Mechatronics 21 (2011) 886–898

Contents lists available at ScienceDirect

Mechatronics

journal homepage: www.elsevier.com/locate/mechatronics

Hybrid formation control of the Unmanned Aerial Vehicles

Ali Karimoddini [a,b], Hai Lin [b,*], Ben M. Chen [b], Tong Heng Lee [b]

[a] National University of Singapore, Graduate School for Integrative Sciences and Engineering (NGS), Centre for Life Sciences (CeLS), #05-01, 28 Medical Drive, 117456 Singapore, Singapore
[b] Department of Electrical and Computer Engineering, Faculty of Engineering, National University of Singapore, Engineering Drive 3, 117576 Singapore, Singapore

ARTICLE INFO

Article history:
Available online 15 December 2010

Keywords:
Unmanned Aerial Vehicles (UAVs)
Formation control
Hybrid supervisory control
Polar partitioning

ABSTRACT

An essential issue in the formation control of Unmanned Aerial Vehicles (UAVs) is to design a reliable controller in their path planner level to handle all interactions between the continuous dynamics of the system and inherent discrete nature of the decision making unit, which has been embedded to coordinate the control submodules. In this paper, we have proposed a new approach of hybrid supervisory control of UAVs for a two-dimensional leader follower formation scenario. The approach is able to comprehensively capture internal relations between the path planner dynamics and the decision making unit of the UAVs. To design such a hybrid supervisory controller for the formation problem, we have introduced a new method of abstraction, based on polar partitioning of the state space. Furthermore, we have utilized the properties of multi-affine vector fields over the polar partitioned space. Within this framework, we design a modular decentralized supervisor in the path planner level of the UAVs to achieve two major goals: first, *reaching the formation* and second, *keeping the formation*. In addition, an *inter-collision avoidance* mechanism has been considered in the controller structure. The approach is robust against uncertainty in the initial state of the system, in the sense that it can bring the follower UAV to the desired position, starting from any arbitrary initial position inside the control horizon. Moreover, the velocity bounds are applied through the design procedure so that the generated velocity references can be given to the lower level of the control hierarchy, as the references to be followed.

1. Introduction

Formation control of cooperative multi-robot systems [1,2], is a rapidly growing area that aims at reaching and keeping a particular form of movement and has numerous applications in ground and aerial robotics. Cooperative control in general and formation control in particular, provide a framework for analysis and design of the team behavior of several autonomous vehicles. A team of robots, taking a cooperative structure, is more robust against the failures in the agents or in the communication links. Moreover, using several simpler robots instead of a complex one, results in a more powerful and flexible structure and could leverage the team efficiency [3].

In the area of the aerial robotics, formation control of the Unmanned Aerial Vehicles (UAVs) has aroused a challenging hot research area and has attracted both academic and military communities [4,5]. This is due to the fact that UAVs are not subjected to the limitations of the ground robots like movement constraints and vision range limitations and therefore, they are quite fit solutions for missions such as terrain and utilities inspection [6], search and coverage [7], search and rescue [8], disaster monitoring [9], aerial mapping [10], traffic monitoring [11], reconnaissance mission [12], and surveillance [13].

A typical formation control scenario consists of several parts, including: *reaching the formation*, *keeping the formation*, and *inter-collision avoidance*. Starting from an initial state, the UAVs should achieve the desired formation within a finite time (*reaching the formation*). Then, they should be able to maintain the achieved formation, while the whole structure needs to track a certain trajectory (*keeping the formation*). Meanwhile, in all of the previous steps, the collision between the agents should be prevented (*inter-collision avoidance*). Definitely, this interacting control structure imposes lots of switching between the control submodules and hence, a decision making unit should be embedded in the control architecture to support this complicated orchestra. The existing formation control strategies, mainly focus on the keeping formation problem in which the formation problem could be reduced to the design of a controller for a system, which has been slightly deviated from the desired configuration [14–16]. There are also some methods that focus on the reaching the formation such as those are based on MILP programming, navigation function and potential field approaches [17–20]. These approaches usually suffer

* Corresponding author.
E-mail addresses: karimoddini@nus.edu.sg (A. Karimoddini), elelh@nus.edu.sg (H. Lin), bmchen@nus.edu.sg (B.M. Chen), eleleeth@nus.edu.sg (T. Heng Lee).

0957-4158/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.mechatronics.2010.09.007

from the high computation cost and difficulty to be implemented decentralizedly.

To integrate all three parts of the formation problem, one solution might be designing one controller for each part based on the continuous dynamics of the system and then, combining them to achieve the whole goal [21]. However, focusing on the continuous dynamics of the system and ignoring the inter relations between the submodules and the effects of the switching between different operation modes may degrade the reliability of the system. Moreover, due to the high risk of crash, UAVs need a fully trustable operation algorithm. Therefore, it is necessary to find a way to take the discrete part into account in addition to the continuous dynamics of the system.

In this paper we propose a *hybrid supervisory control* architecture for the path planner level of the helicopter UAVs, involved in a leader follower formation scenario. Hybrid modeling and control [22], is a powerful framework that can capture both discrete and continuous dynamics of the system, simultaneously and collectively. It provides a comprehensive analysis for interactions between the discrete part and the continuous evolution of the system. There are some efforts of capturing the formation control within a hybrid framework [23–26]. However, they have mostly ended only with the hybrid modeling rather than hybrid analysis of the system. The reason is that despite the strength of hybrid modeling theory, the hybrid analysis tools are typically difficult to apply to the system. In contrast, the proposed approach in this paper, provides a tractable framework for hybrid synthesis of the formation control. Within this framework, we have introduced a new method of abstraction based on polar partitioning of the state space. Furthermore, we have utilized multi-affine function properties over the partitioned space to construct a hybrid model that can be captured by a finite discrete event system (DES) model. Employing this technique, we will reduce the original hybrid system with infinite states into a finite state machine that can be effectively handled by well-established theories of DES supervisory control. Within the DES supervisory control framework, we can design the controller for reaching the formation, keeping the formation, and collision avoidance, modularly.

After designing the DES supervisor, we will show that due to the bisimulation relation between the plant and the abstracted system, the designed DES supervisor for the abstracted system can be directly applied to the original hybrid model of the plant so that the behavior of the hybrid plant and its DES model are the same.

So far, the abstraction approaches based on bisimulation relation are limited to a few simple classes of systems such as timed-automata, multi rate automata, initialized rectangular automata and order minimal hybrid systems [27–30]. Recently, multi-affine vector fields, have been used as a wider and more practical class of hybrid systems, since they are decidable systems under triangulation and rectangulation of the state space [31,32]. However, formulating a formation problem within a rectangulated or triangulated space is not optimal, in the sense that the direct path to reach the desired point is not applicable. Instead, the proposed method of abstraction based on polar partitioning of the state space, can be appropriately applied to the formation problem.

Although the ground robots are not as sensitive as the aerial robots and they may not need such complicated control system, the proposed approach in this paper with some modifications on the path planner dynamics can be also applied to the ground robots, as they have a 2-d path planner in the their embedded control structure.

The rest of this paper is organized as follows. After describing the problem in Section 2, the principles of polar partitioning and the properties of multi-affine vector fields over the partitioned space are proposed in Section 3. In Section 4, we design several multi-affine controllers for each partition such that the controlled system can flexibly stay in the current partition or move to either of its adjacent partitions. Section 5 focuses on the extraction of the finite DES model of the system. Then, in Section 6, a DES supervisor has been designed for both the formation control and the inter-collision avoidance. In Section 7, the designed supervisor has been applied to the original continuous system. The simulation results are presented in Section 8. Section 9 depicts the future plan of our research work. The paper is concluded in Section 10.

2. Problem description

In a leader follower scenario in which the leader tracks an arbitrary generated path, the follower should reach the formation, starting from an initial position inside the control horizon. After reaching the formation, it should be maintained, while the whole formation, as a rigid body, needs to jointly follow the trajectory generated by the leader [33].

Usually the control structure of a UAV is a hierarchical architecture in which the attitude of the UAV is controlled in the lower level and then, the path planner is responsible for the movement of the UAV in the higher level of this hierarchy [34,35]. Therefore, in this structure, if the generated path respects the physical constraints of the UAV motion, the lower level controller of the UAV is able to track it [23]. Hence, for autonomous helicopters, as a large class of UAVs, it is rational to consider the path planner dynamics of the UAV helicopter as a mass point model with the following dynamics:

$$\dot{x} = u \quad x \in \mathbb{R}^2, \quad u \in U \subseteq \mathbb{R}^2 \tag{1}$$

where x is the position of the UAV; u is the UAV velocity, and U is the constraint set. Here, we assume that the UAVs are flying in the same altitude, the altitude control works well, and the follower velocity is in the following form:

$$V_{follower} = V_{leader} + V_{rel} \tag{2}$$

where the follower should reach and keep the formation by an appropriate selection of relative velocity, V_{rel}. Alternatively, one can consider a relatively fixed frame (Fig. 1), in which the follower moves with the velocity of V_{rel}. Here, the control horizon is a neighborhood of the desired position with the radius of R_m. Now, the formation problem can be expressed as follows:

Problem 1. Given the dynamics of the path planner as (1) and the velocity of the follower in the form of (2), design the formation controller to generate the relative velocity of the follower (V_{rel}), such that starting from any initial state inside the control horizon, it eventually reaches the desired position, while avoiding the inter-collision between the leader and the follower. Moreover, after reaching the formation, the follower UAV should remain at the desired position for ever.

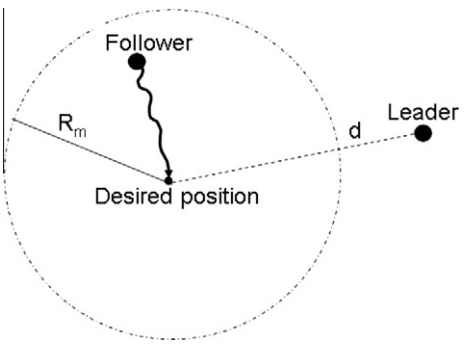

Fig. 1. Relative frame; the follower should reach the desired position starting from any point inside the circle as the control horizon.

To tackle this problem, we will propose the polar partitioning of the state space in which the direct path to the desired position is applicable. We also will utilize the properties of multi-affine functions over the polar abstracted space that will result in a hybrid system and can be captured by a finite DES model which is bisimilar to the original model of the plant. Then, we will design a DES supervisor for the obtained DES model and apply the resulting supervisor to the original hybrid model of the plant, as the bisimulation relation guarantees an equivalent behavior for the DES model and the original one [27].

3. Multi-affine vector fields over polar partitioning of the state space

3.1. Polar partitioning of the state space

In the polar coordinates with $0 \leqslant \theta \leqslant 2\pi$, consider a circle C_{R_m}, with the radius of R_m, which has been partitioned by the curves $\{r = r_i | 0 \leqslant r_i \leqslant R_m,\ \text{for } i < j : r_i < r_j,\ i = 1, \ldots, n_r,\ r_1 = 0,\ r_{n_r} = R_m\}$ and $\{\theta = \theta_j | 0 \leqslant \theta_j \leqslant 2\pi,\ \text{for } j < k : \theta_j < \theta_k,\ j = 1, \ldots, n_\theta, \theta_1 = 0, \theta_{n_\theta} = 2\pi\}$. As an example, $\left\{r_i = \frac{R_m}{n_r - 1}(i-1),\ i = 1, \ldots, n_r\right\}$ and $\left\{\theta_j = \frac{2\pi}{n_\theta - 1}(j-1),\ j = 1, \ldots, n_\theta\right\}$ are such curves that we will use through our further derivations.

Using this notation, we will have $(n_r - 1)(n_\theta - 1)$ partitioning elements. An element $R_{i,j} = \{p = (r,\theta) | r_i \leqslant r \leqslant r_{i+1},\ \theta_j \leqslant \theta \leqslant \theta_{j+1}\}$ is a subset of the circle C_{R_m} surrounded by the above curves. For the partitioned space C_{R_m} we have: $\bigcap int(R_{i,j}) = \emptyset$, and $\bigcup R_{i,j} = C_{R_m}$, where $int(R_{i,j})$ is the interior of the element $R_{i,j}$. Fig. 2, illustrates an example of such a partitioned space with $n_r = 3$ and $n_\theta = 9$.

The intersection between the element $R_{i,j}$ and the partitioning curves generates the *vertices* and the *edges* (Fig. 3). We use $V(e)$ to denote the set of vertices that belong to the edge e, and $E(v)$ for the set of edges that vertex v belongs to them.

The vertices of the element $R_{i,j}$ are arranged as shown in Fig. 4. We use the following notation to label the vertices:

$$\begin{cases} v_0 = v_{00} & r = r_i, \theta = \theta_j \\ v_1 = v_{01} & r = r_{i+1}, \theta = \theta_j \\ v_2 = v_{10} & r = r_i, \theta = \theta_{j+1} \\ v_3 = v_{11} & r = r_{i+1}, \theta = \theta_{j+1} \end{cases} \tag{3}$$

To detect which partitioning curves have generated the vertex $v_{t,u}$, we define the functions Ψ_1: $\{t,u\} \to \{0,1\}$ and Ψ_2: $\{t,u\} \to \{0,1\}$ as:

$$\Psi_1(x) = \begin{cases} 1 & x = 0 \\ 0 & x = 1 \end{cases} \tag{4}$$

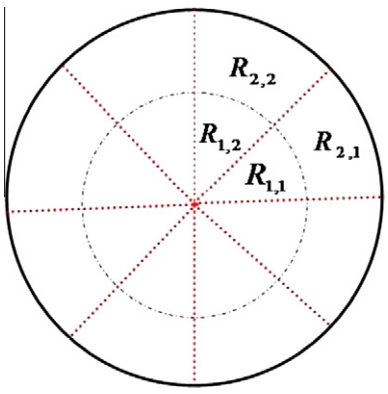

Fig. 2. Partition labels.

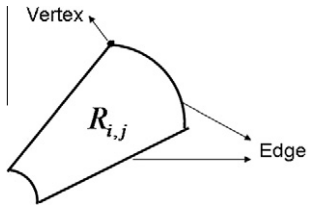

Fig. 3. Vertices and edges in the element $R_{i,j}$.

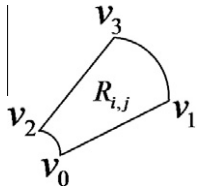

Fig. 4. Vertices of the element $R_{i,j}$.

$$\Psi_2(x) = \begin{cases} 0 & x = 0 \\ 1 & x = 1 \end{cases} \tag{5}$$

The element $R_{i,j}$ has four edges $\{E_r^+, E_r^-, E_\theta^+, E_\theta^-\}$ and correspondingly, four outer normal vectors $\{n_r^+, n_r^-, n_\theta^+, n_\theta^-\}$ (Fig. 5).

Remark 1. The element with $r_i = 0$ (Fig.6), is a special case of the element $R_{i,j}$. In fact, in this element, v_0 and v_2 are coincident, since for both vertices we have $r = 0$; however, their value of θ are different.

3.2. The properties of multi-affine vector fields over the partitioning elements

In this section we will use the properties of multi-affine vector fields over the mentioned partitioned system in the polar domain.

The affine and multi-affine functions are defined as follows:

Definition 1 (*Affine function*). A function $g : \mathbb{R} \to \mathbb{R}$ is said to be affine, if:$\forall x_1, x_2 \in \mathbb{R}$ and $\alpha_1, \alpha_2 \in \mathbb{R}$ satisfying $\alpha_1 + \alpha_2 = 1 : g(\alpha_1 x_1 + \alpha_2 x_2) = \alpha_1 g(x_1) + \alpha_2 g(x_2)$

Definition 2 (*Multi-affine function*).

- A function $g : \mathbb{R}^n \to \mathbb{R}$ is said to be multi – affine, if it is affine in each parameter, meaning that by fixing all parameters, except one, the resulting function is affine with respect to the free parameter.
- A function $g : \mathbb{R}^n \to \mathbb{R}^m$ is said to be multi-affine, if it is multi-affine in each fucntion $g_i : \mathbb{R}^n \to \mathbb{R},\ i = 1, \ldots, m$.

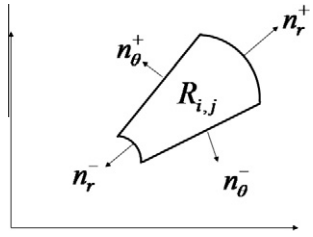

Fig. 5. Outer normals of the element $R_{i,j}$.

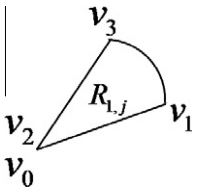

Fig. 6. $R_{1,j}$ is a special case of the element $R_{i,j}$.

In the following proposition, we will show a very useful property of multi-affine functions over the circle C_{R_m} and its partitioning elements $R_{i,j}$. According to the following propositions, a multi-affine function over the element $R_{i,j}$ can be expressed uniquely in terms of the values of the function at the vertices of $R_{i,j}$.

Proposition 1. *Consider a multi – affine function $g(x) : \mathbb{R}^2 \to \mathbb{R}^2$ over the region $R_{i,j}$. The following property always holds true:*

$$\forall x = (r,\theta) \in R_{i,j} : g(x) = \sum_{i=0}^{3} \lambda_i g(v_i) \tag{6}$$

where v_i, $i = 0,\ldots,3$, are the vertices of the element $R_{i,j}$ and λ_i, $i = 0,\ldots,3$, are obtained as follows:

$$\lambda_i = \lambda_r^{\Psi_2(u)}(1-\lambda_r)^{\Psi_1(u)}\lambda_\theta^{\Psi_2(t)}(1-\lambda_\theta)^{\Psi_1(t)} \tag{7}$$

where t and u are the corresponding binary digits of the index i of v_i as declared in (3) and

$$\lambda_r = \frac{r-r_i}{r_{i+1}-r_i} \quad \lambda_\theta = \frac{\theta-\theta_j}{\theta_{j+1}-\theta_j}$$

Proof. Let $x = (r,\theta) \in R_{i,j}$. Then, from the partitioning procedure, we have: $r_i \leqslant r \leqslant r_{i+1}$ and $\theta_j \leqslant \theta \leqslant \theta_{j+1}$. Hence, r and θ can be written affinely as follows:

$$\begin{cases} r = (1-\lambda_r)r_i + \lambda_r r_{i+1} & 0 \leqslant \lambda_r \leqslant 1 \Rightarrow \lambda_r = \frac{r-r_i}{r_{i+1}-r_i} \\ \theta = (1-\lambda_\theta)\theta_j + \lambda_\theta \theta_{j+1} & 0 \leqslant \lambda_\theta \leqslant 1 \Rightarrow \lambda_\theta = \frac{\theta-\theta_j}{\theta_{j+1}-\theta_j} \end{cases}$$

Now, consider a trajectory starting from the vertex v_0 to the point x, and moving only along the polar directions. As an example, $v_0 = (r_i,\theta_j) \overset{step1}{\to} x_1 = (r,\theta_j) \overset{step2}{\to} x_2 = x = (r,\theta)$ is such a trajectory. In fact, in each step of this trajectory, we change only one parameter, and fix the other one to take the advantages of multi-affine functions. When in the function g, the parameter θ is fixed and only r is varying, we use the notation g_θ to highlight the fixedness of θ. The notation $g_\theta|_{\theta_k}(r)$ is used to show that in the function g, the parameter θ is fixed at θ_k and only r can change. Similarly, we can define g_r and $g_r|_{r_k}(\theta)$ for the case that r is fixed and θ is varying. According to the definition, as g is a multi-affine function, g_θ and g_r are affine. Using the properties of the affine functions, since in Step 1, the parameter θ is fixed and only r changes:

$$\begin{aligned} g(x_1) &= g_\theta|_{\theta_j}(r) = g_\theta|_{\theta_j}((1-\lambda_r)r_i + \lambda_r r_{i+1}) \\ &= (1-\lambda_r)g_\theta|_{\theta_j}(r_i) + \lambda_r g_\theta|_{\theta_j}(r_{i+1}) \\ &= (1-\lambda_r)g(r_i,\theta_j) + \lambda_r g(r_{i+1},\theta_j) \end{aligned}$$

In Step 2, θ changes and r is fixed. Therefore:

$$\begin{aligned} g(x_2) &= g_r|_r(\theta) = g_r|_r((1-\lambda_\theta)\theta_j + \lambda_\theta\theta_{j+1}) \\ &= (1-\lambda_\theta)g_r|_r(\theta_j) + \lambda_\theta g_r|_r(\theta_{j+1}) = (1-\lambda_\theta)g(r,\theta_j) + \lambda_\theta g(r,\theta_{j+1}) \\ &= (1-\lambda_\theta)g_\theta|_{\theta_j}(r) + \lambda_\theta g_\theta|_{\theta_{j+1}}(r) \end{aligned}$$

In the first step, we have already obtained $g_\theta|_{\theta_j}(r)$. The same procedure, can be followed to obtain $g_\theta|_{\theta_{j+1}}(r)$. Substituting these two values in the second step, we will obtain $g(x_2)$ as follows:

$$\begin{aligned} g(x_2) &= (1-\lambda_\theta)[(1-\lambda_r)g(r_i,\theta_j) + \lambda_r\ g(r_{i+1},\theta_j)] + \lambda_\theta[(1-\lambda_r)g(r_i,\theta_{j+1}) + \lambda_r\ g(r_{i+1},\theta_{j+1})] \\ &= (1-\lambda_\theta)(1-\lambda_r)g(v_0) + (1-\lambda_\theta)\lambda_r\ g(v_1) + \lambda_\theta(1-\lambda_r)g(v_2) + \lambda_\theta\lambda_r\, g(v_3) \end{aligned}$$

which is equivalent to (6). □

Remark 2. It can be verified that for all $x = (r,\theta) \in R_{i,j}$, the resulting coefficients λ_i, $i = 0,\ldots,3$, have the property that $\lambda_i \geqslant 0$ and $\sum_{i=0}^{3}\lambda_i = 1$.

Corollary 1. *For a multi – affine function defined over the element $R_{i,j}$ and for all edges E_q^s of $R_{i,j}$, $q \in \{r,\theta\}$ and $s \in \{+,-\}$, the following property holds true:*

$$\forall x = (r,\theta) \in E_q^s : g(x) = \sum_{v_i \in V(E_q^s)} \lambda_i g(v_i) \tag{8}$$

where λ_i can be obtained as follows:

- *For edges E_r^+ and E_r^- : $\lambda_i = \lambda_\theta^{\Psi_2(t)}(1-\lambda_\theta)^{\Psi_1(t)}$*
- *For edges E_θ^+ and E_θ^- : $\lambda_i = \lambda_r^{\Psi_2(u)}(1-\lambda_r)^{\Psi_1(u)}$*

Proof. : This is a special case of Proposition 1 and it needs to follow only Step 1 of the proof of the previous proposition. □

In the following proposition, we will show that the coefficients in Proposition 1 and Corollary 1 are unique and there is one and only one multi-affine function over C_{R_m} that have the fixed values of $g(v_i)$ at the vertices of $R_{i,j}$.

Proposition 2. *Consider a map $g : \mathbb{R}^2 \to \mathbb{R}^2$ over the region $R_{i,j}$. There exists one and only one multi-affine function $f : \mathbb{R}^2 \to \mathbb{R}^2$ satisfying $f(v_i) = g(v_i)$, for all $i = 0,1,2,3$.*

Proof. The existence has been already guaranteed by Proposition 1. The proof of uniqueness is by contradiction. Assume that f is not unique, and there is another multi-affine function f' such that $f(v_i) = f'(v_i) = g(v_i)$, or equivalently, $f''(v_i) = f(v_i) - f'(v_i) = 0$, for $i = 0,1,2,3$. Since f and f' are multi-affine, it follows from the definition that $f'' = f - f'$ also is multi-affine. Hence, using Proposition 1, $\forall x \in R_{i,j} : f''(x) = \sum_{i=0}^{3}\lambda_i f''(v_i) = 0$. Therefore, $\forall x \in R_{i,j}$, $f(x) = f'(x)$, which contradicts with the assumption. □

4. Two control features over the polar partitioning elements

Using the properties of the multi-affine functions, we are interested in the behavior of the trajectories of the system over the elements $R_{i,j}$. In particular, we will investigate that under what conditions the trajectories would remain inside an element $R_{i,j}$ for ever *(Invariant region)*, or would deterministicly leave through a particular edge *(Exit edge)*.

4.1. Controller for an invariant region

In an invariant region, the trajectories of the system will remain inside the region, regardless of the initial state of the system. The formal definition of the invariant region is given as follows:

Definition 3 (*Invariant region*). In the circle C_{R_m} and the vector field $\dot{x} = g(x), g : \mathbb{R}^2 \to \mathbb{R}^2$, the region $R_{i,j}$ is said to be invariant region (Fig. 7), if $\forall x(0) \in int(R_{i,j})$, then $x(t) \in R_{i,j}$ for $t \geqslant 0$.

In the following theorem, we will find the conditions to have the region $R_{i,j}$ as an invariant region.

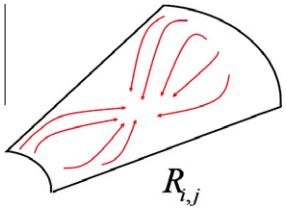

Fig. 7. Invariant region.

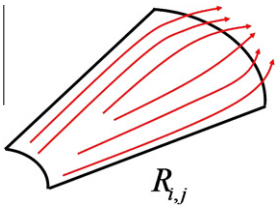

Fig. 8. Exit edge.

Theorem 1 (*Sufficient condition for $R_{i,j}$ to be an invariant region*). *For a continuous multi – affine vector field $\dot{x} = g(x), g : \mathbb{R}^2 \to \mathbb{R}^2$, the element $R_{i,j}$ is an invariant region if for each edge E_q^s of $R_{i,j}$ and its corresponding outer normal n_q^s, $q \in \{r,\theta\}$ and $s \in \{+,-\}$, we have:*

$$n_q^{s\,T} \cdot g(v_i) < 0 \quad \forall v_i \in V(E_q^s) \tag{9}$$

Proof. Each edge E_q^s of $R_{i,j}$ has two vertices. According to the Corollary 1, for each E_q^s, we have the property that $\forall x = (r,\theta) \in E_q^s : g(x) = \sum \lambda_i g(v_i) \quad v_i \in V(E_q^s)$, where $\lambda_i \geqslant 0$ and $\sum \lambda_i = 1$. Hence, knowing that $n_q^{s\,T} \cdot g(v_i) < 0$ *for all* $v_i \in V(E_q^s)$ will result in $n_q^{s\,T} \cdot g(x) < 0$ for all $x \in E_q^s$. Since $g(x)$ is continuous, it will be concluded that $n_q^{s\,T} \cdot g(x) < 0$, for the neighborhood of all $x \in E_q^s$. Therefore, the trajectories of the system cannot touch the edge E_q^s. Alternatively, the trajectories of the system will never leave $R_{i,j}$ through the edge E_q^s. Since this is true for all edges, the trajectories of the system do not leave the region $R_{i,j}$. □

Corollary 2 (*Controller for an invariant region*). *For a continuous multi-affine vector field $\dot{x} = h(x,u(x)) = g(x), g : \mathbb{R}^2 \to \mathbb{R}^2$, $R_{i,j}$ is an invariant region if there exists a controller $u : \mathbb{R}^2 \to U \subseteq \mathbb{R}^2$, such that for each vertex v_i, $i = 0,1,2,3$, with incident edges $E_q^s \in E(v_i)$, and corresponding outer normals n_q^s, $q \in \{r,\theta\}$ and $s \in \{+,-\}$:*

$$U_i = U \cap \left\{ u \in \mathbb{R}^2 | n_q^{s\,T} \cdot g(v_i) < 0, \quad \text{for all} \quad E_q^s \in E(v_i) \right\} \neq \emptyset \tag{10}$$

where the convex set U represents the velocity bounds.

Corollary 2 is the direct implication of Theorem 1. The difference is that: firstly, here, the conditions are arranged in terms of the vertices rather than the edges. Secondly, we have selected the controller such that the controlled system respects the velocity bounds. If one of U_i is empty, this means that the controller is infeasible.

Remark 3. In the case that all $U_i \neq \emptyset$, it is sufficient to pick an arbitrary value for $u(v_i)$ from the set U_i. By this selection, $g(x) = h(x,u(x))$, as a multi-affine function, satisfies the conditions of Theorem 1 and makes $R_{i,j}$ as an invariant region. In addition, using Proposition 1, if we construct $u(x)$ as a multi-affine function, since U is a convex set, the multi-affineness of $u(x)$ will guarantee that $u(x) \in U$, for all $x \in R_{i,j}$. In this case, the feedback controller $u(x)$ at each value of $x \in R_{i,j}$ is uniquely determined by the value of u at the vertices of the region $R_{i,j}$.

4.2. Controller for an exit edge

In the case that the trajectories of the system leave the region from a unique edge, regardless of the initial state of the system, this edge is called the exit edge and formally can be defined as follows:

Definition 4 (*Exit edge*). In the circle C_{R_m} with the vector field $\dot{x} = g(x), \; g : \mathbb{R}^2 \to \mathbb{R}^2$, the edge $E_q^s, \; q \in \{r,\theta\}$ and $s \in \{+,-\}$, is said to be an exit edge (Fig. 8), if $\forall x(0) \in int(R_{i,j})$, there exist $\tau(finite) > 0$ and $\varepsilon > 0$ satisfying:

1. $x(t) \in int(R_{i,j}) \quad for\ t \in [0,\tau)$
2. $x(t) \in E_q^s \quad for\ t = \tau$
3. $x(t) \notin R_{i,j}$ for $t \in (\tau, \tau + \varepsilon)$

In the following theorem, we will explain the condition to have a particular edge in $R_{i,j}$ as an exit edge.

Theorem 2 (*Sufficient condition for an exit edge*). *For a continuous multi – affine vector field $\dot{x} = g(x), \; g(x) : \mathbb{R}^2 \to \mathbb{R}^2$, the edge E_q^s with the outer normal n_q^s, is an exit edge if:*

1. $n_{q'}^{s'\,T} \cdot g(v_i) < 0 \quad \forall E_{q'}^{s'} \neq E_q^s \quad and \quad \forall v_i \in V(E_{q'}^{s'})$
2. $n_q^{s\,T} \cdot g(v_i) > 0 \quad \forall v_i \in R_{i,j}$

where $q, q' \in \{r,\theta\}$ and $s, s' \in \{+,-\}$.

Proof. The first requirement guarantees that the trajectories of the system do not leave $R_{i,j}$ through the edges $E_{q'}^{s'} \neq E_q^s$. This has been already proven in Theorem 1. The second requirement is to drive the trajectories of the system out through the edge E_q^s. According to the Proposition 1, for the multi-affine function g, there exist $\lambda_i \geqslant 0, \; i = 0,\ldots,3$, such that $\forall x = (r,\theta) \in R_{i,j} : g(x) = \sum_{i=0}^{3} \lambda_i g(v_i)$. Since λ_i are positive, then, $n_q^{s\,T} \cdot g(v_i) > 0$, for all v_i, will result in $n_{q'}^{s'T} \cdot g(x) = \sum_{i=0}^{2} \lambda_i n_{q'}^{s'T} \cdot g(v_i) > 0$, for all $x \in R_{i,j}$.This means that the trajectories of the system have a strictly positive velocity in the direction of n_q^s, steering them to exit from $R_{i,j}$ through the edge E_q^s. □

Corollary 3 (*Controller for an Exit edge*). *For a continuous multi-affine vector field $\dot{x} = h(x,u(x)) = g(x), \; g : \mathbb{R}^2 \to \mathbb{R}^2, E_q^s$ with the outer normal $n_q^s, q \in \{r,\theta\}$ and $s \in \{+,-\}$, is an exit edge if there exists a controller $u : \mathbb{R}^2 \to U \subseteq \mathbb{R}^2$, such that for each vertex v_i, $i = 0,1,2,3$, the following property holds true:*

$$\begin{aligned} U_i = U \bigcap \{ u \in \mathbb{R}^2 | & n_q^s \cdot g(v_i) > 0, \text{ for all } v_i \text{ and} \\ & n_{q^*}^{s^*} \cdot g(v_i) > 0, \text{ for all } E_{q^*}^{s^*} \neq E_q^s, \quad v_i \in \left(E_{q^*}^{s^*} \right) \} \neq \emptyset \end{aligned} \tag{11}$$

where the convex set U represents the velocity bounds.

Corollary 3 is the direct implication of Theorem 2 for which we check the feasibility of the feedback controller, applying the velocity constraints U. Clearly, if one of the above sets U_i, $i = 0,\ldots,3$, is empty, the controller is not feasible. Moreover, if we construct u as a multi-affine function, since U is a convex set, the multi-affineness of $u(x)$ will guarantee that the resulting system respects the velocity bounds.

Remark 4. The strictly positive inequalities in Theorem 2, guarantee that the trajectories of the system, immediately leave the exit edge upon reaching this edge so that we can ignore the time duration that the trajectories spend on the exit edges.

4.3. Construction of the controller

In previous sections, the feasibility conditions for an invariant region and an exit edge were explained. Here, we will explain

the construction of the controller by computation of the sets U_i which are introduced in Corollaries 2 and 3.

First, we construct the controller C_0 to make the region $R_{i,j}$ as an invariant region. According to the Theorem 1, in order to construct an invariant region, (9) should be satisfied for all four edges of $R_{i,j}$. Starting from the edge E_r^+, we then need to have:

$$n_r^{+T} \cdot g(v_i) < 0 \quad \text{for } v_1 \quad \text{and} \quad v_3 \tag{12}$$

where n_r^+ varies with the angle β as:

$$n_r^+ = 1\angle\beta \quad \theta_j \leqslant \beta \leqslant \theta_{j+1} \tag{13}$$

The solutions of (12) for a particular value of β can be selected from the set $\{g|\beta+\frac{\pi}{2} < \angle g < \beta+\frac{3\pi}{2}\}$, which is shown by the gray region in Fig. 9. Consequently, the solutions that satisfy (12) for all values of β inside the interval $[\theta_j,\theta_{j+1}]$, are the the vectors selected from the set $g_{E_r^+} = \{g|\theta_{j+1}+\frac{\pi}{2} < \angle g < \theta_j+\frac{3\pi}{2}\}$, as it is shown in Fig. 10.

Following this procedure, the legal regions for other edges can be obtained accordingly:

$$g_{E_r^-} = \left\{g|\theta_{j+1} - \frac{\pi}{2} < \angle g < \theta_j + \frac{\pi}{2}\right\}$$
$$g_{E_\theta^+} = \{g|\theta_{j+1} - \pi < \angle g < \theta_{j+1}\}$$
$$g_{E_\theta^-} = \{g|\theta_j < \angle g < \theta_j + \pi\}$$

It follows from $v_0 \in V(E_r^-) \cap V(E_\theta^-)$ that the legal value for $g(v_0)$ should be selected from the set $\Omega_0 = g_{E_r^-} \cap g_{E_\theta^-}$.Therefore, we can find the legal sets Ω_i, for all v_i, $i = 0,1,2,3$, as follows:

$$\begin{cases} \Omega_0 = \{g|\theta_j < \angle g < \theta_j + \frac{\pi}{2}\} \\ \Omega_1 = \{g|\frac{\pi}{2} + \theta_{j+1} < \angle g < \pi + \theta_j\} \\ \Omega_2 = \{g|\theta_{j+1} - \frac{\pi}{2} < \angle g < \theta_{j+1}\} \\ \Omega_3 = \{g|\pi + \theta_{j+1} < \angle g < \theta_j + \frac{3\pi}{2}\} \end{cases} \tag{14}$$

According to Theorem 1, to have the region $R_{i,j}$ as an invariant region, it suffices to select an arbitrary value for $g(v_i)$ from the set Ω_i. In other words, it is required to design the controller $u(v_i)$ such that $g(v_i)$ be located inside the set Ω_i. Here, for the path planner dynamics, given by (1), we have $\dot{x} = g(x) = u$. Therefore, to have the region $R_{i,j}$ as an invariant region, it is sufficient to select the values of $u(v_i)$ from the set $U_i = U \cap \Omega_i$ and construct the controller $u(x)$ based on Proposition 1. Recall that the set U represents the velocity constraints. Therefore, since U is a convex set and u is constructed as a multi-affine function, selecting $u(v_i)$, $i = 0,\ldots,3$ from the set U, will result in $\forall x \in R_{i,j}$, $u = g(x) \in U$.

As it can be seen in (14), the sets Ω_i, $i = 0,\ldots,3$, are not empty. Moreover, if the velocity constraints are expressed as a convex set containing the origin, the sets $U_i = U \cap \Omega_i$, $i = 0,\ldots,3$, are not empty. Hence, for the dynamics $\dot{x} = u$, the controller C_0 exists and can be constructed as discussed above.

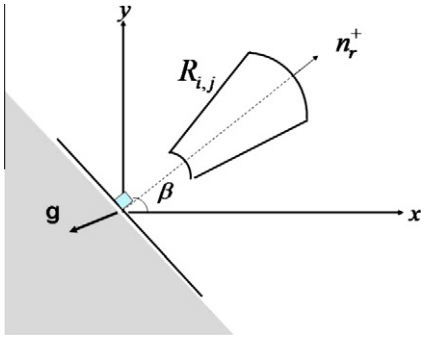

Fig. 9. The solution of (12) for a particular value of β.

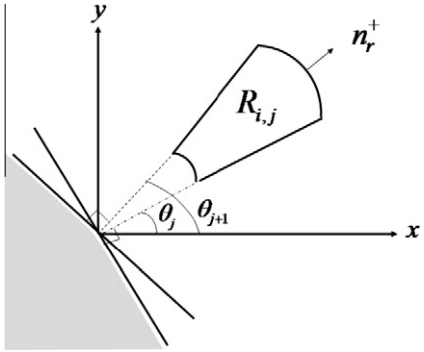

Fig. 10. All solution of (12) for $\theta_j \leqslant \beta \leqslant \theta_{j+1}$.

The same procedure can be followed for the exit edges. As an example, the sets U_i for the edge E_r^-, as an exit edge, are as follows:

$$\begin{cases} U_0 = U_1 = U \bigcap \{g|\theta_{j+1} + \frac{\pi}{2} < \angle g < \theta_j + \pi\} \\ U_2 = U_3 = U \bigcap \{g|\theta_{j+1} + \pi < \angle g < \theta_j + \frac{3\pi}{2}\} \end{cases} \tag{15}$$

In summary, for each region $R_{i,j}$, there exist five controllers that can keep the trajectory of the system inside the region or drive it out through one of four exit edges. This result has been formally illustrated in the following proposition.

Proposition 3. [Existence of the Controllers] *Let $u : \mathbb{R}^2 \to \mathbb{R}^2$ as a multi-affine feedback controller for the system described by $\dot{x} = u$, over a polar partitioned space C_{R_m} with the partitioning elements $R_{i,j}$ and with the given convex velocity constraints that contains the origin. Then, there exists a controller, labeled by C_0, that makes the element $R_{i,j}$ as an invariant region and also, there exist the controllers, labeled by C_q^s, that make the edges E_q^s as exit edges where $q \in \{r,\theta\}$ and $s \in \{+,-\}$.*

5. DES model of the system

Using the above partitioning procedure and then, finding the controllers to either make each region $R_{i,j}$ as an invariant region or make an edge as an exit edge for this region, we will obtain a hybrid system with infinite state that can be converted to a finite state machine bisimilar to the original one. For a finite state machine, one can construct a supervisor using control synthesis techniques based on the Discrete Event Systems (DES) supervisory control theory initiated by Ramadge and Wonham [36]. The following part will introduce some notations to describe a finite state machine by an automaton and then, we will extract the DES model of the system within this framework.

5.1. Representation of a finite state machine

Formally, a finite state machine can be represented by an automaton which is a quintuple $G = (X,\Sigma,\alpha,X_0,X_m)$, where X is the set of states; $X_0 \subseteq X$ is the set of initial states; X_m is the set of final (marked) states; Σ is the (finite) set of events, and $\alpha: X \times \Sigma \to X$ is the transition function. α is a partial function and determines the possible transitions in the system caused by an event.

The sequence of events composes a string. ε is an empty string, and Σ^* is the set of all possible strings over the set Σ including ε.

The function α can be extended to the strings as $\alpha_{ext}: X \times \Sigma^* \to X$. In this case, α_{ext} can be constructed inductively, as follows:

- $\alpha_{ext}(x, \varepsilon) = x$
- $\alpha_{ext}(x, s\sigma) = \alpha(\alpha_{ext}(x,s), \sigma)\ \forall s \in \Sigma^*$ and $\sigma \in \Sigma$

The language of the automaton G, denoted by $L(G)$ or L_G, is the sequence of strings that can be generated by G, following the legal transition relations as:

$$L(G) = \{s \in \Sigma^* | \exists x_0 \in X_0 \ s.t. \quad \alpha_{ext}(x_0, s) \ is\ defined.\}$$

The marked language, denoted by $L_m(G)$ or L_{m_G}, consists of the strings that can be generated by the automaton G and end with the marked states:

$$L_m(G) = \{s \in \Sigma^* | \exists x_0 \in X_0 \ s.t. \quad \alpha_{ext}(x_0, s) \quad is\ defined \quad \text{and} \quad \alpha_{ext}(x_0, s) \in X_m\}$$

The event set Σ consists of two types of events: controllable event set Σ_c and uncontrollable event set Σ_{uc}. The controllable events are those that can be disabled or enabled by an external supervisor; however, the uncontrollable events cannot be affected by the supervisor.

5.2. Extracting the DES model of the system

Using the above notations, one can extract the DES model of the partitioned system. Here, the states of the system are regions $R_{i,j}$, for which the controller will determine the next destination state of the transitions, deterministicly. According to Proposition 3, for the given dynamics $\dot{x} = u$ and for each region $R_{i,j}$, there exist four multi-affine controllers C_q^s that can lead the trajectories of the system out through the exit edges E_q^s with $q \in \{r, \theta\}$ and $s \in \{+, -\}$, and there exists one controller C_0 that can keep the state of the system inside the region $R_{i,j}$.

In addition to the controllable events, when the trajectories of the system cross the partitioning curves, they generate observable events which inform the DES controller about the current state of the system. Indeed, according to Corollary 3, after issuing a command $\{C_q^s | q \in \{r, \theta\}, s \in \{+, -\}\}$, the trajectories of the system will leave the region form the corresponding exit edge. Hence, after issuing a controllable event C_q^s, the system remains in a detection state $d_{i,j}$ until the trajectory of the system crosses the exit edge. Upon crossing the exit edge, the event $O_{i,j}$ will be generated which can be adopted as a feedback information for the supervisor. These events are inherently uncontrollable, in the sense that depending on the initial state of the system, according to Corollary 3, the generated trajectory eventually will cross the exit edge within a finite time, but the exact time is not known and the system should wait to observe the event.

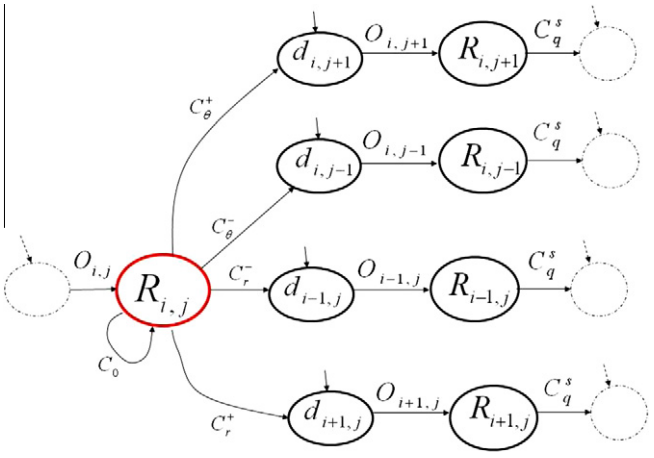

Fig. 11. DES model of the system with controllable and uncontrollable event sets $\Sigma_c = \{C_r^+, C_r^-, C_\theta^+, C_\theta^-, C_0\}$ and $\Sigma_{uc} = \{O_{i,j} | 1 \leqslant i \leqslant n_r - 1, \quad 1 \leqslant j \leqslant n_\theta - 1\}$.

Moreover, at the initial step of the algorithm, the plant starts from one of the detection states and recognizes in which state now the system is. Then, based on detected information, an appropriate controllable event will be generated by the supervisor to control the system, accordingly.

The graph representation that models some of the states and transitions in this system has been depicted in Fig. 11. The arrows starting from $R_{*,*}$ and ending with $d_{*,*}$ are labeled the same as the controller labels. These labels can be captured by the controllable events in the DES model. The arrows starting from $d_{*,*}$ and ending with $R_{*,*}$ are labeled with the observable events $O_{i,j}$, $1 \leqslant i \leqslant n_r - 1$, $1 \leqslant j \leqslant n_\theta - 1$. The entering arrows stand for the initial states. As it is shown in Fig. 11, the system can start from any of the detection states. The controller job is to bring the trajectories of the system to one of the regions $R_{1,j}, 1 \leqslant j \leqslant n_\theta - 1$, as the final states, regardless of the initial state of the system.

Regarding the above discussions, the DES model of the plant can be represented as follows:

$$G = (X, \Sigma, \alpha, X_0, X_m)$$

- $X = \{R_{i,j} | 1 \leqslant i \leqslant n_r - 1, \ 1 \leqslant j \leqslant n_\theta - 1\} \bigcup \{d_{i,j} | 1 \leqslant i \leqslant n_r - 1, 1 \leqslant j \leqslant n_\theta - 1\}$
- $X_0 = \{d_{i,j} | 1 \leqslant i \leqslant n_r - 1, 1 \leqslant j \leqslant n_\theta - 1\}$
- $X_m = \{R_{1,j} | 1 \leqslant j \leqslant n_\theta - 1\} \subseteq X$
- $\Sigma = \Sigma_c \cup \Sigma_{uc}$ where
  - $\Sigma_c = \{C_r^+, C_r^-, C_\theta^+, C_\theta^-, C_0\}$
  - $\Sigma_{uc} = \{O_{i,j} | 1 \leqslant i \leqslant n_r - 1, 1 \leqslant j \leqslant n_\theta - 1\}$
- $\alpha(R_{i,j}, \sigma) = \begin{cases} R_{i,j} & \sigma = C_0 \\ d_{i+1,j} & \sigma = C_r^+ \ for\ i \neq n_r - 1 \\ d_{i-1,j} & \sigma = C_r^- \quad for\ i \neq 1 \\ d_{i,j+1} & \sigma = C_\theta^+ \ for\ j \neq n_\theta - 1 \\ d_{i,1} & \sigma = C_\theta^+ \ for\ j = n_\theta - 1 \\ d_{i,j-1} & \sigma = C_\theta^- \ for\ j \neq 1 \\ d_{i,n_\theta - 1} & \sigma = C_\theta^- \ for\ j = 1 \end{cases}$

$\alpha(d_{i,j,\sigma}) = R_{i,j} \quad \sigma = O_{i,j}$

Remark 5. In the definition of the transition function α, we have two exceptions. Firstly, for the states $R_{1,j}$, $1 \leqslant j \leqslant n_\theta - 1$, the controller C_r^- is not defined, since, if we define the controller C_r^-, the transition to the next state is unknown and it becomes nondeterministic. Secondly, for the states $R_{n_r-1,j}$, $1 \leqslant j \leqslant n_\theta - 1$, the transition C_r^+ is undefined, due to the fact that this controller will lead the trajectories of the system outwards the control horizon, C_{R_m}.

5.3. Refining the DES model of the plant

Since in the obtained DES model the initial state is not unique, hence, the resulting DES model is nondeterministic. Moreover, depending on n_r and n_θ, the number of states could be too large to analyze the system. Therefore, it is required to refine the plant representation to obtain a deterministic model with reduced number of states. To refine the plant model, it is sufficient to merge the states with similar situation. In this case each of the finial states $R_{1,j}$, the last states $R_{n_r-1,j}$, the states $R_{i,j}$, $i \neq 1$ and $i \neq n_r - 1$, and the detection states $d_{i,j}$ will fall into new separate states. The refined model and the original plant are language equivalent, meaning that $L(G) = L(G_{ref})$ and $L_m(G) = L_m(G_{ref})$ [37]. The resulting refined model of the plant is shown in Fig. 12.

In the next section, we will design a supervisor for the extracted DES model of the system to satisfy the control requirements.

6. Designing a DES supervisor for the system

Here, the control objective is to bring the trajectories of the system to the desired position, regardless of the initial state of

the system. Meanwhile, we should avoid the collision between the UAVs. The design procedure is modular, i.e., we first will design the controller for reaching and keeping the formation and then, we will design the controller for collision avoidance. Finally, the whole controller will be obtained using the parallel composition operation.

6.1. Designing a DES supervisor for reaching and keeping the formation

Using the polar partitioning approach, the formation can be achieved if the controller drives the system directly to the regions $R_{1,j}, 1 \leqslant j \leqslant n_\theta - 1$, as it is shown in Fig. 13. Clearly, after reaching the formation by this scenario, the controller should maintain the state of the system at the final state to keep the formation. This specification is realized by the automaton H in Fig. 14. The language generated by this automaton is denoted by L_{spec}. The specification is controllable with respect to the language $L(G) = L(G_{ref})$ and the event set Σ_{uc} since:

$$\forall s \in L_{spec} \quad \text{and} \quad \sigma \in \Sigma_{uc} \quad \text{and} \quad s\sigma \in L(G), \text{then} \quad s\sigma \in L_{spec}.$$

In words, L_{spec} is controllable since 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 L_{spec} [37].

To design the supervisor for the above controllable specification, we use the parallel composition as a binary operation to disable undesirable strings.

Definition 5 (*Parallel Composition*). [38] Given two automata $G_1 = (X_1, \Sigma_1, \alpha_1, x_{0_1}, X_{m_1})$ and $G_2 = (X_2, \Sigma_2, \alpha_2, x_{0_2}, X_{m_2})$, $G = G_1 \| G_2 = (X, \Sigma, \alpha, x_0, X_m)$ is said to be the parallel composition of G_1 and G_2 with:

- $X = X_1 \times X_2$
- $\Sigma = \Sigma_1 \cup \Sigma_2$
- $x_0 = x_{0_1} \times x_{0_2}$
- $X_m = X_{m_1} \times X_{m_2}$
- $\forall (x_1, x_2) \epsilon X, \sigma \epsilon \Sigma$, *then* $\alpha(x, \sigma) =$

$$\begin{cases} \bullet\ (\alpha_1(x_1,\sigma), \alpha_2(x_2,\sigma)) \\ \quad \textit{if } \alpha_1(x_1,\sigma)! \textit{ and } \alpha_2(x_2,\sigma)! \textit{ and } \sigma \epsilon \Sigma_1 \bigcap \Sigma_2 \\ \bullet\ (\alpha_1(x_1,\sigma), x_2) \quad \textit{if } \alpha_1(x_1,\sigma)! \textit{ and } \sigma \epsilon \Sigma_1 - \Sigma_2 \\ \bullet\ (x_1, \alpha_2(x_2,\sigma)) \quad \textit{if } \alpha_2(x_2,\sigma)! \textit{ and } \sigma \epsilon \Sigma_2 - \Sigma_1 \\ \bullet\ \textit{undefined} \quad \textit{otherwise} \end{cases}$$

In this operation, the common events should happen synchronously, whereas the private events can happen independently.

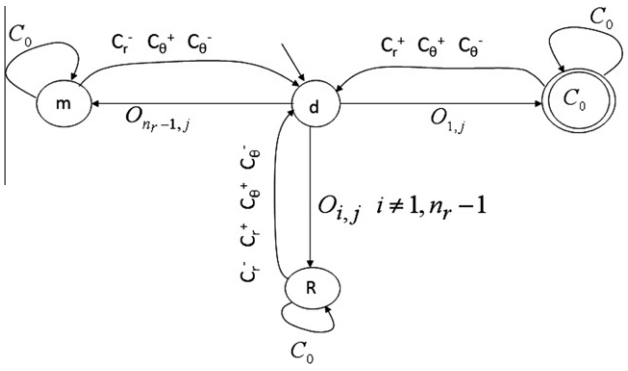

Fig. 12. Refined model of the plant G_{ref}.

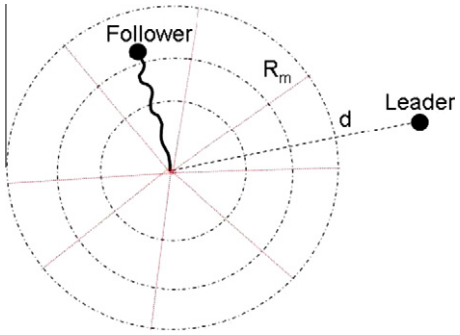

Fig. 13. The control idea is driving the state of the system directly to the regions $R_{1,j}$.

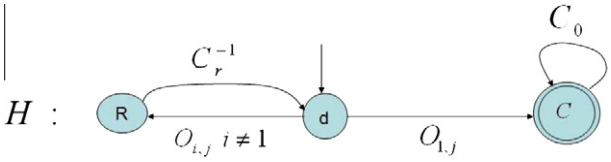

Fig. 14. The control specification of the system.

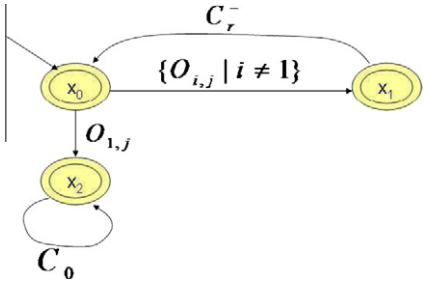

Fig. 15. Supervisor automaton S_f for the formation control.

Using the parallel composition operator, we design the supervisor S_f as it is shown in Fig. 15. It can be seen that the supervisor and the specification have a similar structure. The difference is that all the states of the supervisor are marked. This will allow that the marked states of the closed loop system to be solely determined by the plant.

The closed loop system, can be obtained based on the parallel composition operator as $G_{cl} = G \| S_f$. Since all the events are common between the plant and the supervisor and the specification is a sublanguage of the plant language, the generated and marked languages of the closed loop system are:

$$L_{G_{cl}} = L(G \| S_f) = L(G) \cap L(S_f) = L(G_{ref}) \cap L(S_f) = L(S_f) = L_{spec}$$
$$L_{m_{cl}} = L(G \| S_f) \cap L_m(G) = L_{spec} \cap L_m(G) = L_{m_{spec}}$$

The refined closed loop is shown in (Fig. 16).

6.2. Designing a DES supervisor for the collision avoidance

So far, we had assumed that the leader is not inside the control horizon ($R_m < d$). For the general case, that the leader can be anywhere, it is possible to have collision between the leader and the follower when the follower is in region $R_{i,j}$ and the leader is in region $R_{k,j}$, $k < i$ (Fig. 26). To prevent the inter-collision, it is sufficient that the follower do not enter to the region in which the leader is

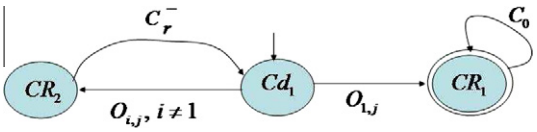

Fig. 16. The refined closed loop system.

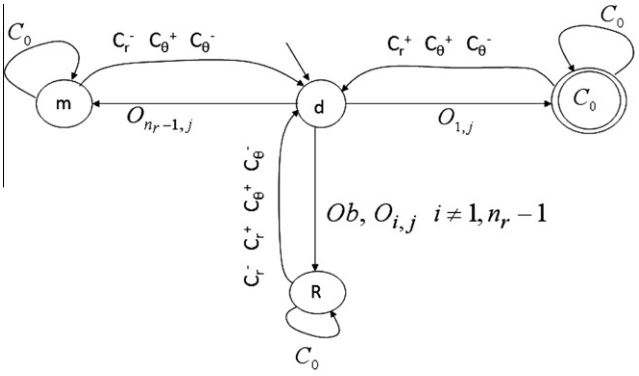

Fig. 17. Modified DES model of the plant.

located. This scenario can be implemented by another supervisor S_{Ca}. Before that, we need to slightly modify the plant such that it generates an event for the detection of the leader as an obstacle (Fig. 17). Accordingly, the supervisor S_f will be changed so that it does nothing when facing with a collision situation. The collision avoidance will be then tackled by the supervisor S_{Ca}, in a modular

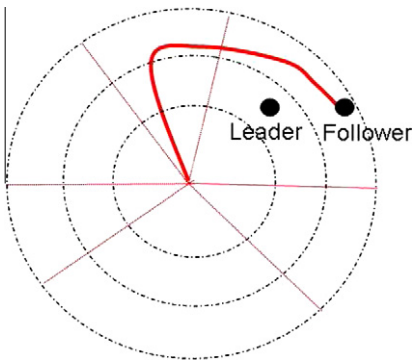

Fig. 18. Inter-collision avoidance scheme.

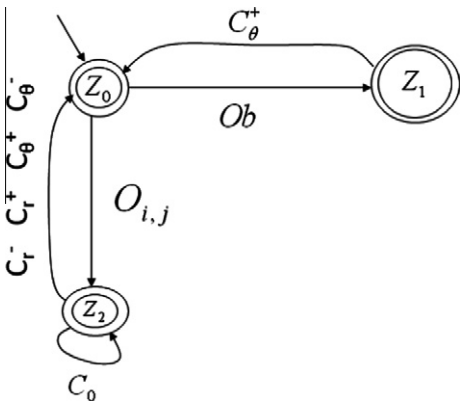

Fig. 19. Inter-collision avoidance supervisor.

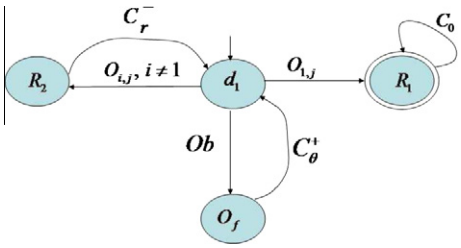

Fig. 20. The refined closed loop system automaton $G_{cl_{ref}}$, where $G_{cl_{ref}} = S||G_{ref}$ and $S = S_f||S_{Ca}$.

fashion. In the modified plant, if the leader is on the way of the follower to the origin, knowing the position of the leader, the follower will generate the event *Ob* to inform the supervisor about the danger of collision.

Fortunately, the leader has a fix position in the relative frame. Therefore, to guarantee the inter-collision avoidance between the leader and the follower, it is sufficient that after observing the event *Ob*, the follower turns anticlockwise, and then, again moves towards the desired position (Fig. 18). This plan can be designed in a modular way, i.e., first, we design a supervisor for the collision avoidance and then, we combine it with the modified supervisor S_f. The designed collision avoidance supervisor S_{Ca} has been shown in Fig. 19.

6.3. The closed loop system

Obtaining the formation supervisor and collision avoidance supervisor as above, the whole controller can be achieved through the parallel composition of these control submodules, $S = S_f||S_{Ca}$. The refined closed loop system is the parallel composition of the refined plant and the whole supervisor, $G_{cl_{ref}} = S||G_{ref}$. The refined closed loop system is shown in Fig. 20.

7. Applying the controller to the original continuous plant

In the previous sections, utilizing the proposed polar partitioning method combined with the designed multi-affine controllers, we achieved a hybrid system that was abstracted to a finite DES model.

The advantage of the abstraction is that it reduces an infinite state system into a finite state machine which can be properly handled by the well-established theories of DES supervisory control, as we saw in the previous section.

Indeed, if we can construct the abstracted system such that it bisimulates the original continuous system, the designed supervisor for the abstracted system is guaranteed to work for the original plant as well. By convention, we say that the original system is similar to the abstracted one when for any transition in the original plant, there exists a transition in the abstracted system. If the converse is also true, we say that these two systems are bisimilar. In fact, the bisimulation relation, guarantees the same behavior in two systems [27].

By construction and by utilizing the multi-affine controllers, the exit edge for each of the control labels is unique. Moreover, for each exit edge, the adjacent partition that is common in that exit edge is unique. Therefore, the transitions in the DES model are deterministic and the DES model bisimulates the abstracted model (Fig. 21).

The bisimulation relation between the DES plant and the original continuous model, will result in the same behavior in these systems, so that the DES supervisor for the abstracted system can be easily applied to the original plant as it is shown in Fig. 22. To refine the DES supervisor, it is sufficient to apply the corresponding

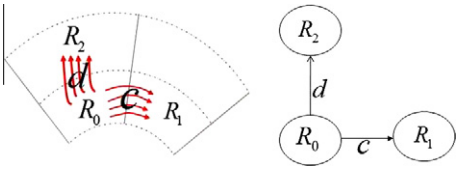

Fig. 21. Bisimulation relation between the plant and abstracted model.

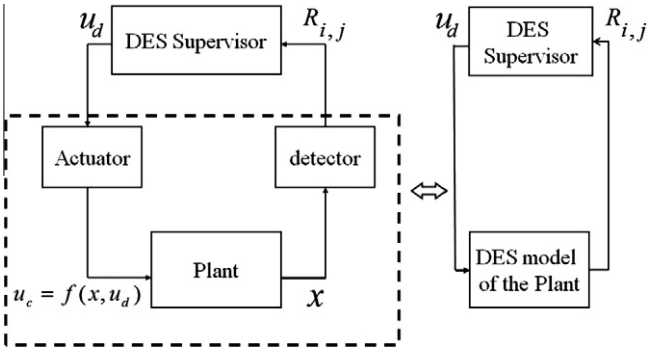

Fig. 22. Applying the discrete supervisor to the original plant.

continuous controller $u_c(x)$ instead of the control labels $u_d \in \Sigma_c$, where $u_c(x) = f(x, u_d) = \sum_{i=0}^{3} \lambda_i(x) u_d(v_i)$ where $u_d(v_i)$, $i = 0, \ldots, 3$, are determined at the vertices v_i, corresponding to the control label u_d. This translation from control labels to the continuous control law is the role of *Actuator* block shown in Fig. 22. In this figure, the *Detector* block recognizes the current position of the UAV and considering the partitioning curves, it generates the observable events for the supervisor. The set of Plant, Detector, and Actuator blocks together can be captured by the DES model of the system, as discussed in Section 5.

8. Simulation

This control architecture has been applied to a leader- follower formation problem. Let $R_m < d$, and assume that the follower path planner dynamics is given as (1). The control horizon is 20 m, and we have selected $n_r = 5$ and $n_\theta = 13$. Using Corollaries 2 and 3, the sets U_i for the control labels C_0 and C_r^- are given as (14) and (15), respectively. It is sufficient that we pick up the appropriate values for $u(v_i)$ at each vertex from the sets U_i, while respecting the velocity bounds. With these values of $u(v_i)$, we can construct the feedback control law $u(x)$ based on Corollaries 2 and 3. Here, the velocity amplitude of the UAV in the relative frame is required to be <1.5 m/s. Therefore, to meet the velocity bounds, it is sufficient to select the amplitude of $u(v_i) < 1.5$ m/s.

According to the above discussion, for the controller C_0, the selected values of $u(v_i)$ at the vertices of the region $R_{i,j}$ are selected as follows:

$$\begin{cases} u(v_0) = 1\angle\left(\frac{\pi}{2} + \theta_j - 0.1(\theta_{j+1} - \theta_j)\right) \\ u(v_1) = 1\angle\left(\frac{\pi}{2} + \theta_{j+1} + 0.1(\theta_{j+1} - \theta_j)\right) \\ u(v_2) = 1\angle\left(\frac{3\pi}{2} + \theta_{j+1} + 0.1(\theta_{j+1} - \theta_j)\right) \\ u(v_3) = 1\angle\left(\frac{3\pi}{2} + \theta_j - 0.1(\theta_{j+1} - \theta_j)\right) \end{cases}$$

where $1 \leqslant i \leqslant n_r - 1, 1 \leqslant j \leqslant n_\theta - 1$.

The controller $u(x)$, corresponding to the control label C_0 for the region $R_{i,j}$, is in the form of $u(x) = \sum_{i=0}^{3} \lambda_i u(v_i)$, where $\lambda_i(x)$ can be obtained through Proposition 6. We can follow the same procedure to construct the controller for the control label C_r^-.

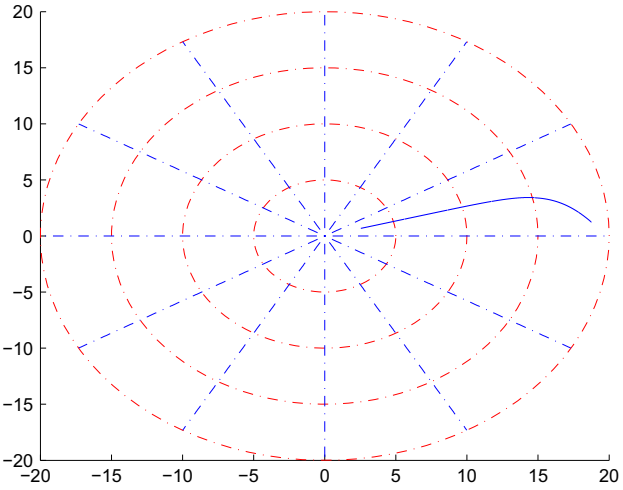

Fig. 23. Simulation of the system for an initial state inside the region $R_{4,1}$.

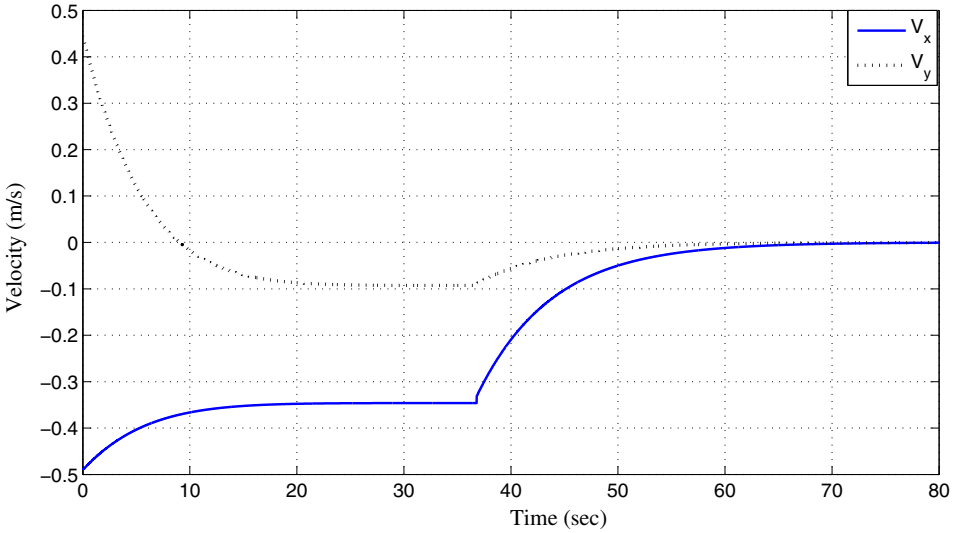

Fig. 24. Generated velocities V_x and V_y for an initial state inside the region $R_{4,1}$.

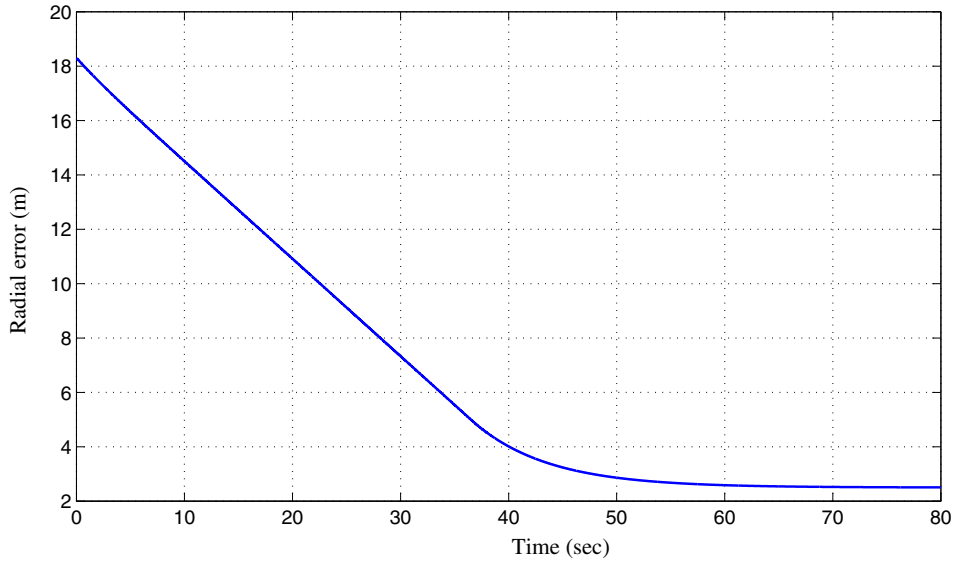

Fig. 25. Absolute distance from the desired position.

Using the above mentioned multi-affine controllers and applying the DES supervisor, for an initial state inside the region $R_{4,1}$, the trajectory of the system and the generated velocities are shown in Figs. 23 and 24, respectively. As it can be seen in Fig. 24, the generated velocities respect the velocity bounds. These generated velocities should be given to the lower level of control architecture as the references to be tracked.

The absolute value of the distance of the UAV from the desired position is shown in Fig. 25. It has finally reached the first circle, i.e., one of the regions $R_{1,j}$, and the general specification has been achieved; however, it has not exactly reached the desired position and there is a steady state error for this controller. This can be treated by two approaches. First, we can make the partitions finer, or at least reduce the diameter of the first circle. Note that other circles do not have any effect on the steady state error, hence, the final precision depends only on the first circle diameter. However, from the practical point of view, the diameter cannot be reduced drastically, due to the limitations on the sampling frequency and the accuracy of the sensors. An alternative approach is tuning the vector fields in the regions $R_{1,j}$. In this example, we have selected symmetric values for $u(v_i)$, $i = 0,\ldots,3$, at all four vertices of $R_{1,j}$. Therefore, in the case of invariant region, the equilibrium point will be at the center of the region $\left(\theta_f = \frac{\theta_j+\theta_{j+1}}{2} \text{ and } r_f = \frac{r_1}{2}\right)$. However, by selecting a smaller value for $u(v_0)$ and $u(v_2)$, the equilibrium point will be pushed towards the origin and the steady state error will be reduced.

In another example, to simulate the collision avoidance, when $R_m > d$, assuming that the leader is located in $R_{3,1}$, and the initial state of the follower is inside the region $R_{4,1}$, the generated path for the collision avoidance mechanism and reaching the formation is depicted in Fig. 26. The control mechanism was explained in Section 6.2.

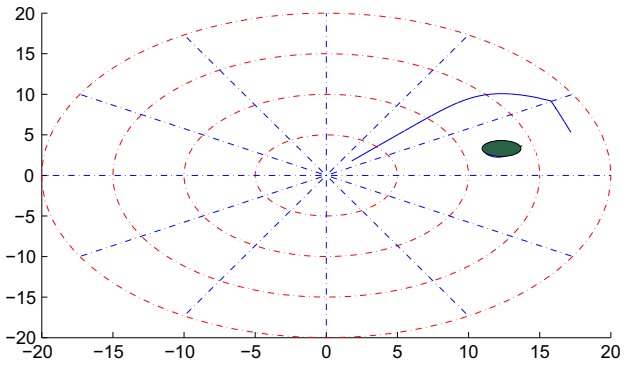

Fig. 26. Collision avoidance mechanism.

9. Future plan

The developed framework for hybrid formation control, is going to be implemented on the cooperative test-bed of NUS UAV helicopters. This test-bed consists of two small-scale UAV helicopters as shown in Fig. 27. The avionic systems of these UAVs are composed of a *PC*\104 computer as an onboard airborne computer system that is modularly extended by extension boards such as an A\D card, a DC–DC convertor card, and a serial communication board. In this avionic system, the IMU sensor (NAV420, Crossbow), as the major sensor, provides three-axis velocities, acceleration, and angular rates in the body frame, as well as longitude, latitude, relative height, and heading, pitch, and roll angles. The avionic system is also equipped with several servo actuators that can manipulate the helicopter to move in different directions. All of these actuators are controlled by a servo board (HBC101, Pontech) as a local controller. The details of the hardware of these UAVs are explained in [39]. The onboard computer program is developed using the QNX Neutrino as a real-time operating system. The developed software is able to communicate with the data acquisition board (DIAMOND-MM-32-AT, Diamond), servo board, and the IMU sensor. It is also able to log the data, compute the control signals, and communicate with the ground station unit via a wireless communication board (IM-500X008, FreeWave). The details of the onboard program are illustrated in [40]. In [35], it has been shown that these UAVs are able to have perfect altitude control during the flight missions. Moreover, they can follow the generated path as long as it respects the velocity bounds [34]. Therefore, this cooperative test-bed is appropriate for implementation of our hybrid formation scenario.

Although with a perfect heave control, the formation of the UAVs with a fixed altitude can be reduced to a two-dimensional problem, a more interesting problem, that we will focus in our fu-

Fig. 27. NUS cooperative UAVs test-bed.

ture research work, is to develop and formulate a 3D multiple follower scenario algorithm for the formation problem within hybrid supervisory control framework.

The achieved hybrid system is completely flexible, in the sense that the supervisor can select either of the adjacent partitions as the transition destination. Using this flexibility, as a next stage, we are going to implement more complex scenarios within the symbolic control framework to achieve more complicated specifications expressed in temporal logics such as LTL or CTL.

10. Conclusion

In this paper, we proposed a new approach of hybrid supervisory control for the leader follower formation problem. The approach was based on the polar partitioning of the state space. Assuming that the dynamics of the path planner of the UAV is in the form of $\dot{x} = u$, several multi-affine feedback controllers were designed to keep the system inside a partitioning element or drive it out through the desired direction. These multi-affine feedback controllers were then used to establish a hybrid controller that makes the controlled system able to reach the formation, starting from any arbitrary initial position inside the control horizon. In addition, an inter-collision avoidance mechanism was embedded in the controller in a modular way. After reaching the formation, the supervisor will keep the obtained configuration. The switching between different modes of operation were properly handled by the supervisor and it can be guaranteed that the system will reach the final states due to the bisimilarity relation between the abstracted system and the original plant in the proposed abstraction approach. Moreover, the velocity bounds were applied through the design procedure so that the generated velocities can be followed by the lower levels of control hierarchy. The control structure is decentralized, that is, the controllers for the leader and the follower act locally. Furthermore, the control design procedure was modular, i.e., we designed the controller for the formation control and collision avoidance separately and then, they have been combined using parallel composition operation, to achieve the whole controller and to satisfy the general specification.

Acknowledgement

The financial supports from TDSI and TL@NUS are gratefully acknowledged.

References

[1] Fierro R, Song P, Das A, Kumar V. Cooperative control of robot formations. In: Cooperative control and optimization. Series on applied optimization, vol. 66. US: Springer; 2002. p. 79–93.
[2] Lee G, Chong NY. Decentralized formation control for small-scale robot teams with anonymity. Mechatronics 2009;19(1):85–105.
[3] Pereira GAS, Kumar V, Campos MFM. Formation control with configuration space constraints. In: Proceedings of inernational conference on intelligent robots and systems, Las Vegas, 2003. p. 2755–60.
[4] Vachon MJ, Ray RJ, Walsh KR, Ennix K. F/A-18 performance benefits measured during the autonomous formation flight project. NASA technical report, no. NASA-TM-2003-210734, NASA Dryden Flight Research Center; 2003.
[5] Marconi L, Naldi R. Aggressive control of helicopters in presence of parametric and dynamical uncertainties. Mechatronics 2008;18(7):381–9.
[6] Metni N, Hamel T. A UAV for bridge inspection: visual servoing control law with orientation limits. J Automat Constr 2007;17(1):3–10.
[7] Ahmadzadeh A, Buchman G, Cheng P, Jadbabaie A, Keller J, Kumar V, Pappas G. Cooperative control of UAVs for search and coverage. In: Proceedings of the AUVSI conference on unmanned systems, 2006.
[8] Doherty P, Rudol P. A UAV search and rescue scenario with human body detection and geolocalization. In: Advances in artificial intelligence. Lecture notes in computer science, vol. 4830/2007. Berlin/Heidelberg: Springer; 2007. p. 1–13.
[9] Sasa S, Matsuda Y, Nakadate M, Ishikawa K. Ongoing research on disaster monitoring uav at jaxas aviation program group. In: SICE annual conference, 2008. p. 978–81.
[10] Bryson M, Sukkarieh S. Cooperative localisation and mapping for multiple UAVs in unknown environments. In: Proceedings of 2007 IEEE aerospace conference, 2007. p. 1–12.
[11] Puri A, Valavanis K, Kontitsis M. Statistical profile generation for traffic monitoring. In: Using real-time UAV based video data, 2007 mediterranean conference on control and automation, 2007. p. 1–6.
[12] Tian J, Shen L, Zheng Y. Formulation and a MOGA based approach for multi-UAV cooperative reconnaissance. In: Cooperative design, visualization, and engineering. Lecture notes in computer science, 4101/2006. Berlin/Heidelberg: Springer; 2006. p. 99–106.
[13] Tian X, Bar-Shalom Y, Pattipati KR. Multi-step look-ahead policy for autonomous cooperative surveillance by uavs in hostile environments. In: 47th IEEE conference on decision and control, 2008. p. 2438–43.
[14] Hassan G, Yahya K, ul Haq I. Leader-follower approach using full-state linearization via dynamic feedback. IEEE Int Conf Emer Technol 2006:297–305.
[15] Giuliettia F, Innocentib M, Napolitanoc M, Pollini L. Dynamic and control issues of formation flight. J Aerosp Sci Technol 2005;9(1):65–71.
[16] Stipanovic DM, Inalhan G, Teo R, Tomlin C. Decentralized overlapping control of a formation of Unmanned Aerial Vehicles. Automatica 2004;40(8):1285–96.
[17] Leonard N, Fiorelli E. Virtual leaders, artificial potentials and coordinated control of groups. In: Proceedings of the 40th IEEE conference on decision and control 3, 2001. p. 2968–73.
[18] Koditschek D, Rimon E. Robot navigation functions on manifolds with boundary. J Adv Appl Math 1990;11(4):412–42.
[19] Mukai M, Azuma T, Fujita M. A collision avoidance control for multi-vehicle using PWA/MLD hybrid system representation. In: Proceedings of the 2004 IEEE international conference on control applications, vol. 2, 2004. p. 872–7.
[20] Richards A, How J. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In: Proceedings of the 2002 American control conference, vol. 3, 2002. p. 1936–41.
[21] Wang X, Yadav V, Balakrishnan S. Cooperative UAV formation flying with obstacle/collision avoidance. IEEE Trans Control Syst Technol 2007;15(4):672–9.
[22] Antsaklis PJ, Kohn W, Lemmon MD, Nerode A, Sastry S, editors. Hybrid systems V. Lecture notes in computer science, vol. 1567. Springer; 1999.
[23] Bayraktar S, Fainekos G, Pappas G. Experimental cooperative control of fixed-wing Unmanned Aerial Vehicles. In: 43rd IEEE conference on decision and control, vol. 4, 2004. p. 4292–8.
[24] Zelinski S, Koo T, Sastry S. Hybrid system design for formations of autonomous vehicles. In: 42nd IEEE conference on decision and control, vol. 1, 2003. p. 1–6.
[25] Chen Q, Ozguner U. A hybrid system model and overlapping decomposition for vehicle flight formation control. In: 42nd IEEE conference on decision and control, vol. 1, 2003. p. 516–21.
[26] Karimoddini A, Lin H, Chen BM, Lee TH. Developments in hybrid modeling and control of Unmanned Aerial Vehicles. In: Proceedings of the 7th IEEE international conference on control and automation, Christchurch, New Zealand, December 2009. p. 228–33.

[27] Alur R, Henzinger TA, Lafferriere G, Pappas GJ. Discrete abstractions of hybrid systems. Proc IEEE 2000;88(7):971–84.
[28] Lafferriere G, Pappas GJ, Yovine S. A new class of decidable hybrid systems. Lecture notes in computer science, vol. 1569. Springer-Verlag; 1999. p. 137–51.
[29] Lafferriere G, Pappas GJ, Sastry S. O-Minimal hybrid systems. J Math Control, Signals, Syst 2000;13:1–21.
[30] Henzingera TA, Kopkeb PW, Puria A, Varaiya P. What's decidable about hybrid automata? J Comput Syst Sci 1998;57(1):94–124.
[31] Belta C, Habets L. Constructing decidable hybrid systems with velocity bounds. In: 43rd IEEE conference on decision and control, vol. 1, 2004. p. 467–72.
[32] Belta C, Habets L. Controlling a class of nonlinear systems on rectangles. IEEE Trans Autom Control 2006;51(11):1749–59.
[33] Tabuada PLP, Pappas GJ. Feasible formations of muti-agent systems. In: Proceedings of the American control conference, 2001. p. 56–61.
[34] Karimoddini A, Cai G, Chen BM, Lin H, Lee TH. Hierarchical control design of a UAV helicopter. In: advances in flight control systems. ISBN: 978-953-7619-X-X, INTECH, in press.
[35] Karimoddini A, Cai G, Chen BM, Lin H, Lee TH. Multi-layer flight control synthesis and analysis of a small-scale Uav helicopter. In: Proceedings of the 4th IEEE international conference on robotics, automation and mechatronics, Singapore, 2010. p. 321–6.
[36] Ramadge P, Wonham W. The control of discrete event systems. Proc IEEE 1989;77(1):81–98.
[37] Cassandras CG, Lafortune S. Introduction to discrete event systems. Springer; 2008.
[38] Kumar R, Garg VK. Modeling and control of logical discrete event systems. The Springer international series in engineering and computer science, vol. 300. Springer; 1995.
[39] Cai G, Lin F, Chen BM, Lee TH. Systematic design methodology and construction of UAV helicopters. Mechatronics 2008;18(10):545–58.
[40] Dong M, Chen BM, Cai G, Peng K. Development of a real-time onboard and ground station software system for a UAV helicopter. AIAA J Aerosp Comput, Inform, Commun 2007;4(8):933–55.