Decentralized Modular Hybrid Supervisory Control for the Formation of Unmanned Helicopters

Ali Karimoddini\(^1\), Mohammad Karimadini\(^2\), Hai Lin\(^3\)

Abstract

This paper presents a collision-free decentralized hybrid supervisory control approach for the Unmanned Aerial Vehicles (UAVs) that are involved in a leader-follower formation mission. Towards this end, a symbolic motion planning scheme is developed to polarly partition the motion space, which results in a finite state discrete model for the motion dynamics of each UAV. Then, a modular discrete supervisor is designed for different components of the formation mission including reaching the formation, keeping the formation, and collision avoidance. While for reaching and keeping the formation, each UAV can satisfy the desired performance independently, for the collision avoidance, a tight cooperation of the UAVs is required. For this purpose, a top-down technique is developed to design the local supervisors decentralizedly, so that the locally supervised distributed agents, as a whole, can cooperatively satisfy the desired specification. The proposed decentralized supervisory algorithm is verified through a hardware-in-the-loop simulator for the formation control of unmanned helicopters.

I. INTRODUCTION

Cooperative control particularly concerns with the interactions and teamwork among agents in order to achieve global specifications in a more efficient way. Taking a cooperative control strategy, it is possible to instruct a group of agents to accomplish a certain mission, collectively. Such a cooperative platform is more robust against failures in team members or in communication

---

\(^1\) A. Karimoddini is with the Department of Electrical and Computer Engineering, North Carolina Agricultural and Technical State University, Greensboro, USA, akarimod@ncat.edu.

\(^2\) M. Karimadini is with the Department of Electrical Engineering, Arak University of Technology, Arak, Iran, karimadini@arakut.ac.ir.

\(^3\) H. Lin is with the Department of Electrical Engineering, University of Notre Dame, Notre Dame, USA, hlin1@nd.edu.
This has motivated aerial industries to deploy cooperative strategies for teams of Unmanned Aerial Vehicles (UAVs) to improve their capabilities and enable them to autonomously involve in cooperative missions such as formation control [4], [5]. Formation control is a typical cooperative task in which several agents move with a relatively fixed distance [6], [7], [8]. This feature can leverage sophisticated functionalities of a team of UAVs to have a more effective performance in missions such as cooperative SLAM, coverage and reconnaissance, and security patrols [9], [10], [11], [12], [13].

For the cooperative formation control of multiple UAV, many studies have been conducted in the literature and different approaches have been developed based on graph theory [14], virtual structure [15], potential fields [16], optimization techniques [17], invariant sets [18], game theory [19], artificial intelligence [20], and behavioral control approaches [21], [22]. However, addressing the iterations between UAVs encounters a major challenge: locally controlling each UAV, while satisfying the global logical specifications for the team. This requires to handle the interplay between the control of UAVs continuous dynamics and supervisory control of their discrete event evolutions. To address these problems, a proper solution is hybrid modelling and control theory [23], [24], which provides a unified framework for mixed event-triggered discrete logic and time-triggered continuous dynamics, with effective tools for mathematical representation and analysis of variety of applications. In our recent study [25], a hybrid supervisory control framework was introduced for the formation control of UAVs. The basic idea is to use polar abstraction of the motion space and utilize the properties of multi-affine functions [26] over the partitioned space. The abstraction techniques [27] can convert the original continuous system with infinite states into a finite state machine for which one can use the well developed theory of supervisory control of discrete event systems (DES) [28]. The method introduced in [25] is applicable to two UAVs involved in a leader-follower scenario; however, more technical considerations are required to extend the method to the multi-follower case. In particular, the UAVs need to tightly cooperate to safely accomplish the assigned mission and avoid the collision.

This paper therefore addresses the hybrid supervisory control of more than two UAVs that are involved in a leader-follower formation scenario. After abstraction of the motion space and achieving a DES model for the motion dynamics of each agent, the formation requirements, as global specifications, are formulated by logical expressions for which we have modularly designed the discrete supervisors for different components of the formation including reaching
the formation, keeping the formation, and collision avoidance. The main challenge here is how to implement the supervisor modules in a decentralized way so that the control decisions be distributed amongst the agents. For this purpose, the designed global supervisor is systematically decomposed to local supervisors through the natural projection into local event sets. In this decentralized structure, each agent proceeds with its sensor readings from its sensors or its neighbors and sends commands to its actuators while synchronizing with neighbors on some shared events in order for common actions. The orchestrating among events is therefore decentralized, based on information exchanged with neighbors, while fulfilling a global task. In this framework the interaction of agents is captured by parallel composition, where the global event set is the union of overlapping local event sets such that each agent has some private events for individual transitions and some shared events for synchronized actions. This paper generalizes the preliminary work in [29], from two followers into an arbitrary finite number of followers, and uses automaton decomposition technique in [30], for supervisor decomposition to obtain the local supervisors for each individual follower UAV. It is furthermore proven that upon the decomposability of the supervisor, the entire closed loop system will satisfy the global task. Therefore, the main contributions of this paper is developing a decentralized hybrid supervisory control approach for multiple UAVs that are involved in a leader-follower formation mission. Furthermore, utilizing the polar partitioning technique, the motion dynamics of the follower helicopters are abstracted to a discrete model and then, a modular supervisor is designed to accomplish different parts of the formation. Then, to implement the supervisor in a decentralized way, it is decomposed into local supervisors, so that the locally supervised distributed agents, as a whole, can cooperatively satisfy the desired specification. Finally, the proposed algorithm is verified through a hardware-in-the-loop simulator.

The rest of this paper is organized as follows. First, the problem of formation control is formulated in Section II. Then, in Section III, using polar abstraction of the motion space, a discrete model is obtained for the motion dynamics of each UAV. In Section IV, a discrete supervisor is modularly designed and then decomposed into local supervisors for decentralized implementation of the control structure. Section V verifies the proposed algorithm through a hardware-in-the-loop simulation platform. Finally, the paper is concluded in Section VI.
II. PROBLEM DESCRIPTION

This paper addresses the problem of a leader-follower formation mission in which having the position and velocity of the leader, follower UAVs should achieve the formation and maintain it during the mission. The proposed approach is a decentralized hybrid supervisory control algorithm, based on the polar abstraction of motion space and the decomposition of the global supervisor. The hardware-in-the-loop simulator platform of the NUS UAV team is then utilized to verify the proposed algorithm. The modelling and low-level control structure of the NUS UAV helicopters are explained in [31], [32] and [33]. For each of these helicopters we have used a hierarchical control structure whose inner-loop controller stabilizes the system using $H_\infty$ control design techniques and its outer-loop is used to drive the UAV towards the desired position (Fig. 1). In [33] it is shown that the inner-loop is fast enough to track the given references, so that the outer loop dynamics can be approximately described as follows:

$$\dot{x} = u, \quad x \in \mathbb{R}^2, \quad u \in U \subseteq \mathbb{R}^2,$$

where $x$ is the position of the UAV; $u$ is the UAV velocity reference generated by the formation algorithm, and $U$ is the convex set of velocity constraints.

Here, we assume that the UAVs are flying at the same altitude, and the velocity of the k’th follower is in the following form:

$$V_{\text{follower}_k} = V_{\text{leader}} + V_{\text{rel}_k},$$
where the k’th follower should reach the desired position with respect to the leader and maintain it by controlling the relative velocity, \( V_{relk} \). Alternatively, one can consider a relatively fixed frame for each follower UAV, in which the follower moves with the relative velocity, \( V_{relk} \).

Now, the formation problem can be expressed as follows:

**Problem 1**: Given the dynamics of the follower UAVs as (1) and their velocity in the form of (2), design the formation controller to generate the relative velocity of the followers, \( V_{relk} \), such that starting from any initial state inside the control horizon, the follower UAVs eventually reach the desired position, while avoiding the collision with other follower UAVs. Moreover, after reaching the formation, the follower UAVs should remain at the desired position.

To address this problem, we propose a decentralized modular hybrid supervisor to achieve three major goals: reaching the formation, keeping the formation, and collision avoidance. For this purpose, we first find a discrete model for each of the follower UAVs based on the polar abstraction of the motion space [25], and then will design discrete supervisors for reaching the formation, keeping the formation, and collision avoidance, modularly as described in the following sections.

### III. Discrete Model of the UAV Motion Dynamics Over the Polar Partitioned Space

Abstraction is a technique which reduces the system’s complexity and provides a computational effective tool for the analysis of the system [27]. In this paper, an abstraction technique will be used based on the polar partitioning of the motion space [25]. Having the velocity of k’th follower in the form of (2), one can consider a relatively fixed frame, in which the k’th follower moves over a partitioned circle with the radius of \( R_m \) that is centered at the desired position of this follower. With the aid of the partitioning curves \( \{ r_i = \frac{R_m}{n_r-1} (i-1), i = 1, ..., n_r \} \) and \( \{ \theta_j = \frac{2\pi}{n_\theta-1} (j-1), j = 1, ..., n_\theta \} \), this circle can be partitioned into \((n_r-1)(n_\theta-1)\) partitioning elements. In this partitioned space, an element \( R_{i,j} = \{ p = (r, \theta) | r_i \leq r \leq r_{i+1}, \theta_j \leq \theta \leq \theta_{j+1} \} \), has four vertices, \( v_0, v_1, v_2, v_3 \) (Fig. 2(a)), four edges, \( E^+_i, E^-_i, E^+_\theta, E^-_\theta \) (Fig. 2(b)), and correspondingly, four outer normal vectors \( n^+_i, n^-_i, n^+_\theta, n^-_\theta \) (Fig. 2(c)). The set \( V(\ast) \) stands for the vertices that belong to \( \ast \) (\( \ast \) can be an edge, or a region \( R_{i,j} \)).

In [25], it is shown that for a system with a multi-affine dynamics \( \dot{x} = h(x, u(x)) \), the region \( R_{i,j} \) can be an invariant region (the trajectories of the system remain inside the region for ever) if
there exist control values $u(v_m)$ at the vertices $v_m$ so that $n^s_q \cdot h(v_m, u(v_m)) < 0$, $m = 0, 1, 2, 3$, where $n^s_q$, with $q \in \{r, \theta\}$ and $s \in \{+, -\}$, are the outer normal vectors of the edges incident with $v_m$. Also, it is possible to design a controller to drive the system’s trajectory to exit form the edge $E^s_q$, $q \in \{r, \theta\}$ and $s \in \{+, -\}$, by choosing the control values $u(v_m)$ at the vertices so that the following conditions hold:

1) $n^{s'}_q \cdot h(v_m, u(v_m)) < 0$ for all $E^{s'}_q \neq E^s_q$ and $v_m \in V(E^{s'}_q)$,

2) $n^{sT}_q \cdot h(v_m, u(v_m)) > 0$ for all $v_m$, $m = 0, 1, 2, 3$,

where the first condition guarantees that the system’s trajectories do not leave the region through non-exit edges $E^{s'}_q \neq E^s_q$, while the second condition ensures us that the trajectories leave the region through the exit edge $E^s_q$. According to the property of multi-affine systems, the control value at any point inside the region can be achieved based on the control values at the vertices as $u(x) = \sum_{m=0}^3 \lambda_m(x)u(v_m)$, where $\lambda_m$, $m = 0, 1, 2, 3$, are the control coefficients with $\lambda_m \geq 0$ and $\sum_{m=0}^3 \lambda_m = 1$. We denote the controller for having a region invariant by $C_{0k}$. We also denote $C^+_{rk}$, $C^-_{rk}$, $C^+_{\theta k}$, and $C^-_{\theta k}$ as the controllers for having the edges $F^+_r$, $F^-_r$, $F^+_\theta$, and $F^-_\theta$, as the exit edges, respectively.

Now, this model of the UAV motion over the partitioned space can be abstracted to a finite state machine which simulates the original system and contains all of its behavior. This finite state model of the motion dynamics can be represented by a discrete automaton, formally defined as follows:

**Definition 1:** (Automaton)[34]. A deterministic automaton is a tuple $A := (Q, q_0, E, \delta, Q_m)$ consisting of a set of states $Q$, an initial state $q_0 \in Q$, a set of events $E$ that causes transitions between the states, and a transition relation $\delta \subseteq Q \times E \times Q$ (with a partial map $\delta : Q \times E \rightarrow Q$), such that $(q, e, q') \in \delta$ if and only if state $q$ transits to state $q'$ by event $e$, denoted by $q \xrightarrow{e} q'$ (or
\[ \delta(q, e) = q' \]. \( Q_m \subseteq Q \) represents the marked (accepting or final) states to assign a meaning of accomplishment to some states. For an automaton whose all states are marked, \( Q_m \) is omitted from the tuple.

For \( \text{UAV}_1 \), the discrete model of the system over the partitioned space can be described by the automaton \( A_1 = (Q_1, q_0, E_1, \delta_1, Q_{m_1}) \), where \( Q_1 = \{R_1, O_1\} \) is the set of discrete states, and \( E_1 = C_1 \cup \{C_{01}\} \cup D_1 \cup E_x \) is the event set, where \( C_1 = \{C_{r_1}, C_{r_1}, C_{\theta_1}, C_{\theta_1}\} \) and \( D_1 = \{d_{i,j} \mid 1 \leq i \leq n_r - 1, 1 \leq j \leq n_{\theta} - 1\} \). When \( \text{UAV}_1 \) is in one of the regions \( R_{i,j} \), in the abstract model it is considered to be in the discrete state \( R_1 \). Then, one of the actuation commands belong to \( C_1 \) drives the UAV to one of its adjacent regions. In this case, right after issuing the actuation commands, the system transits to the detection state \( O_1 \) and waits until the UAV enters a new region. Crossing boundaries of the new region, a detection event \( d_{i,j} \) belonging to \( D_1 \) will be generated which shows the UAV enters the new region \( R_{i,j} \). The command \( C_{01} \), keeps the UAV in the current region and does not change the discrete state of the system. We use the notation \( D_{M1} = \{d_{i,j} \mid 1 \leq i = 1, 1 \leq j \leq n_{\theta} - 1\} \) to denote the detection events which show entering a region in the first circle, and \( d_{1} = D_1 - D_{M1} = \{d_{i,j} \mid 1 \leq i < n_r - 1, 1 \leq j \leq n_{\theta} - 1\} \) for the rest of detection events. Here, \( E_x = CA_1 \cup \mathcal{R}_1 \cup \{\text{Stop}_1, \text{Stop}_2, \ldots, \text{Stop}_n\} \) is the set of external events which are required for the collision avoidance and do not change the state of the system. The events belong to \( CA_1 = \bigcup_{i=1}^{n} (CA_{i1} \cup CA_{i1}) \) show the collision alarms, where the events in \( CA_{ij} = \{ca_{ij,F}, ca_{ij,N}\} \) show that \( \text{UAV}_j \) enters the alarm zone of \( \text{UAV}_i \). The detail will be discussed in Section IV-B when the collision avoidance mechanism is described. The events \( \text{Stop}_k \) is the command that request \( \text{UAV}_k \) to stop at its current position in the relative frame. The event set \( \mathcal{R}_1 = \{\mathcal{R}_{21}, \mathcal{R}_{12}, \mathcal{R}_{31}, \mathcal{R}_{13}, \ldots, \mathcal{R}_{n1}, \mathcal{R}_{1n}\} \) is used to release the stopped UAVs in which \( \mathcal{R}_{ij} \) is a command that \( \text{UAV}_i \) sends to release \( \text{UAV}_j \) which has been previously requested to stop. For this discrete model of \( \text{UAV}_1 \), the discrete state \( R_1 \) is considered both the initial and mark states. The graph representation of the discrete model of \( \text{UAV}_1 \) is shown in Fig. 3. In this graphs, the arrows starting from one state and ending to another state represent the transitions, labeled by the events belong to the event set, \( E_1 \). The entering arrows stands for the initial state. The marked state is shown by double circles. The DES model of \( \text{UAV}_k, k = 2, \ldots, n, \) can be constructed in a similar way.

In the discrete model of the plant \( A = (Q, q_0, E, \delta, Q_m) \), the definition of the transition relation can be extended from the domain of \( Q \times E \) into the domain of \( Q \times E^* \) to define transitions
over the strings \( s \in E^* \), where \( E^* \) stands for the Kleene Closure of \( E \) (the collection of all finite sequences of events over elements of \( E \)).

**Definition 2:** (Transition on strings) For a deterministic automaton, the existence of a transition over a string \( s \in E^* \) from a state \( q \in Q \) is denoted by \( \delta(q, s) \). and inductively defined as \( \delta(q, \varepsilon) = q \), and \( \delta(q, se) = \delta(\delta(q, s), e) \) for \( s \in E^* \) and \( e \in E \).

The existence of a set \( K \subseteq E^* \) of strings from a state \( q \in Q \) is then denoted as \( \delta(q, K) \) and read as \( \forall s \in K : \delta(q, s) \). The language generated by the automaton \( A \) is defined as \( L(A) = \{ s \in E^* | \delta(q_0, s) \} \). The set of all substrings of a string \( s \) is called the prefix-closure of \( s \), denoted by \( \overline{s} \). Prefix-closure of a language is defined as the set of prefix-closure of all of its strings. The marked language, \( L_m(A) \), is the set of strings that belong to \( L(A) \) and end with the marked states and can be formally defined as \( L_m(A) := \{ s | s \in L(A) \land \delta(q_0, s) \in Q_m \} \). Recall that the transition relation is a partial relation, and in general some of the states might not be accessible from the initial state.

In the DES model of \( UAV_k \), the event set \( E_k \) consists of the controllable event set \( E_{ck} = \{ C_{0k}, C_{rk}^+, C_{rk}^-, C_{\theta k}^+, C_{\theta k}^- \} \cup \{ Stop_1, \ldots, Stop_n, \mathcal{R}_{12}, \mathcal{R}_{21}, \ldots, \mathcal{R}_{1n}, \mathcal{R}_{n1} \} \) and the uncontrollable event set \( E_{uc} = \{ d_{i,j,k} | 1 \leq i \leq n_r - 1, 1 \leq j \leq n_{\theta} - 1 \} \). The uncontrollable events are those that cannot be affected by the supervisor. A language \( K \) is controllable with respect to the language \( L(A) \) and the event set \( E_{uc} \) if and only if \( \forall s \in K \) and \( \sigma \in E_{uc} \), if \( s\sigma \in L(A) \), then \( s\sigma \in K \). In words, \( K \) is controllable if uncontrollable events need not to be disabled. Indeed, the controllability is the existence condition of a supervisor for the control goal described by the specification \( K \) [34]. Next, we will explain how to design a supervisor for the formation mission by controlling the plant over the controllable set of events.
IV. Designing a Decentralized Modular Supervisor for the Formation Control of UAVs

Given the discrete model of follower UAVs over the partitioned space, it is possible to design the supervisor to achieve a desired order of events to accomplish the formation. Indeed, the supervisor, $S$, observes the executed strings of the plant $A$ and disables the undesirable controllable events. Here, we assume that all of the events are observable. The generated language and marked language of the closed-loop system, $L(S/A)$ and $L_m(S/A)$, can be constructed as follows:

1. $\varepsilon \in L(S/A)$
2. $[(s \in L(S/A)) \text{ and } (s\sigma \in L(A)) \text{ and } (\sigma \in L(S))] \Leftrightarrow (s\sigma \in L(S/A))$
3. $L_m(S/A) = L(S/A) \cap L_m(A)$

where $s$ is the string that has been generated so far by the plant $A$, and $\sigma$ is an event, which the supervisor $S$ should decide whether keep it active or not in the supervised system $S/A$.

Within this framework one can use parallel composition to facilitate the control synthesis. Parallel composition is a binary operation between two automata which can be defined as follows:

**Definition 3:** (Parallel Composition [34]) Consider two automata $A_i = (Q_i, q^0_i, E_i, \delta_i, Q_{m_i}), i = 1, 2$. The parallel composition (synchronous composition) of $A_1$ and $A_2$ is the automaton $A_1||A_2 = (Q = Q_1 \times Q_2, q_0 = (q^0_1, q^0_2), E = E_1 \cup E_2, \delta, Q_m = Q_{m_1} \times Q_{m_2})$, with $\delta$ defined as follows:

\[
\delta((q_1, q_2), e) = \begin{cases} 
(\delta_1(q_1, e), \delta_2(q_2, e)), & \text{if } \delta_1(q_1, e)! \text{ and } \delta_2(q_2, e)!, e \in E_1 \cap E_2; \\
(\delta_1(q_1, e), q_2), & \text{if } \delta_1(q_1, e)! \text{ and } e \in E_1 \setminus E_2; \\
(q_1, \delta_2(q_2, e)), & \text{if } \delta_2(q_2, e)! \text{ and } e \in E_2 \setminus E_1; \\
\text{undefined}, & \text{otherwise.}
\end{cases}
\]

Here, the parallel composition is used to combine the plant’s discrete model and the supervisor as follows:

**Lemma 1:** [35] Let $A = (Q, q_0, E, \alpha, Q_m)$, be the plant automaton and $K \subseteq E^*$ be the desired marked language. There exists a nonblocking supervisor $S$ such that $L_m(S/A) = L_m(S||A) = K$ if $\emptyset \neq K = \bar{K} \cap L_m(A)$ and $K$ is controllable. In this case, $S$ could be any automaton with $L(S) = L_m(S) = \bar{K}$.

Now, using the above lemma, it is possible to design the supervisor for the formation problem
described in Problem 1. In fact, for the discrete model of the UAVs, Problem 1, can be represented as follows:

**Restatement of Problem 1:** Given the discrete model of the follower UAVs over the partitioned space by the automata $A_k$, $k = 1, \ldots, n$, design the formation supervisor such that the follower UAVs eventually reach the formation by visiting one of the sectors in the first circle $R_{1,j}$ of their partitioned motion space and remain there for ever. Meanwhile, the UAVs should avoid collision so that no pair of UAVs visit the same region at the same time.

This problem indeed evokes two specifications: 1- Reaching and keeping the formation and 2- Avoiding collision. To simplify the design, it is possible to modularly design the supervisors to achieve these two specifications. Next lemma describes how to design the supervisors in a modular way.

**Lemma 2:** [35] Let $A = (Q, q_0, E, \alpha, Q_m)$ be the plant automaton and the prefix-closed controllable languages $K_1, K_2 \subseteq E^\ast$ be the desired marked specifications. Suppose there exist nonblocking supervisors $S_1$ and $S_2$ such that $L_m(S_1/A) = L_m(S_1\|A) = K_1$ and $L_m(S_2/A) = L_m(S_2\|A) = K_2$, then $S = S_1\|S_2$ is a nonblocking supervisor with $L(S\|A) = K_1 \cap K_2$. ■

Using this lemma, we will then modularly design the supervisors for the reaching the formation and collision avoidance.

A. **Designing the supervisor for reaching and keeping the formation**

For reaching the formation, it is sufficient to drive each of the follower UAVs directly towards one of the regions $R_{1,j}$, $1 \leq j \leq n_\theta - 1$, located in the first circle in their corresponding partitioned motion space. After reaching $R_{1,j}$, the UAVs should remain inside it, to keep the formation. The specifications $K_{Fk}$ for reaching and keeping the specification for $UAV_k$ are realized in Fig. 4. When the k'th follower UAV is not in the first circle, the command $C_{r_k}$ will be generated to push the UAV towards the origin. Entering a new region, one of the events from $d_k = \{d_{i,jk} | 1 < i \leq n_r - 1, 1 \leq j \leq n_\theta - 1\}$ will appear. This will continue until one of the events from $D_{Mk} = \{d_{i,jk} | i = 1, 1 \leq j \leq n_\theta - 1\}$ is generated, which shows that the formation is reached. In this case, the event $C_{0k}$ is activated, which keeps the system trajectory inside the first region. If a collision alarm happens to $UAV_k$, the formation supervisor does not change the generable language after the events belonging to $CA$, and lets the collision avoidance supervisor handle it until the collision be avoided and the UAV be released to resume the formation task.
It can be seen that $K_{F_k}$ is controllable with respect to the plant language $L(A_k)$ and the event set $E_{uck}$, as they do not disable any uncontrollable event. Therefore, based on Lemma 1, there exist supervisors that can control the plants $A_k$ to achieve this specifications. The supervisor is the realization of the above specification in which all states are marked. Marking all states of the supervisors allows the closed-loop marked states to be solely determined by the plants’ marked states. The supervisor for reaching the formation and keeping the formation of $UAV_k$ is denoted by $A_{F_k}$.

Fig. 4. The specification for reaching and keeping the formation for $UAV_k$.

B. Designing the supervisor for collision avoidance

When $UAV_i$ is going to reach its desired position, in some situations, another follower, $UAV_j$, may enter the alarm zone of $UAV_i$ (Fig. 5) which requires these UAVs to cooperatively avoid the collision. For this purpose, first, $UAV_i$ asks $UAV_j$ to stop in the relative frame and then, $UAV_i$ finds a path to safely get away from $UAV_j$. After avoiding the collision, $UAV_i$ releases $UAV_j$ and both UAVs resume their normal operation for reaching the formation. More specifically, to avoid collision, by command $Stop_j$, $UAV_i$ informs $UAV_j$ to stop for a while to safely manage the situation. The event $ca_{ij,F}$ shows that $UAV_j$ is in front of the path of $UAV_i$ towards its destination. In this case, to avoid the collision it is sufficient that $UAV_i$ turns anticlockwise to change its azimuth angle, $\theta$, by activating the command $C_{\theta_k}^+$. Also, the event $ca_{ij,N}$ shows that
$UAV_j$ has entered the alarm zone of $UAV_i$ but it is not in front of the path of $UAV_i$ towards its destination and hence, to avoid the collision it is sufficient that $UAV_i$ moves forward by activating the command $C_{r_k}^{-}$. This will continue until removing the collision alarm. Then, $UAV_i$ releases $UAV_j$, and reaching the formation can be resumed by reaching formation supervisor, $A_{F_k}$, which was explained in the previous section. Meanwhile, if $UAV_i$ enters one of the regions in the first circle, one of the events belong to $D_{Mi}$ appears which means that $UAV_i$ has reached its desired formation and should remain there for the rest of the mission. If neither of collision avoidance alarms happens, then $UAV_i$ and $UAV_j$ can do their normal operations by independent enabling of events $C_i$ and $C_j$ followed by the detection signals $D_i$ and $D_j$ in any order. In this case, the other module, the reaching formation supervisor, will manage this situation. Fig. 6 shows this specification, $K_C$, for two follower UAVs, in which the left side shows that after appearing one of the events $ca_{12F}$ or $ca_{12N}$, $UAV_i$ realizes that $UAV_2$ has entered its alarm zone. Similarly, the right side of Fig. 6, shows the collision avoidance mechanism when $UAV_1$ enters the alarm zone of $UAV_2$.

![Diagram](image_url)

Fig. 5. $UAV_j$ enters the alarm zone of $UAV_i$.

It can be verified that $K_C$ is controllable with respect to the language $\bigcup_{k=1}^{n} A_k$ and the event set $\bigcup_{k=1}^{n} E_{uc_k}$. Therefore, based on Lemma 1, there exists a supervisor $A_C$ that can control the plants $A_k$, $k = 1, ..., n$, to achieve this joint specification. The supervisor is the realization of the specification $K_C$ in which all states are marked.
C. Decomposed local supervisors for the collision avoidance

The collision avoidance supervisor, $A_C$, is a centralized supervisor which manages $UAV_k$, $k = 1, \ldots, n$. To make this supervisor decentralized and to achieve local supervisors, we will utilize our proposed decomposition scheme introduced in [30] and [36]. Here, local supervisors can be achieved by the projection of the global supervisor to each agent’s local event set. The projection of the global supervisor $A_C$ to the event set of $UAV_k$, $E_k$, is denoted by $P_{E_k}(A_C)$, and can be obtained by replacing the events that belong to $E \setminus E_k$ by $\varepsilon$-moves, and then, merging the $\varepsilon$-related states. The $\varepsilon$-related states form equivalent classes can be defined as follows.

**Definition 4:** (Equivalent class of states, [37]) Consider an automaton $A_C = (Q, q_0, E, \delta)$ and an event set $E' \subseteq E$. Then, the relation $\sim_{E'}$ is the minimal equivalence relation on the set $Q$ of states such that $q' \in \delta(q, e) \land e \notin E' \Rightarrow q \sim_{E'} q'$, and $[q]_{E'}$ denotes the equivalence class of $q$ defined on $\sim_{E'}$.

The set of equivalent classes of states over $\sim_{E'}$, is denoted by $Q/\sim_{E'}$ and defined as $Q/\sim_{E'} = \{[q]_{E'}|q \in Q\}$. Note that $\sim_{E'}$ is an equivalence relation as it is reflexive ($q \sim_{E'} q$), symmetric ($q \sim_{E'} q' \Leftrightarrow q' \sim_{E'} q$) and transitive ($q \sim_{E'} q' \land q' \sim_{E'} q'' \Rightarrow q \sim_{E'} q''$).

It should be noted that the relation $\sim_{E'}$ can be defined for any $E' \subseteq E$, for example, $\sim_{E_i}$ and $\sim_{E_i \cup E_j}$, respectively denote the equivalence relations with respect to $E_i$ and $E_i \cup E_j$. Moreover, when it is clear from the context, $\sim_i$ is used to denote $\sim_{E_i}$, for simplicity.

The natural projection is formally defined on the strings as

**Definition 5:** (Natural Projection on Strings, [34]) Consider a global event set $E$ and an event set $E' \subseteq E$. Then, the natural projection $p_{E'} : E^* \rightarrow E'^*$ is inductively defined as $p_{E'}(\varepsilon) = \varepsilon$, and $\forall s \in E^*, e \in E : p_{E'}(se) = \begin{cases} p_{E'}(s)e & \text{if } e \in E'; \\ p_{E'}(s) & \text{otherwise.} \end{cases}$

The natural projection is then formally defined on an automaton as follows.

**Definition 6:** (Natural Projection on Automaton) Consider a deterministic automaton $A_C = (Q, q_0, E, \delta)$ and an event set $E' \subseteq E$. Then, $P_{E'}(A_C) = (Q_i = Q/\sim_{E'}, [q_0]_{E'}, E', \delta')$, with $[q']_{E'} \in \delta'([q]_{E'}, e)$ if there exist states $q_1$ and $q_1'$ such that $q_1 \sim_{E'} q, q_1' \sim_{E'} q'$, and $\delta(q_1, e) = q_1'$. Again, $P_{E'}(A_C)$ can be defined into any event set $E' \subseteq E$. For example, $P_{E_i}(A_C)$ and $P_{E_i \cup E_j}(A_C)$, respectively denote the natural projections of $A_C$ into $E_i$ and $E_i \cup E_j$. When it is clear from the context, $P_{E_i}$ is replaced with $P$, for simplicity.
Once the local supervisor automata are derived through the natural projection, the decentralized supervisor is then obtained using the parallel composition of local supervisor automata. Parallel composition captures the logical behavior of concurrent distributed systems by allowing each subsystem to evolve individually on its private events, while synchronize with its neighbors on shared events for cooperative tasks.

The parallel composition of $A_i, i = 1, 2, ..., n$ (for example local plants or local supervisors) is called a parallel distributed system, and is defined based on the associativity property of parallel composition [34] as

$$n \parallel_{i=1} A_i := A_1 \parallel ... \parallel A_n := (A_{n-1} \parallel (\cdots \parallel (A_2 \parallel A_1))).$$

The obtained decentralized supervisor is then compared with the original global supervisor automaton using the bisimulation relation, in order to ensure that they, as a team of agents, can resemble the global monolithic supervisor, collectively. Bisimulation-based decomposability preserves the nondeterminism and branching properties that might appear in the decentralized supervisor, even for deterministic global supervisor automata.

**Definition 7:** (Bisimulation [34]) Consider two automata $A_i = (Q_i, q_0^i, E_i, \delta_i), i = 1, 2$. The automaton $A_1$ is said to be similar to $A_2$ (or $A_2$ simulates $A_1$), denoted by $A_1 \prec A_2$, if there exists a relation $R$ from $A_1$ to $A_2$ over $Q_1 \times Q_2$ and with respect to $E_i$, such that (1) $(q_0^1, q_0^2) \in R$, and (2) $\forall (q_1, q_2) \in R, q_1' \in \delta_1(q_1, e), \exists q_2' \in Q_2$ such that $q_2' \in \delta_2(q_2, e), (q_1', q_2') \in R$.

Automata $A_1$ and $A_2$ are said to be bisimilar (bisimulate each other), denoted by $A_1 \cong A_2$ if $A_1 \prec A_2$ with a simulation relation $R_1$, $A_2 \prec A_1$ with a simulation relation $R_2$ and $R_1^{-1} = R_2$, where $R_1^{-1} = \{(y, x) \in Q_2 \times Q_1 | (x, y) \in R_1\}$.

Based on these definitions we may now formally define the decomposability of a supervisor automaton with respect to parallel composition and natural projections as follows.

**Definition 8:** (Supervisor automaton decomposability) A supervisor automaton $A_C$ with the event set $E$ and local event sets $E_i, i = 1, ..., n, E = \bigcup_{i=1}^{n} E_i$, is said to be decomposable with respect to parallel composition and natural projections $P_i, i = 1, \cdots, n$, when $\parallel_{i=1}^{n} P_i(A_C) \cong A_C$.

Following result represents the decomposability conditions with respect to arbitrary finite number of agents.

**Lemma 3:** (Theorem 1 in [36]) A deterministic automaton $A_C = (Q, q_0, E = \bigcup_{i=1}^{n} E_i, \delta)$ is decomposable with respect to parallel composition and natural projections $P_i, i = 1, ..., n$, such that $A_C \cong \parallel_{i=1}^{n} P_i(A_C)$ if and only if
suppose there exists a deterministic global controller automaton and let the global specification is given by a task automaton $A$ over a parallel distributed system for the satisfaction of global task by the team. The global specification in a decentralized architecture, decomposable, then whether the designed supervisor can be implemented in a decentralized way.

Problem 2: (Decentralized cooperative control problem) Consider a plant, represented by a parallel distributed system $A_P := \bigparallel_{i=1}^{n} A_{P_i}$, with local event sets $E_i$, $i = 1,...,n$, and let the global specification is given by a deterministic task automaton $A_S$ over $E = \bigcup_{i=1}^{n} E_i$. Furthermore, suppose there exists a decomposable deterministic global controller automaton $A_C \cong \bigparallel_{i=1}^{n} P_i(A_C)$, so that $L_{m}(A_P \parallel A_C) \cong L_{m}(A_S)$. Then, whether the local controllers can lead the team to satisfy the global specification in a decentralized architecture, $L_{m}(\bigparallel_{i=1}^{n} (A_{P_i} \parallel P_i(A_C))) \cong L_{m}(A_S)$.

Following result considers a team of $n$ local plants and introduces the supervisor decomposability and satisfaction of the global task by each local supervised plant as a sufficient condition for the satisfaction of global task by the team.

Theorem 1: (Decentralized cooperative control using supervisor decomposition) Consider a plant, represented by a parallel distributed system $\bigparallel_{i=1}^{n} A_{P_i}$, with local event sets $E_i$, $i = 1,...,n$, and let the global specification is given by a task automaton $A_S$ over $E = \bigcup_{i=1}^{n} E_i$. Furthermore, suppose there exists a deterministic global controller automaton $A_C \cong \bigparallel_{i=1}^{n} P_i(A_C)$, so that $L_{m}(A_C/A_P) = L_{m}(A_C \parallel A_P) \cong L_{m}(A_S)$. Then, the decentralized closed loop system satisfies
the global specification, in the sense of bisimilarity, i.e., \( L_m(\bigvee_{i=1}^n (A_{P_i} \parallel P_i(A_C))) \cong L_m(A_S) \), provided the decomposability conditions for \( A_C \).

**Proof:** Following lemma is used for the proof of Theorem 1.

**Lemma 4:** (Associativity of parallel composition [34]) \( A_1 \parallel A_2 \parallel \cdots \parallel A_{n-1} \parallel A_n \cong A_n \parallel (A_{n-1} \parallel (\cdots (A_2 \parallel A_1)) \).

Now, Theorem 1 is proven as follows. Satisfying the decomposability conditions given in Lemma 3, we can decompose \( A_C \) into local supervisor automata \( P_i(A_C) \), \( i = 1, \ldots, n \), such that \( A_C \cong \bigvee_{i=1}^n P_i(A_C) \). Then, applying local controllers \( P_i(A_C) \) to local plants in a decentralized configuration \( \bigvee_{i=1}^n (A_{P_i} \parallel P_i(A_C)) \) leads the entire system to satisfy the global specification as \( \bigvee_{i=1}^n (A_{P_i} \parallel P_i(A_C)) \cong (\bigvee_{i=1}^n (A_{C_i})) \parallel (\bigvee_{i=1}^n (A_{P_i})) \cong A_C \parallel A_P \), where the first bisimilarity comes from Lemma 4, and the second bisimilarity is due to the decomposability of \( A_C \) and the definitions of parallel distributed plant. On the other hand, based on the the premise of Theorem 1 and form the design of the global supervisor we have \( L_m(A_C \parallel A_P) \cong L_m(A_S) \) which leads to \( L_m(\bigvee_{i=1}^n (A_{P_i} \parallel P_i(A_C))) \cong L_m(A_S) \). ■

The significance of this result is the decentralized implementation of the global supervisor, \( A_C \). Fig. 6 shows the realization of the collision avoidance specification, \( K_C \), for two UAVs. This collision avoidance mechanism was explained in Section IV-B. The collision avoidance supervisor \( A_C \), is the realization of the specification \( K_C \) in which all states are marked. Based on Theorem 1, it is possible to implement \( A_C \) in a decentralized way, subjected the decomposability of \( A_C \), described in Lemma 3. For two agents, the decomposability conditions in Lemma 3, can be reduced to the following conditions:

**Lemma 5:** (Theorem 4 in [38]) A deterministic automaton \( A_C = (Q, q_0, E = E_1 \cup E_2, \delta) \) is decomposable with respect to parallel composition and natural projections \( P_i, i = 1, 2 \), such that \( A \cong P_1(A_C) || P_2(A_C) \) if and only if \( A_C \) satisfies the following decomposability conditions (DC):

\( \forall e_1 \in E_1 \setminus E_2, e_2 \in E_2 \setminus E_1, q \in Q, s \in E^* \),

- **DC1:** \([\delta(q, e_1)! \land \delta(q, e_2)!] \Rightarrow [\delta(q, e_1 e_2)! \land \delta(q, e_2 e_1)!];
- **DC2:** \([\delta(q, e_1 e_2)! \Rightarrow \delta(q, e_2 e_1)!];
- **DC3:** \( \forall s, s' \in E^*, s \neq s', p_{E_1 \cap E_2}(s), p_{E_1 \cap E_2}(s') \) start with the same common event \( a \in E_1 \cap E_2, q \in Q; \delta(q, s)! \land \delta(q, s')! \Rightarrow \delta(q, p_1(s)[p_2(s')! \land \delta(q, p_1(s')|p_2(s))!;
- **DC4:** \( \forall i \in \{1, 2\}, x, x_1, x_2 \in Q_i, x_1 \neq x_2, e \in E_i, t \in E_i^*, x_1 \in \delta_i(x, e), x_2 \in \delta_i(x, e);
\[ \delta_i(x_1, t) \Leftrightarrow \delta_i(x_2, t) \].

Here, the decomposability conditions \( DC_1 \) and \( DC_2 \) respectively guarantee that any decision on the selection or order of two transitions can be done by the team of agent, while conditions \( DC_3 \) and \( DC_4 \) respectively ensure that the interaction of local automata \( P_1(A_C) \) and \( P_2(A_C) \) neither allows an illegal string that is not in \( A_C \), nor stops a legal string of \( A_C \).

As it can be seen in \( A_C \), the successive and adjacent events from pairs of private event sets (from different local event sets) \((C_{01}, C_2), (C_{01}, D_2), (C_{02}, C_1), (C_{02}, D_1), (C_1, D_2), (C_2, D_2)\), appear in both orders in the global supervisor automaton. Therefore, \( DC_1 \) and \( DC_2 \) are satisfied. Moreover, among common events \( \mathcal{R}_{12}, \mathcal{R}_{21} \), \( CA_{12} = \{ca_{12F}, ca_{12N}\} \), \( CA_{21} = \{ca_{21F}, ca_{21N}\} \), \( Stop_1 \), and \( Stop_2 \), the events \( \mathcal{R}_{12}, \mathcal{R}_{21}, Stop_1, \) and \( Stop_2 \) are not shared between different
Strings just share the events $CA_1$, $CA_2$, where the corresponding local strings do not interleave on these events because of predecessor common events before $CA_1$, $CA_2$. Therefore, $DC3$ also is fulfilled. Finally, $DC4$ is satisfied because of the determinism of local automata $P_1(A_C)$ and $P_2(A_C)$, and hence, the supervisor automaton $A_C$ is decomposable into $A_{C1} = P_1(A_C)$ and $A_{C2} = P_2(A_C)$, shown in Fig. 7, so that $A_{C1} \parallel A_{C2} \cong A_C$.

V. VERIFYING THE ALGORITHM THROUGH A HARDWARE-IN-THE-LOOP SIMULATION PLATFORM

To verify the proposed algorithm, we have used a hardware-in-the-loop simulation platform [39] developed for NUS UAV helicopters [31]. In this platform, the nonlinear dynamics of the UAVs have been replaced with their nonlinear model, and all software and hardware components that are involved in a real flight test remain active during the simulation so that the simulation results achieved from this simulator are very close to the actual flight tests. This multi-UAV simulator test bed is used to verify the proposed algorithm. For this purpose, consider two followers that should track a leader UAV with a desired distance, as shown in Fig. 8. The distance between the desired position of the $Follower_1$ and $Follower_2$ and the leader UAV are $\Delta_{d1} = (12, 10)$ and $\Delta_{d2} = (-12, -10)$, respectively. The follower UAVs initially are not at the desired
position. The initial distance between \textit{Follower}_1 and its desired position is $\Delta_{01} = (-41.9, -0.9)$, and the initial distance between \textit{Follower}_2 and its desired position is $\Delta_{02} = (-17.5, 0.5)$. \textit{Follower}_1 after 34.8 sec and \textit{Follower}_2 after 14.3 sec reach the formation and then, they will keep the formation. Fig. 9(a) shows the trajectory of \textit{Follower}_1 in the relative frame which starts from $R_{13,11}$ and ends with $R_{1,11}$. Similarly, Fig. 9(b) shows the trajectory of \textit{Follower}_2 which starts from region $R_{10,6}$ and finally reaches the region $R_{1,6}$. Both UAVs have reached one of the first regions in the first circle of the partitioned space.

![Formation Scenario Diagram](image)

**Fig. 8.** The schematic of a formation scenario with two followers and one leader

After 50 sec, the formation switches. For the new formation, the desired distance of the followers from the leader are $\Delta_{d1} = (-30, -10)$ and $\Delta_{d2} = (0, 10)$, while their initial distances from the desired position are $\Delta_{01} = (40.5, 23.3)$ and $\Delta_{02} = (-14.5, -23)$. When the followers are trying to reach the desired formation, at $t = 55.8$ sec, \textit{Follower}_2 enters the alarm zone of \textit{Follower}_1. As described in Section IV-B, to avoid collision, \textit{Follower}_1 asks \textit{Follower}_2 to stop in the relative frame, and then it turns to handle the situation. After removing the collision alarm, both followers have resumed their normal operation to reach and keep the formation. The indices of the traversed regions for $\theta$ and $r$ are shown in Fig. 12. For the new formation, the trajectories of \textit{Follower}_1 and \textit{Follower}_2 in the relative partitioned space are shown in Fig. 10(a) and Fig. 10(b), respectively. As it can be seen both UAVs have reached one of the regions in the first circle in the partitioned space. Moreover, to avoid collision, \textit{Follower}_1 has turned and then, has moved towards the origin. The position of the UAVs in x-y plane is shown in Fig. 11.
Fig. 9. (a) The trajectory of $Follower_1$ in the partitioned relative space. (b) The trajectory of $Follower_2$ in the partitioned relative space.

Fig. 10. (a) The trajectory of $Follower_1$ in the partitioned relative space after switching the formation. (b) The trajectory of $Follower_2$ in the partitioned relative space after switching the formation.
VI. CONCLUSION

In this paper a decentralized hybrid supervisory framework was introduced for the formation of unmanned helicopters. First, using the polar partitioning of the motion space, a finite state discrete model was obtained for the motion dynamics of the UAVs and then, formulating the formation mission over the partitioned space, a modular discrete supervisor was designed to accomplish three main tasks: reaching the formation, keeping the formation, and collision avoidance. Among these tasks, reaching and keeping the formation can be done individually, however, to avoid collision, a tight cooperation of the UAVs is required. Therefore, For the collision avoidance, a scalable decentralized cooperative control mechanism was introduced so
that local (decomposed) supervisors can treat the distributed agents to achieve a globally safe and collision-free environment. The efficiency of the proposed approach was verified through hardware-in-loop simulation results.

ACKNOWLEDGMENT

The authors would like to thank the NUS UAV team particularly Professor Ben. M. Chen, Dr. Xiangxu Dong, Dr. Guowei Cai, and Dr. Feng Lin for their technical support during the flight simulations.

REFERENCES


