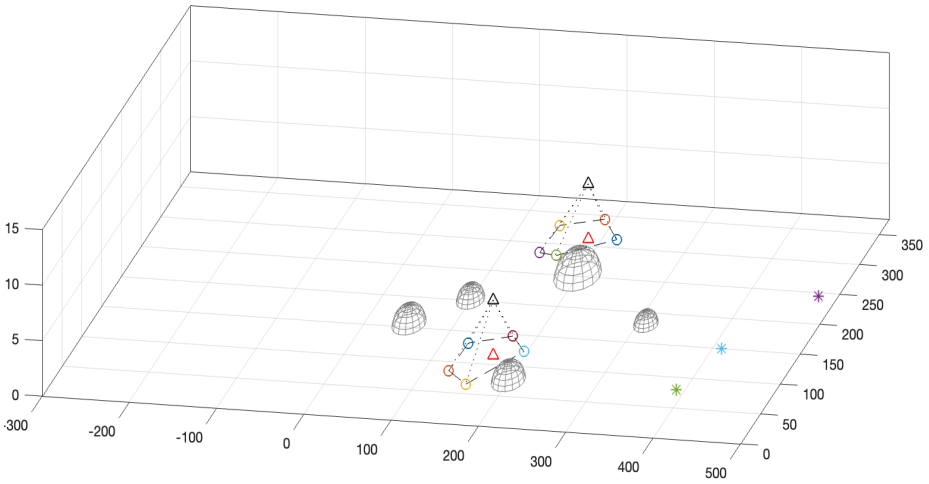
(a) The initial state of all agents.



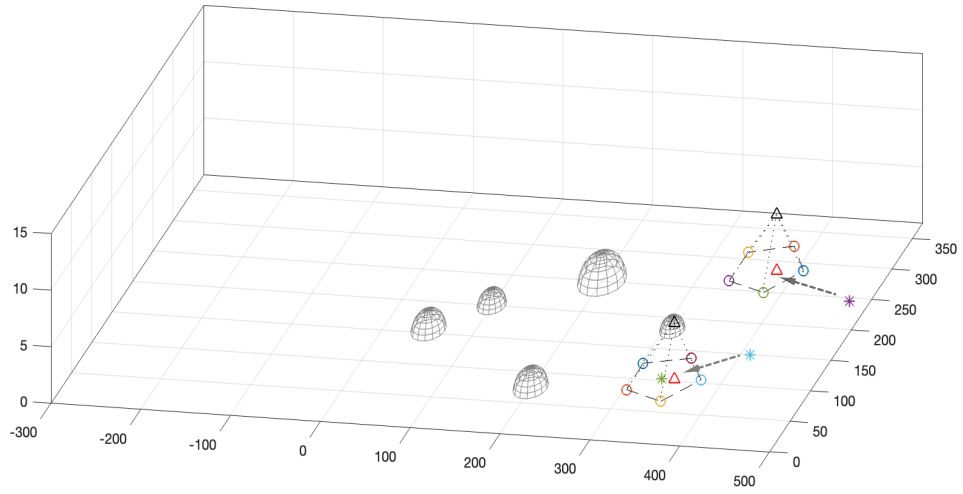
(b) The rescuers are randomly divided into two clusters and gradually form a formation towards their goal, during which the agents avoid each other.



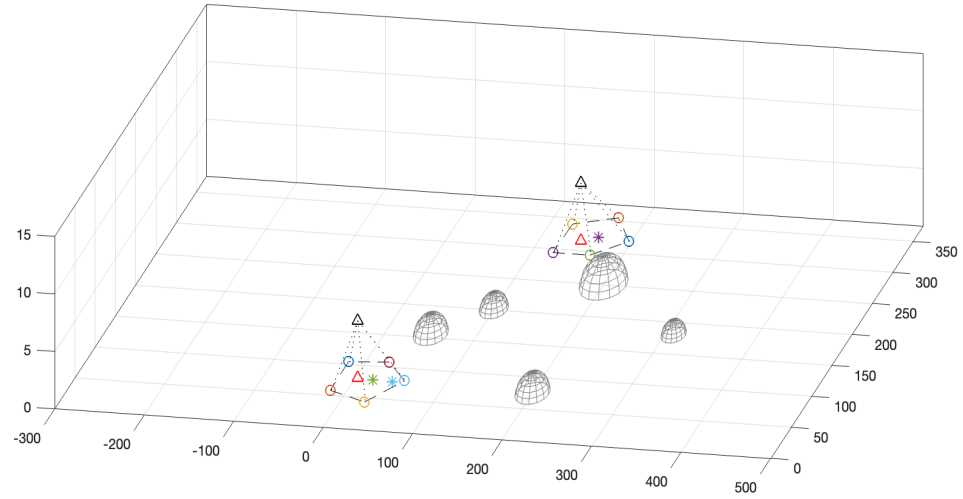
(c) Two groups of pentagonal formations are formed.



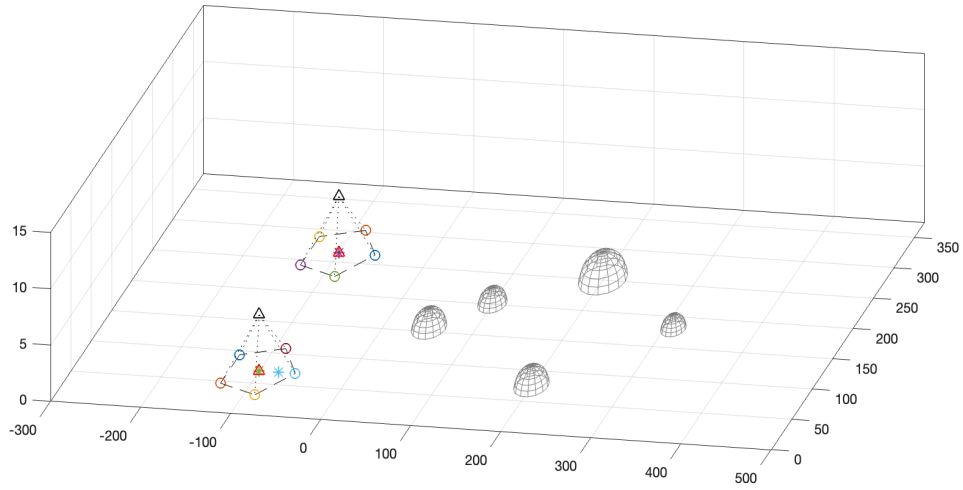
(d) Explorers lead team to move towards the location of the Lost robot, while avoid obstacles during process.



(e) The leader find followers, starts communicating with them, eventually converges them into a convex hull.



(f) Lost robot continuously tracks the convex hull, while the leader robot avoids obstacles to ensure the Lost robot is not damaged.

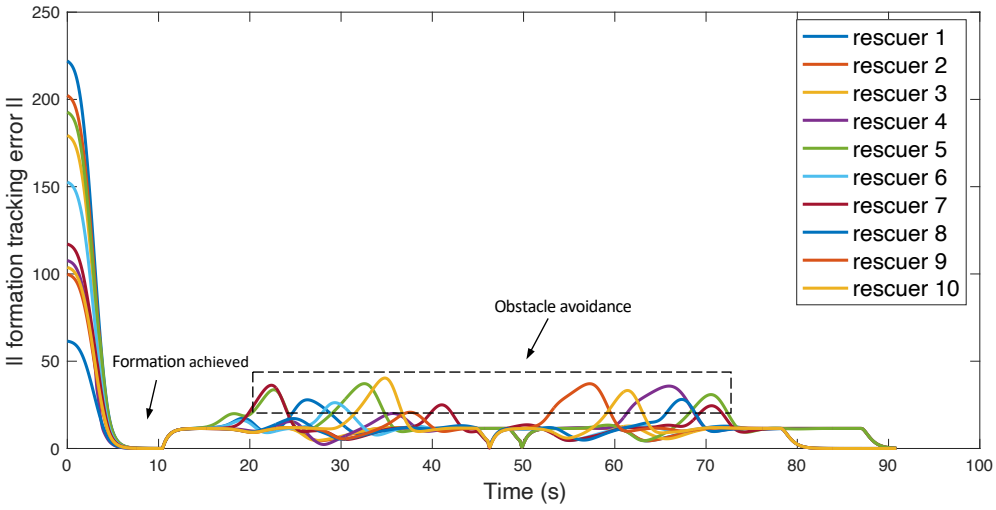


(g) Eventually, each agent reaches its respective target position. Note that a lost robot only communicates with three leaders, so its final position is the weighted average position of the three nearby leader robots.

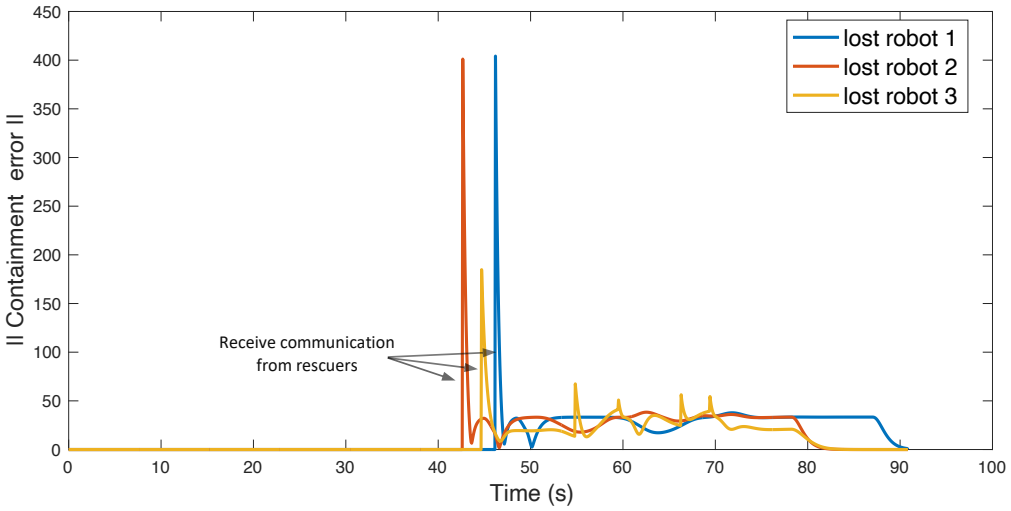
Figure 15: The agent's state at seven different moments (a) - (f) during a rescue mission.

In Figure 15, the status of each agent is depicted, with the system comprising UAVs (explorers) represented by black triangles, UGVs (rescuers) denoted by circles, lost robots indicated with stars, and a virtual target shown as a red triangle. It is noteworthy that the red triangle represents the UAV's projection within the two-dimensional plane, and obstacles are displayed in the environment. At this moment, the UAV sends the projection's position to the rescuer followers via the communication network. It is noted that, in this simulation, we assume all rescuer UGVs in one swarm are connected with their explorer UAV. (b) shows the rescuers being randomly divided into two swarms, gradually approaching the corresponding virtual target, and forming a formation in the process, allowing them to avoid collisions with each other. (c) displays a team of rescuers achieving a pentagon formation surrounding the virtual target via the formation controller, after which the virtual target will guide the rescuer swarm to the vicinity of the lost robot. The black dashed line represents the formation of rescuers, and the black dotted line represents the communication link between rescuers and explorers. (d) demonstrates that during the process, the swarm can avoid obstacles while maintaining the original pentagon shape as much as possible. (e) shows one swarm arriving near lost robots 2 and 3, with the two lost robots entering the safe region inside. Meanwhile, another swarm is approaching a lost agent. (f) indicates that all lost robots have been found and guided into the safe area, navigating around obstacles (containment achieved). (g) confirms that all lost robots have been returned to the safe area, thus successfully completing the search and rescue mission. We note that we configured lost robot 2 (indicated by the blue star) to communicate only with three rescuers in its corresponding swarm. Therefore, as shown in figure (f), it ultimately stabilized at the centroid of the right three rescuers. In contrast, we set the other two lost robots to communicate with all the rescuers within their respective swarms, leading their final positions to align with the centroid of a pentagon, which is the average position of these five lost robots.

In the simulation, we plotted the 2-norms of the formation tracking error for the rescuers, denoted as $\|\zeta_{\text{rescuers}}\|$, and the containment error for the lost robots, denoted as $\|\xi_{\text{lost}}\|$, to demonstrate that the control errors converge to near zero under the proposed protocol. These results are presented in Figure 12(a) and (b).



(a) $\|\zeta_{\text{rescuers}}\|$ - t graph



(b) $\|\xi_{\text{lost}}\|$ - t graph

Figure 16: Formation tracking error $\|\zeta_{\text{rescuers}}\|$ and containment error $\|\xi_{\text{lost}}\|$.

As depicted in Figure 16 (a), it shows the formation tracking error various with the time. Initially, due to the influence of the formation controller, the tracking error converged to zero within the first 10 seconds. Between 20s and 46s, the rescuer team navigated past various obstacles, resulting in increased tracking error, as indicated by the appearance of some 'peaks.' This was caused by the activation of the obstacle avoidance mechanism. At t=46s and t=50s, two rescuer swarms approached the nearby of the lost robots, after 50s, the rescuer leads the lost robot back to the designated safe area, and then the tracking error gradually decreases to zero. During this period, the formation tracking error also increases when there is a need for obstacle avoidance. Notably, a persistent tracking error is observed whenever the UAV's virtual target moves, due to a fixed and relatively small formation control scaling factor k_f. Introducing an adaptive k_f in the control protocol in the future could address this issue, and this need a more advanced controller law need.

Figure 16 (b) shows the containment error, which remains at zero before 40 s as the lost robots had not received any communication signals from the rescuers. Upon the explorers locating the lost robot and the lead rescuer arriving nearby, the lost robots received signals and swiftly moved into the convex hull, evidenced by the high peak in the figure. Subsequently, they returned to a safe area under the guidance of the rescuers. Similar to the tracking error, a persistent containment error occurs during movement due to a small value of k_c. Designing an adaptive containment error scaling factor, denoted as k_c, could address this issue.

4.2 Discussion

The experimental results suggests that the proposed cluster-based formation containment and collision avoidance controllers successfully guided the agents in completing a rescue mission. The Fig 16 (g) also show that the system's error eventually approaches zero.However, it's important to note that the simulation was conducted in an ideal environment. In the real world, various practical constraints also need to be considered: However, it's important to note that the simulation was conducted in an ideal environment. In the real world, various practical constraints also need to be considered and can be conducted

in the future work: 1) communication delays, also known as time delays; 2) the non-linear properties of agent dynamics; 3) physical environmental problems, such as motion friction and external turbulence (e.g., wind); 4) system hardware faults, including inaccuracies in the perception or sensing systems, and sensor or actuator faults.

In addition, to enhance the robustness of our control algorithms, there are several aspects of the algorithms themselves that can be optimized:

1. Fixed-time formation containment problem: In our control algorithm, the convergence of formation tracking and containment errors to exactly zero is guaranteed only over an infinite time horizon. However, in many real-world examples, the network needs to achieve this within a required time due to task-specific demands, necessitating fixed-time control. Literature [34] discusses this issue and provides a solution.

2. Other redundancy/fault-tolerant strategies related to formation control: the simplicity of the leader-follower formation control method is widely utilized; however, if the leader encounter a fault, the system could become compromised. Some redundant/ fault-tolerant strategies, such as a leader res-election mechanism from among the followers need to be properly designed.

3. A trade-off exists between the objectives of formation control and obstacle avoidance: agents use the Artificial Potential Field (APF) approach to be repelled from obstacles, which may cause deviations, while the formation objective requires them to track their target positions, that potentially conflicting with obstacle avoidance goals. In our control framework, agents prioritize navigating away from obstacles before reverting to their formation objectives, as the Fig 15 (f) shown, where the formation does not maintain its pentagonal shape during obstacle avoidance maneuvers. Hence, designing a trajectory planning algorithm that simultaneously addresses both formation maintenance and obstacle avoidance in dense environments is a challenging aspect that requires further exploration. (This discussion specifically excludes considerations of containment objectives.)

5 Conclusion and Future Work

5.1 Conclusion

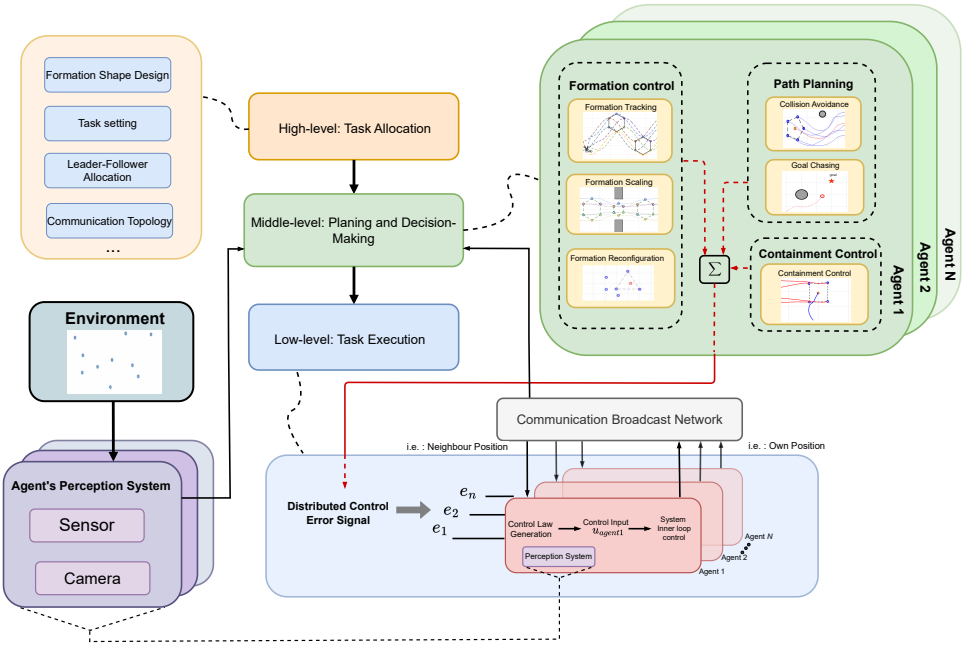


Figure 17: An overview of the hierarchical distributed system framework for multi-agent systems (MAS), focusing on formation containment with path planning considerations.

Multi-agent systems cooperative control boasts a strong engineering background and broad application prospects, making it a highly active topic in both control and robotic research field. This project presents a *hierarchical distributed cooperative control framework* for a homogeneous single integrator multi-agent system. This report presents a hierarchical cooperative control framework designed for a single integrator multi-intelligent system and validates its applicability through a real-time rescue mission simulation.

Within this control framework, relevant controllers and resilience redundancy algorithms are embedded and implemented in a distributed manner, as agents can rely on the relative positional information from its neighbors via a communication network. The proposed control scheme enables networked agents to achieve formation tracking and containment control objectives. Furthermore, by employing a potential repulsive function, an effective path/motion planning algorithm is designed to address the challenges of obstacle and collision avoidance without the need for prior environmental knowledge. The main conclusion shown below.

In the first part, we explored the issue of formation control among multiple agents, specifically employing the leader-follower approach. We developed a distributed formation control protocol that enables all follower agents to form a stable formation based on the leader agent's position, maintaining continuous tracking. To enhance the robustness of formation control, we designed two fault-tolerant algorithms: a) one for quick reformation into a new topology if an agent disconnects, and b) another for formation rescaling when a narrow corridor is detected.

In the second part, we investigated containment control problems within multi-agent systems with static

leaders. We designed a distributed containment control protocol that enables followers to converge within a convex hull (a closed geometric space), of either regular or irregular shape that formed by the leaders.

In the third part, we synthesized the techniques of formation control and containment control, contributing to a formation-containment control protocol. By applying this protocol to a MAS, the leader can create a specific formation, and the followers can move into formation simultaneously.

In the fourth part, to address the need for safe navigation of agents in real-world applications, we incorporated an obstacle avoidance controller based on Artificial Potential Field (APF) theory into the formation-containment control protocol, resulting in a hierarchical cooperative control framework.

Each of these four parts was verified through several simulation experiments, demonstrating the feasibility and effectiveness of the approaches. Additionally, a case study with practical meaning: A real-time rescue mission, was conducted to showcase the applicability of the proposed hierarchical control framework.

5.2 Future Work

There is still considerable scope for improvement in the content of this project, and future research directions as follows:

- Real-world experiments need to be conducted to bridges the gap between theoretical simulations and real-world applications. As mentioned in the discussion on simulations, the gap between simulations and the real world remains significant. Practical constraints such as communication delays, wind turbulence, actuator faults, and sensor inaccuracies must be considered, and proposed control algorithm should be further verified and discussed through real-world experimentation.
- In the practical application, due to the unique characteristics of various robotic platforms, such as aerial and surface robots, further research need to design more comprehensive and advanced robust swarm control techniques to address their distinct nonlinear attributes and complex higher-order dynamics. Additionally, system design (e.g., communication topology, system architecture) must account scalability for large-scale operations.
- High-level decision-making algorithms for multi-agent systems (e.g., task allocation) need to be integrated with the current scheme to collaboratively enhance and achieve a more comprehensive autonomous intelligent system.

From a broader perspective, the robot perception system (localization system, such as purple part on Fig 17), fault detection, robot mechanical architecture design, and robot power management are each pivotal areas contributing to the advancement of robotic systems. These developments enable robots to perform increasingly complex tasks.

In the end of this project, we believe that future research will reveal more intelligent and cooperative swarm behaviors, significantly enhancing task-handling capabilities. In my future studies and research, I am sincerely hoped to provide the robotics community with a more comprehensive, resilient, and realistic set of solutions. We are deeply grateful to the researchers in the robotics and control fields for their continuously effort. It is their relentless pursuit that strengthens our confidence in achieving more intelligent unmanned systems in the future.

Finally, I would like to extend my special appreciation to Dr. Boli Chen and Mr. Hao Sun. I am truly grateful for the meaningful suggestions they have provided over the past few months, whether related to technical issues or personal development.

References

[1] NVIDIA, "Nvidia announces project gr00t foundation model for humanoid robots and major isaac robotics platform update," https://nvidianews.nvidia.com/news/foundation-model-isaac-robotics-platform, Mar. 2024, accessed: 2024-04-08.

[2] Boston Dynamics, "Spot," https://bostondynamics.com/products/spot/, Apr. 2024, retrieved April 5, 2024.

[3] M. Pipattanasomporn, H. Feroze, and S. Rahman, "Multi-agent systems in a distributed smart grid: Design and implementation," in *2009 IEEE/PES Power Systems Conference and Exposition*. IEEE, 2009, pp. 1–8.

[4] J. Hu, A. E. Turgut, T. Krajník, B. Lennox, and F. Arvin, "Occlusion-based coordination protocol design for autonomous robotic shepherding tasks," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 14, no. 1, pp. 126–135, 2020.

[5] J. Hu, P. Bhowmick, I. Jang, F. Arvin, and A. Lanzon, "A decentralized cluster formation containment framework for multirobot systems," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1936–1955, 2021.

[6] Y. Wang, Z. Su, N. Zhang, and R. Li, "Mobile wireless rechargeable uav networks: Challenges and solutions," *IEEE Communications Magazine*, vol. 60, no. 3, pp. 33–39, 2022.

[7] V. Spurnỳ, T. Báča, M. Saska, R. Pěnička, T. Krajník, J. Thomas, D. Thakur, G. Loianno, and V. Kumar, "Cooperative autonomous search, grasping, and delivering in a treasure hunt scenario by a team of unmanned aerial vehicles," *Journal of Field Robotics*, vol. 36, no. 1, pp. 125–148, 2019.

[8] J. Alonso-Mora, S. Baker, and D. Rus, "Multi-robot formation control and object transport in dynamic environments via constrained optimization," *The International Journal of Robotics Research*, vol. 36, no. 9, pp. 1000–1021, 2017.

[9] X. Zhou, X. Wen, Z. Wang, Y. Gao, H. Li, Q. Wang, T. Yang, H. Lu, Y. Cao, C. Xu *et al.*, "Swarm of micro flying robots in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm5954, 2022.

[10] J. Hu, P. Bhowmick, F. Arvin, A. Lanzon, and B. Lennox, "Cooperative control of heterogeneous connected vehicle platoons: An adaptive leader-following approach," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 977–984, 2020.

[11] J. Alonso-Mora, E. Montijano, M. Schwager, and D. Rus, "Distributed multi-robot formation control among obstacles: A geometric and optimization approach with consensus," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 5356–5363.

[12] L. Quan, L. Yin, T. Zhang, M. Wang, R. Wang, S. Zhong, X. Zhou, Y. Cao, C. Xu, and F. Gao, "Robust and efficient trajectory planning for formation flight in dense environments," *IEEE Transactions on Robotics*, 2023.

[13] M. Turpin, N. Michael, and V. Kumar, "Trajectory design and control for aggressive formation flight with quadrotors," *Autonomous Robots*, vol. 33, pp. 143–156, 2012.

[14] G. Wen, C. P. Chen, and Y.-J. Liu, "Formation control with obstacle avoidance for a class of stochastic multiagent systems," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 7, pp. 5847–5855, 2017.

[15] T. Balch and R. C. Arkin, "Behavior-based formation control for multirobot teams," *IEEE transactions on robotics and automation*, vol. 14, no. 6, pp. 926–939, 1998.

[16] M. A. Lewis and K.-H. Tan, "High precision formation control of mobile robots using virtual structures," *Autonomous robots*, vol. 4, pp. 387–403, 1997.

[17] Y. Sun, X. Hu, J. Xiao, G. Zhang, S. Wang, and L. Liu, "Multi-agent cluster systems formation control with obstacle avoidance," in *2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA)*. IEEE, 2020, pp. 1635–1640.

[18] Y.-H. Su and A. Lanzon, "Formation-containment tracking and scaling for multiple quadcopters with an application to choke-point navigation," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 4908–4914.

[19] W. Ren and R. W. Beard, "Consensus seeking in multiagent systems under dynamically changing interaction topologies," *IEEE Transactions on automatic control*, vol. 50, no. 5, pp. 655–661, 2005.

[20] S. Zhao, "Affine formation maneuver control of multiagent systems," *IEEE Transactions on Automatic Control*, vol. 63, no. 12, pp. 4140–4155, 2018.

[21] Y. Cao, D. Stuart, W. Ren, and Z. Meng, "Distributed containment control for multiple autonomous vehicles with double-integrator dynamics: Algorithms and experiments," *IEEE Transactions on Control Systems Technology*, vol. 19, no. 4, pp. 929–938, 2010.

[22] J. Mei, W. Ren, B. Li, and G. Ma, "Distributed containment control for multiple unknown second-order nonlinear systems with application to networked lagrangian systems," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 9, pp. 1885–1899, 2014.

[23] G. Wen, Y. Zhao, Z. Duan, W. Yu, and G. Chen, "Containment of higher-order multi-leader multi-agent systems: A dynamic output approach," *IEEE Transactions on Automatic Control*, vol. 61, no. 4, pp. 1135–1140, 2015.

[24] Y. Yang, H. Modares, D. C. Wunsch, and Y. Yin, "Optimal containment control of unknown heterogeneous systems with active leaders," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 3, pp. 1228–1236, 2018.

[25] X. Dong, Q. Li, Z. Ren, and Y. Zhong, "Formation-containment control for high-order linear time-invariant multi-agent systems with time delays," *Journal of the Franklin Institute*, vol. 352, no. 9, pp. 3564–3584, 2015.

[26] X. Dong, Y. Hua, Y. Zhou, Z. Ren, and Y. Zhong, "Theory and experiment on formation-containment control of multiple multirotor unmanned aerial vehicle systems," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 1, pp. 229–240, 2018.

[27] L. Dai, Q. Cao, Y. Xia, and Y. Gao, "Distributed mpc for formation of multi-agent systems with collision avoidance and obstacle avoidance," *Journal of the Franklin Institute*, vol. 354, no. 4, pp. 2068–2085, 2017.

[28] Y. Yan, X. Li, X. Qiu, J. Qiu, J. Wang, Y. Wang, and Y. Shen, "Relative distributed formation and obstacle avoidance with multi-agent reinforcement learning," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 1661–1667.

[29] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The international journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986.

[30] Z. Pan, C. Zhang, Y. Xia, H. Xiong, and X. Shao, "An improved artificial potential field method for path planning and formation control of the multi-uav systems," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 3, pp. 1129–1133, 2021.

[31] X. Dong, Y. Zhou, Z. Ren, and Y. Zhong, "Time-varying formation tracking for second-order multi-agent systems subjected to switching topologies with application to quadrotor formation flying," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 6, pp. 5014–5024, 2016.

[32] J. Hu, P. Bhowmick, and A. Lanzon, "Distributed adaptive time-varying group formation tracking for multiagent systems with multiple leaders on directed graphs," *IEEE Transactions on Control of Network Systems*, vol. 7, no. 1, pp. 140–150, 2019.

[33] C. Wang, H. Tnunay, Z. Zuo, B. Lennox, and Z. Ding, "Fixed-time formation control of multirobot systems: Design and experiments," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 8, pp. 6292–6301, 2018.

[34] K. Wu, J. Hu, Z. Ding, and F. Arvin, "Finite-time fault-tolerant formation control for distributed multi-vehicle networks with bearing measurements," *IEEE Transactions on Automation Science and Engineering*, 2023.

[35] G. Wang, X. Wang, and S. Li, "Node task expansion based finite-time formation-containment control for ground-air collaboration systems," *IEEE Transactions on Network Science and Engineering*, 2024.

[36] M. Noto and H. Sato, "A method for the shortest path search by extended dijkstra algorithm," in *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics.'cybernetics evolving to systems, humans, organizations, and their complex interactions'(cat. no. 0*, vol. 3. IEEE, 2000, pp. 2316–2320.

[37] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[38] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects: Steven m. lavalle, iowa state university, a james j. kuffner, jr., university of tokyo, tokyo, japan," *Algorithmic and computational robotics*, pp. 303–307, 2001.

[39] G. S. Tewolde and W. Sheng, "Robot path integration in manufacturing processes: Genetic algorithm versus ant colony optimization," *IEEE Transactions on systems, man, and cybernetics-Part A: systems and humans*, vol. 38, no. 2, pp. 278–287, 2008.

[40] M. Zhang, Y. Shen, Q. Wang, and Y. Wang, "Dynamic artificial potential field based multi-robot formation control," in *2010 IEEE Instrumentation & Measurement Technology Conference Proceedings*. IEEE, 2010, pp. 1530–1534.

[41] B. Siciliano and L. Sciavicco, "Villani, and oriolo, robotics: Modelling, planning and control," 2008.

[42] Y. Liu and R. Bucknall, "Efficient multi-task allocation and path planning for unmanned surface vehicle in support of ocean operations," *Neurocomputing*, vol. 275, pp. 1550–1566, 2018.

Appendices

A Additional figures, code link, video link

This is the appendix section where we include some additional figures, code, and video links that are relevant to this project due to limited space.

- Formation scaling simulation when narrow corridor detected by leader.

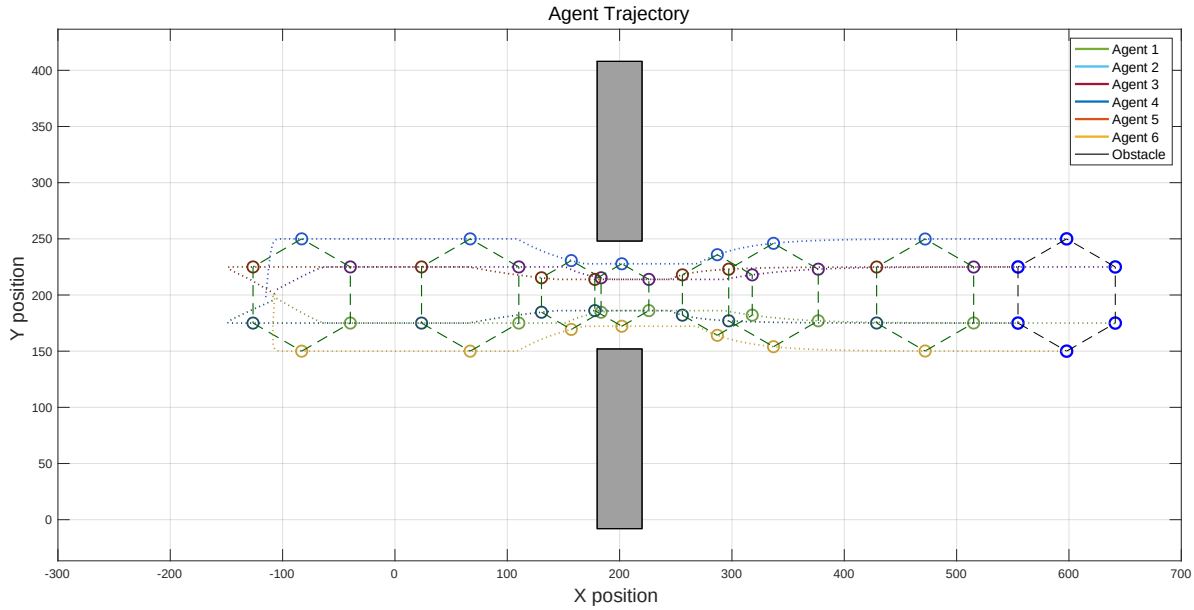
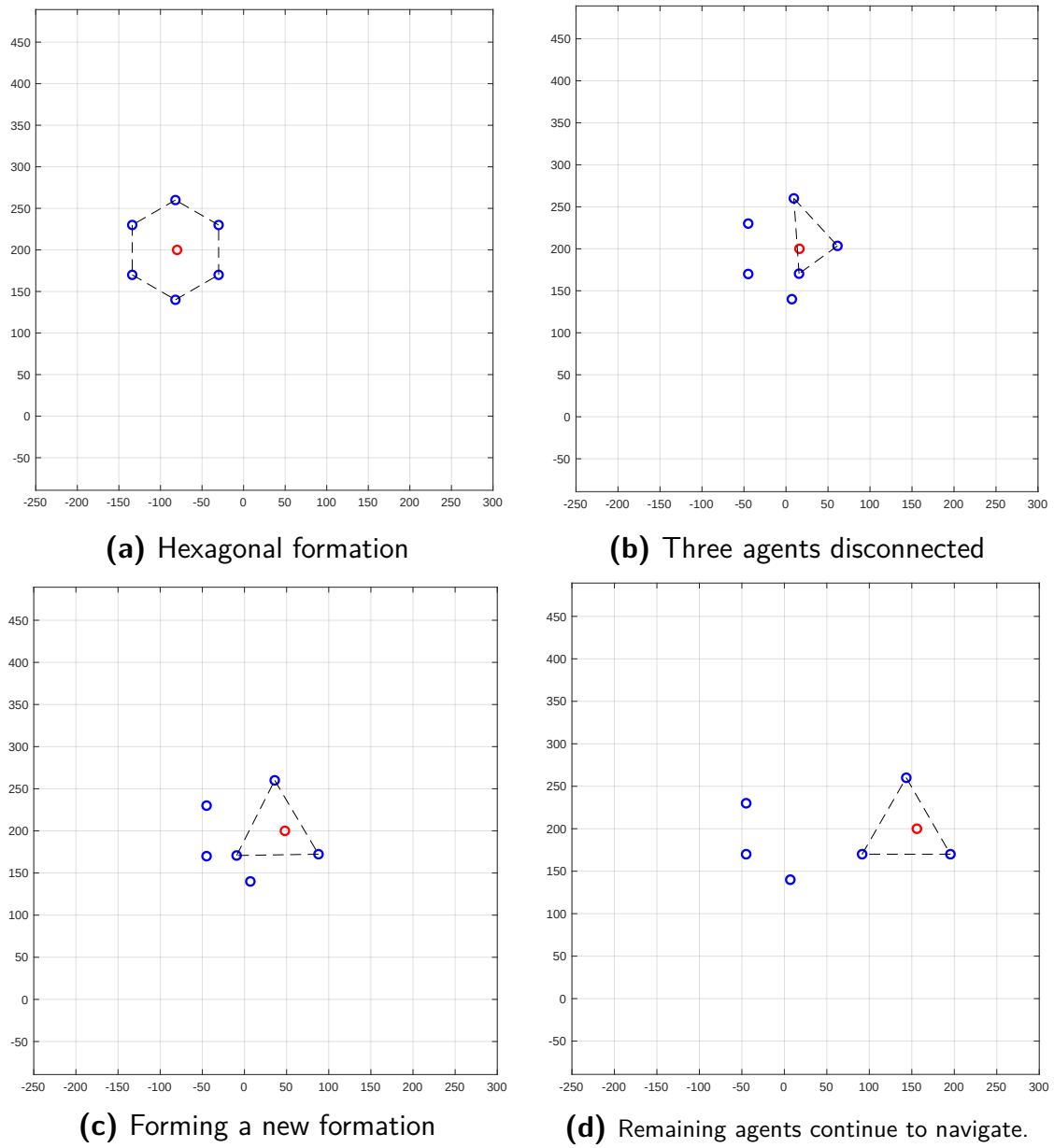


Figure 18: Formation scaling simulation

When the leader detects the presence of a narrow corridor ahead, it computes and adjusts the scaling factor vector s in real time. This ensures that the formation remains intact and passes smoothly through the narrow corridor. In practice, the leader could be equipped with a front-end sensing system to detect the corridor width in real-time. However, in the simulation, the step of measuring the corridor width is skipped. Instead, a scaling factor is preset to facilitate the adjustment process.

- Formation reconfiguration simulation when two agent disconnected.



(a) Hexagonal formation

(b) Three agents disconnected

(c) Forming a new formation

(d) Remaining agents continue to navigate.

Figure 19: Formation reconfiguration simulation

When the leader detects a disconnection among the agents, it dynamically reassign the formation shape vector h_i based on the number of agents remaining connected. This ensures the reassignment of target positions to the remaining agents, allowing them to reform into a new predefined structure and continue their navigation. As the Fig 18 shows, h_{sub3} is aligned.

- Large scaled formation flight with obstacle avoidance

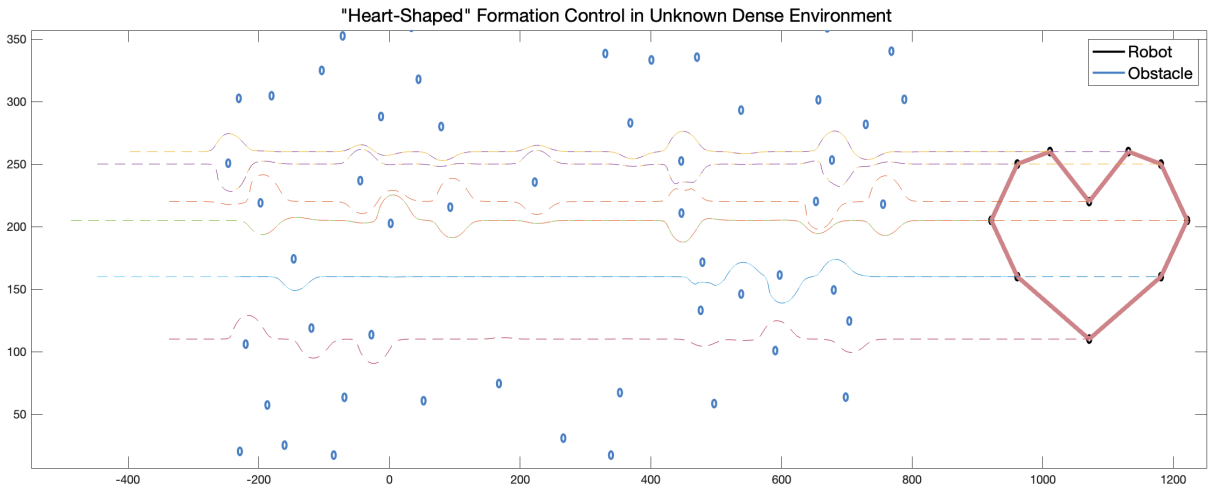


Figure 20: The agent avoids obstacles while maintaining a "heart-shaped formation" as much as possible

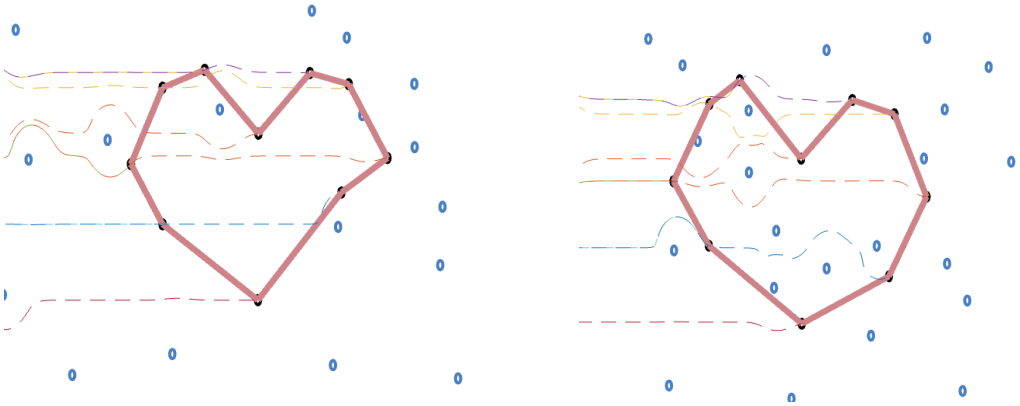


Figure 21: Dramatic changes for the "heart-shaped formation" at two different moment. This is because agents prioritize obstacle avoidance before maintaining formation.

Figure 20 demonstrates a large-scale MAS consisting of 10 agents navigating through an environment abundant with obstacles, under the guidance of equation (9) and equation (27). The dashed lines represent their trajectories. It is noteworthy that our agents prioritize obstacle avoidance before maintaining formation, as shown in Fig 21. Future research should investigate how agents can trade off the trajectories that simultaneously address both obstacle avoidance and formation maintaining requirements, because, agents are repelled to deviate from obstacles using the APF approach, while the formation objective requires agents to track their target positions, which might conflict with the goal of obstacle avoidance.

-MATLAB simulation code: https://github.com/jinyuanzhan

-Relevant simulation videos: https://www.youtube.com/@jinyuanzhang6073