

HYBRID FORMATION CONTROL OF UNMANNED HELICOPTERS

ALI KARIMODDINI

NATIONAL UNIVERSITY OF SINGAPORE

2012

HYBRID FORMATION CONTROL OF UNMANNED HELICOPTERS

ALI KARIMODDINI

(M.Sc., Petroleum University of Technology, Iran)

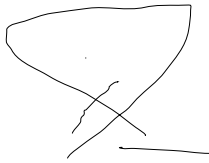
A THESIS SUBMITTED
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
NUS GRADUATE SCHOOL FOR INTEGRATIVE SCIENCES AND
ENGINEERING
NATIONAL UNIVERSITY OF SINGAPORE

2012

Declaration

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

A handwritten signature in black ink, consisting of a large, stylized loop followed by a cross-like shape and a horizontal line extending to the right.

Ali Karimoddini

06/02/2012

“To my parents, parents in law, beloved wife, son, brothers, sisters
and all relatives, friends and teachers for their support, care, and encouragement
during this journey.”

Acknowledgements

First and foremost, I would like to gratefully thank my supervisors Professor T. H. Lee, Professor Hai Lin, and Professor Ben. M. Chen for their great supervision, patience, encouragement and kindness. Without their guidance, this thesis would not have been possible. Moreover, I gratefully thank Professor Panos Antsaklis for supervising me during my staying in the Departments of Electrical Engineering, in the University of Notre Dame as a visiting student. I also thank Professor Kai-Yew Lum and Dr. Chang Chen for their valuable comments during my oral qualifying exams. I would also thank all lecturers in NGS and ECE Department and former teachers who have built my academic background, and all NGS, ECE and NUS staff and laboratory officers for their official supports.

Special thanks are given to the friends and fellow classmates in our UAV research group in the Department of Electrical and Computer Engineering, National University of Singapore. In particular, I would like to thank Dr. Kemao Peng, Dr. Guowei Cai, Dr. Lin Feng, Dr. Biao Wang, Dr. Miaobo Dong, Dr. Biao Wang, Dr. Ben Yu, and my fellow classmates Mr. Xiangxu Dong, Ms. Xiaolian Zheng, Mr. Fei Wang, Mr. Ang Zong Yao, Mr. Jinqiang Cui, Mr. Swee King Phang , Mr. Shiyu Zhao, and Ms. Jing Lin. I had also great time with my friends and fellow classmates in the Hybrid research group in the Department of Electrical and Computer Engineering, National

University of Singapore, especially Dr. Mohammad Karimadini, Mr. Mohsen Zamani, Mr. Alireza Partovi, Ms. Sun Yajuan, Prof. Liu Fuchun, Dr. Yang Yang, Ms. Li Xiaoyang, Mr. Liu Xiaomeng, Ms. Xue Zhengui, Mr. Yao Jin and Mr. Mohammad Reza Chamanbaz.

I would also thank my parents (Mr. Mohammad Mehdi and Ms. Mones) and parents in law (Mr. Mohammad and Ms. Fatemeh), my beloved wife (Najmeh), my son (Kevin), my elder brother and his wife (Mohammad and Atefeh) and all relatives and friends for their support, care, and encouragement during this journey.

Contents

Declaration	i
Acknowledgements	iii
Summary	ix
List of Figures	xii
1 Introduction	1
1.1 Motivation and Background	2
1.2 Existing Works	5
1.2.1 Hybrid Modelling and Control of a Single UAV	5
1.2.2 Hybrid Control for the Formation of the UAVs	8
1.3 Organization of the Thesis	13
2 Modelling and Control Design of a Unmanned Helicopter	16
2.1 Introduction	16
2.2 Testbed Infrastructure	19
2.3 Modeling and Structure of the UAV Helicopter	21
2.4 Controller Design	26

2.4.1	Designing the Controller for Subsystem 1	29
2.4.2	Designing the Controller for Subsystem 2	37
2.5	Experimental Results	43
2.6	Conclusion	44
3	Hybrid Modeling and Control of an Unmanned Helicopter	50
3.1	Introduction	50
3.2	The Regulation Layer	52
3.2.1	Velocity Control Mode	52
3.2.2	Position Control Mode	53
3.2.3	Hybrid Model of the Regulation Layer	54
3.3	Coordination Layer	57
3.4	Supervision Layer	59
3.5	The Composed Hybrid System	60
3.6	Implementation and Experimental Results	64
3.7	Conclusion	67
4	Hybrid Formation Control of Unmanned Helicopters	70
4.1	Introduction	70
4.2	Problem Formulation	71
4.3	Polar Abstraction of the Motion Space	74
4.3.1	Polar Partitioning	74

4.3.2	Properties of Multi-affine Functions over the Partitioned Space	78
4.3.3	Control over the Partitioned Space	82
4.3.4	Abstraction of the Motion Space	88
4.4	Hybrid Supervisory Control of the Plant	92
4.4.1	DES Model of the Plant	92
4.4.2	Design of the Supervisor	95
4.5	Simulation Results	101
4.6	Extension of the Algorithm to a 3-D Space	104
4.6.1	Spherical Partitioning	105
4.6.2	Control over the Spherical Partitioned Space	107
4.6.3	Designing the Supervisor for a Formation Mission over a the Spherically Partitioned Space.	110
4.6.4	Simulation Results	114
4.7	Conclusion	117
5	Implementation Issues and Flight Test Results for the Proposed Hy- brid Formation Algorithm	119
5.1	Introduction	119
5.2	Hierarchical Control Structure for the Formation Control	121
5.2.1	The Interface Layer	122
5.2.2	Applying the Discrete Supervisor to the Continuous Plant via the Interface Layer	124

5.3	Implementation Issues	125
5.3.1	Time Sequencing of the Events	125
5.3.2	Smooth Control over the Partitioned Space	126
5.4	Implementation Results	129
5.5	Conclusion	140
6	Conclusions	141
	Bibliography	146
7	APPENDIX	160
7.0.1	Proof for Theorem 4	160
	List of Publications	163

Summary

Nowadays, the cooperative control of multiple Unmanned Aerial Vehicles (UAVs) has emerged as an attractive research area, due to the rising demands from both military and civilian applications. Although a cooperative team of UAVs provides a more flexible and robust structure and reduces the overall costs, it poses significant theoretical and practical challenges. One of the main concerns is how to integrate coordination and supervision logical rules into the low level continuous control of team members, and how to deal with this essential hybrid nature of the system. To address this problem, traditional approaches treat the discrete and continuous dynamics of the system in a decoupled way and organize a two-layer control structure in which the low layer is responsible for generating continuous control signals based on the continuous dynamics of the system while the higher layer is responsible for managing the system to respect the desired logical rules. This control structure although simplifies the design, but the ignorance of the coupling effect between the discrete and continuous dynamics of the system is questionable whereas for the UAV systems it is crucial that a very reliable control system be provided. This calls for a comprehensive analysis of the system which can capture the interplay between the discrete supervisory logic and the continuous dynamics of the system within a unified framework. A proper solution for such a purpose is hybrid modelling and control framework.

This thesis aims to develop a hybrid supervisory control framework for the formation of unmanned helicopters. Building such a control structure can be divided into two main steps. The first step is to provide a hybrid model and controller for a single UAV to capture the interplay between the manoeuvre switching logic and the corresponding continuous dynamics under each control mode. Then, in the second step, a hybrid framework can be provided for the formation of team of UAVs based on their individual hybrid model.

Hence, starting with a single UAV helicopter, its hardware structure and dynamic model are explained and a control system is provided for the helicopter which makes the UAV able to follow the given references. Then, exploring the application of hybrid modelling and control theory, a hierarchical hybrid structure for a single UAV helicopter is proposed which has three layers: the regulation layer, the coordination layer, and the supervision layer. For each layer, a separate hybrid controller is developed. Then, a composition operator is adopted to capture the interactions between these layers. The resulting closed-loop system can flexibly command the UAV to perform different tasks, autonomously. The designed controller is embedded in the avionic system of the NUS UAV helicopter, and actual flight test results are presented to demonstrate the effectiveness of the proposed control structure.

In the next step, a hybrid supervisory control framework is provided for the formation of unmanned helicopters. Formation is a typical cooperative task and generally consists of three main parts: reaching the formation, keeping the formation, and collision avoidance. Using the proposed approach, all of these subtasks are addressed

within a unified framework. First, a new method of abstraction based on polar partitioning of the space is introduced. Then, utilizing the properties of multi-affine functions, the original continuous system with infinite states is bisimilarly converted to a finite state machine. Using the well developed theory of discrete event systems (DES) a discrete supervisor is designed for all of the subtasks of the formation in a modular way. The bismulation relation between the abstracted model and the original model is proven which guarantees that the discrete supervisor can be applied to the original plant while the closed loop system exhibit behaviours similar to the case that the discrete supervisor was applied to the abstracted model of the plant. In this case, an interface layer is required to link the discrete supervisor to the continuous plant. This interface layer, on the one hand is responsible to convert continuous signals of the plant to some discrete symbols understandable by the discrete supervisor, and on the other hand, it should convert the discrete commands of the supervisor to continuous signals applicable to the continuous plant. The results then are extended to the 3-dimensional case using spherical abstraction instead of polar partitioning of the space. Furthermore, implementation issues for the proposed algorithm are investigated and a smooth control mechanism is provided. Finally, several flight tests are conducted to verify the proposed algorithm.

List of Figures

1.1	Autonomy roadmap [1].	3
1.2	A supervisor to control a UAV for a search mission [2].	6
1.3	Hierarchical hybrid architecture of a UAV helicopter.	9
1.4	Hybrid supervisory control scheme based on polar and spherical abstraction.	12
2.1	NUS Cooperative UAV test-bed.	17
2.2	Schematic diagram of the flight control system	27
2.3	Control schematic for Subsystem 1.	28
2.4	Control schematic for Subsystem 2.	28
2.5	Simulation of the inner-loop of Subsystem 1.	32
2.6	Control structure of Subsystem 1.	32
2.7	Redrawing the control structure of Subsystem 1.	33
2.8	Characteristic loci of G_{in_1}	33
2.9	Robust system diagram.	35
2.10	Redrawing the Subsystem 1 for robust analysis.	35
2.11	Simulation of the outer-loop of Subsystem 1.	37

2.12	Simulation of the inner-loop of Subsystem 2.	39
2.13	Control diagram of Subsystem 2.	40
2.14	Bode plot of entries of G_{in_2}	41
2.15	Redrawing the control diagram of Subsystem 2.	42
2.16	Simulation of the outer-loop of Subsystem 2.	42
2.17	State variables of the UAV for the hovering.	45
2.18	UAV position in $x - y$ plane at hovering	46
2.19	Control signals at hovering	47
2.20	Tracking a desired path.	48
2.21	Circle path tracking in $x - y$ plane.	48
2.22	States of the UAV in the circle path tracking behavior.	49
2.23	Control inputs in the circle path tracking behavior.	49
3.1	Hierarchical hybrid control structure of an autonomous UAV Helicopter.	51
3.2	The controller for the velocity-control of the UAV.	53
3.3	The graph representation of the hybrid automaton H_R for the regulation layer.	54
3.4	The graph representation of the automaton H_{S_1} as the hybrid model for supervision layer for a mission with successive tasks.	60
3.5	Input and output channels for two composed systems.	62
3.6	The layers of the control hierarchy.	64
3.7	The composed system.	65

3.8	State variables of the UAV in a mission with successive tasks.	66
3.9	Control signals of the UAV in a mission with successive tasks.	67
3.10	(a) Zigzag Path Tracking (b) Circle Path Tracking (c) Velocity Control.	68
4.1	Control Structure of the UAV.	73
4.2	Relative frame; the follower should reach the desired position starting from any point inside the control horizon.	73
4.3	Partition labels.	75
4.4	Vertices and Edges in the element $R_{i,j}$	75
4.5	Vertices of the element $R_{i,j}$	76
4.6	Outer normals of the element $R_{i,j}$	77
4.7	$R_{1,j}$ is a special case of the element $R_{i,j}$	77
4.8	Invariant region.	83
4.9	Exit edge.	85
4.10	DES model of the plant.	94
4.11	Realization of reaching and keeping the formation specification.	98
4.12	Realization of the specification for collision avoidance.	100
4.13	The closed loop system.	100
4.14	Simulation of the system for an initial state inside the region $R_{4,1}$	102
4.15	Generated velocities V_x and V_y for an initial state inside the region $R_{4,1}$	102
4.16	Absolute distance from the desired position.	103
4.17	Collision avoidance mechanism.	103

4.18	(a) The partitioned sphere (b) Vertices, edges, and facets of the element $R_{i,j,k}$, and (c) Outer normal vectors of the element $R_{i,j,k}$.	106
4.19	DES model of a spherically partitioned plant.	111
4.20	The realization of reaching and keeping the formation specification.	113
4.21	The realization of collision avoidance specification, K_C .	113
4.22	The closed loop system.	115
4.23	The position of the UAV for the collision avoidance mechanism.	116
4.24	The relative distance between the follower and the desired position for the collision avoidance mechanism projected onto x-y plane.	117
4.25	The position of the UAVs in a circle formation mission.	117
5.1	Linking the discrete supervisor to the plant via an interface layer.	121
5.2	The control value at the vertices while transiting through the regions.	127
5.3	The schematic of the scenario with a real follower and a virtual fixed leader.	129
5.4	The state variables of the follower.	130
5.5	Control signals of the follower UAV.	130
5.6	The leader position in the relative frame.	131
5.7	The schematic of the scenario for a leader-follower case for tracking a line.	131
5.8	The position of the UAVs in the x-y plane.	132
5.9	The state variables of the follower.	133

5.10	Control signals of the follower UAV.	133
5.11	The state variable of the leader.	134
5.12	The distance of the follower from the desired position.	134
5.13	The schematic of the scenario for a leader-follower case for tracking a circle.	135
5.14	The position of the UAVs in the x-y plane.	136
5.15	The state variables of the follower.	136
5.16	Control signals of the follower UAV.	137
5.17	The state variables of the leader.	137
5.18	Control signals of the leader UAV.	138
5.19	The distance of the follower from the desired position.	138
5.20	The position of the UAVs.	139
5.21	The relative distance between the UAVs projected onto x-y plane. . .	139

Chapter 1

Introduction

Over recent years, the control of Unmanned Aerial Vehicles (UAVs) has emerged as a hot research area and have gained much attention in the academic and military communities [3], [4]. This is due to the fact that UAVs are not subjected to the limitations of ground robots like movement constraints and vision range limitations and hence, they have been found as proper solutions for different missions such as terrain and utilities inspection [5], search and coverage [6], search and rescue [7], disaster monitoring [8], aerial mapping [9], traffic monitoring [10], reconnaissance mission [11], and surveillance [12]. Among the UAVs, unmanned helicopters are of particular interest due to their unique features and capabilities such as vertical tacking off and landing, fixed-point hovering, flying at low level altitude, and great maneuverability.

Along with the developments of aerial robots, one of the main challenges is to improve the capabilities of UAVs to be able to autonomously involve in cooperative scenarios. Indeed, a team of robots, taking a cooperative structure, is more robust against the failures in team members or in communication links [13], [14]. Subjected to a proper cooperative tasking [15], [16], the use of several simpler robots instead of a complex one, results in a more powerful, flexible structure and improves the team

efficiency.

Formation control is a typical cooperative task in which several agents move with a relatively fixed distance [13], [17], [18]. Formation of Unmanned Aerial Vehicles (UAVs) can leverage the capabilities of the team to have more effective performance in missions such as cooperative SLAM, coverage and reconnaissance, security patrol, and etc. They can also mutually support each other in a hostile or hazardous environment [19], [20], [21].

1.1 Motivation and Background

In the literature there exist many works on the control of unmanned aerial vehicles. Nevertheless, most of these works focus on the low level performance of UAVs rather than satisfying high level specifications and incorporating the decision making unit into their control loops. Hence, development of autonomous aerial robots attracted worldwide academic and military communities. For example, in a recent road map published by the Department of Defense of United States of America (DOD), improving the autonomy level of UAVs is considered as one of the main challenges that need to be addressed for the next two decades [1]. To have a higher autonomy level and to reduce human interactions, this report then calls for research works on challenges such as robust decision making for individual UAVs and autonomous cooperative control for team of UAVs (Fig. 1.1). Along with these practical and theoretical demands, this thesis aims to develop a formal hierarchical hybrid control framework for unmanned helicopters to make them able to perform different missions autonomously. A typical

mission is composed of several tasks, for which separate controllers are required to be designed. Then, a decision making unit needs to be embedded to coordinate the controllers based on assigned tasks. Hence, the control structure of a UAV has a hybrid nature which includes both continuous and discrete dynamics that interactively coexist in the system [22]. To simplify the design, the discrete and continuous dynamics of the UAVs are usually treated in a decoupled way [23], [24]. However, ignoring the coupling effect between discrete and continuous dynamics of the system degrades the reliability of the overall system and may cause unexpected failures. As a dramatic example, in [25], it has been explained that focusing on embedded computer programs and negligence of the mutual relation between the discrete part and continuous dynamics of the system ended with the crash of Ariane 5 on June 4, 1996.

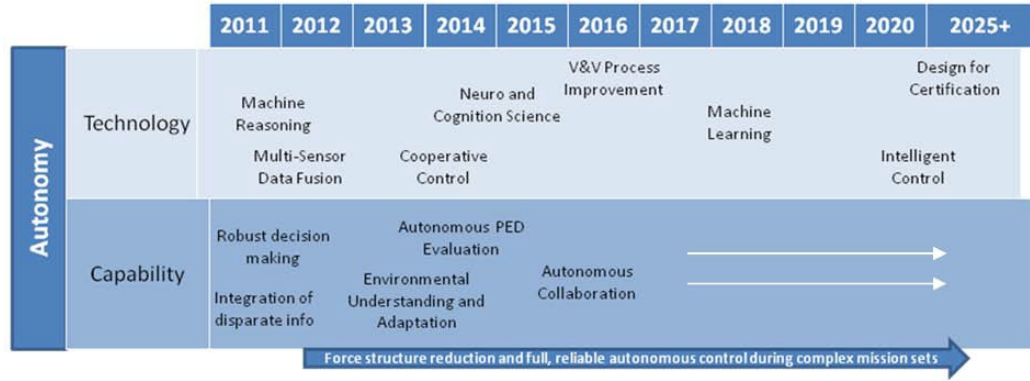


Figure 1.1: Autonomy roadmap [1].

Turning to the cooperative control of team of UAVs, the problem becomes even more complicated. The coordination of multiple UAVs involves a lot of issues such as handling interactions between UAVs, locally controlling each UAV while satisfying

the global goals, applying high level supervisory logical rules. For instance, formation of UAVs as a cooperative task, consists of several subtasks. 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 (*collision avoidance*). To address this problem, similar to the single UAV case, a usual practice is to separately design controllers for each of these subtasks, and then, a decision making unit is needed to coordinate these subcontrollers to achieve the team goal. Although with this method the design procedure has been simplified, success in cooperative control of multiple UAVs require an in-depth understanding of the interplay between the UAVs' continuous dynamics and their supervisory logic.

Hence, we are motivated to propose a congenial control mechanism for unmanned helicopters based on the hybrid modelling and control theory [26], [27], [28], [29]. Hybrid systems refer to a class of complex systems that involve interacting event-triggered discrete logic and time-triggered continuous dynamics. Such kind of system are usually resulted from the integration of logic-decision components with the continuous dynamics and constraints of the system. Within hybrid framework, there are effective tools for mathematical representation and analysis of variety of applications ranging from manufacturing and chemical process to robotics and aerospace control [30], [31], [32], [33]. Next we will briefly review some of the existing results on the hybrid modelling and control of the UAVs.

1.2 Existing Works

1.2.1 Hybrid Modelling and Control of a Single UAV

Several research groups are involved in the modeling and control of UAVs [34], [5], [35]. However, the efforts to use hybrid modelling and control theory approaches in UAV studies are relatively sparse and just recent [36]. Therefore, although a UAV can be naturally seen as a hybrid system, hybrid modelling and control of UAVs is still in its infancy and poses many technical and theoretical challenges. So far, most of the existing works either focus on the continuous evolution of the system [37], [38], concern with the discrete nature of the decision making system [23], [24], [2], or model both the discrete dynamics and continuous dynamics but in a decoupled way [39]. For instance, in [2], a UAV platform has been developed for a search mission in which for the top level of the controller they have implemented a DES supervisor to control the flight modes (Fig. 1.2). In this control structure, once the UAV has arrived at the goal point, the UAV starts image processing services to attempt identification of vehicles in the area. If a vehicle matching the initial signature is found, the UAV starts a new FlyTo mission which uses a proportional navigation controller. Here, the supervisor is purely discrete and it is designed to be independent of the UAV dynamics.

To explore the applications of hybrid theory in the sophisticated structures of UAVs, in [40], a hybrid controller is developed for the control of the altitude and turning rate of a fixed wing UAV. The controller is composed of two separate and

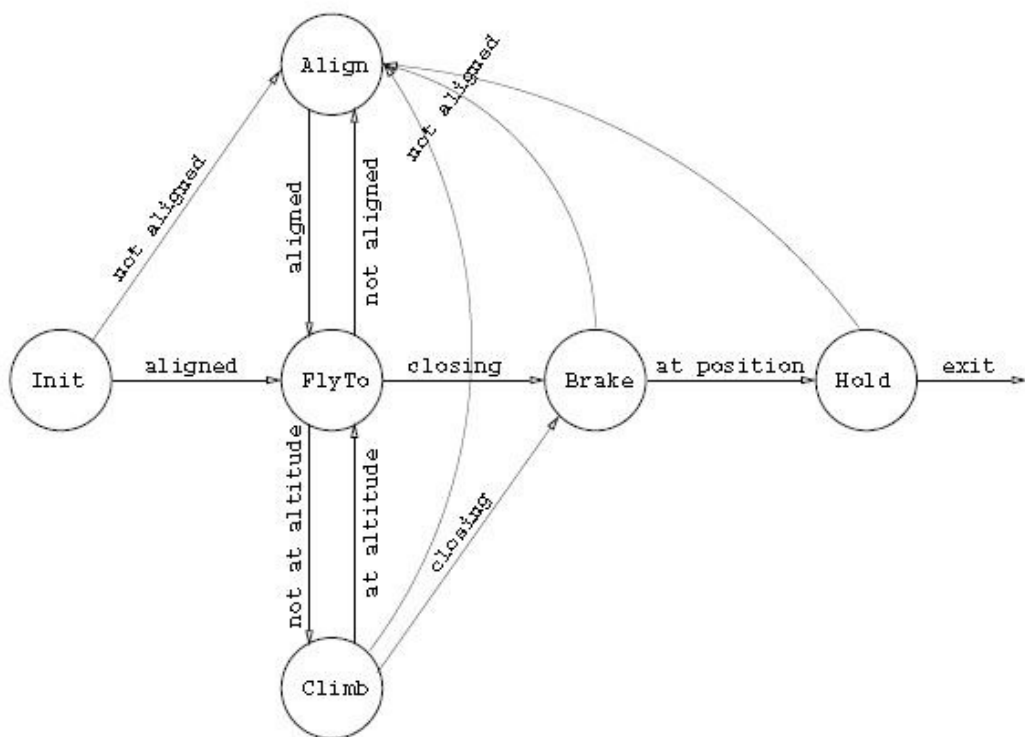


Figure 1.2: A supervisor to control a UAV for a search mission [2].

decoupled parts for the altitude and lateral control of the UAV, and the developed system is used for performing an aerial surveillance mission. For quadrotors, in [41], a hybrid model for the backflit maneuvering is provided for which a forward reachability analysis guarantees the switching sequence for the correct execution of the task. Similarly in [42], a robust reachability analysis is given for taking off and landing of a ducted-fan aerial vehicle. When the vehicle is landing, upon contacting with the ground, the control dynamics will be changed. So, the hybrid controller pushes the switching sequence to safely land on the ground. In [43], a hybrid model for the fuel consumption of a UAV is presented for which a safety specification is determined to be achieved by the designed controller, and the result is verified using the reachability analysis. Here, the safety property is naturally defined as reaching the objective area while having enough amount of fuel. The fuel consumption depends on the UAV mission and could vary with the speed changes of the UAV. As a safety property in the hybrid automaton of the fuel model, the "NOFUEL" mode should always be avoided. In [44], the path planning of a UAV helicopter is translated to a robust hybrid analysis problem and the results are verified through simulations. Using Mixed-Integer Linear Programming (MILP), it is able to convert a hybrid controller design problem into a smooth optimal control problem [45], [46]. In [45], an optimal hybrid control problem of UAVs with logical constraints has been transferred to some inequality and equality constraints involving only continuous variables. As another example, in [46], a hybrid controller for the velocity control of a helicopter is provided where Mixed Integer Linear Programming is used for the optimal reference generation.

Most of these works focus on a specific task, while still there is a need to develop

a hybrid model and controller for an autonomous UAV to be able to involve in different missions. Of course such a control system would be too complicated. Hence, to reduce the complexity of the system and to facilitate the design procedure, this thesis develops a hierarchical control structure in a systematic way to distribute the control tasks among the layers (Fig. 1.3). The use of hierarchical control and its application to coordination problems have been studied for a long time [47], [48], [49]; however, considering the concept of hierarchical control within hybrid framework and its application to autonomous systems still is a challenging problem. Moving towards this ambitious goal, this thesis proposes a formal hierarchical hybrid modelling and control approach for UAV systems. The proposed control system has three layers: the regulation layer which is responsible for the low level control; the coordination layer which is responsible for generating a path to be followed by the regulation layer, and the supervision layer which is the decision making unit and is responsible for managing the switching scenario to perform a mission, autonomously. Each layer has been modelled with an hybrid Input/Output automaton [50]. Then, a composition operator is introduced to synchronize the layers and capture the interplay between them.

1.2.2 Hybrid Control for the Formation of the UAVs

As it was mentioned, a formation mission has three main components: reaching the formation, keeping the formation, and collision avoidance. To address each of these tasks, many studies have been conducted in the literature. For reaching the forma-

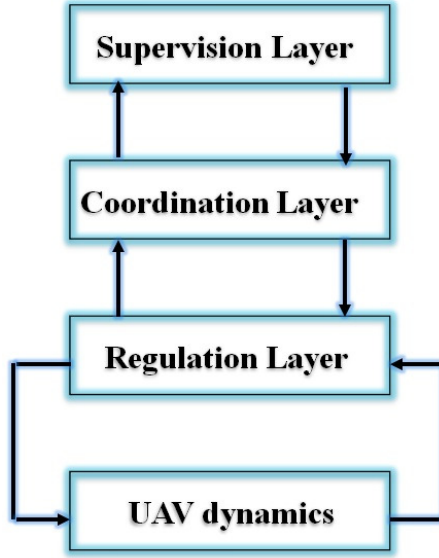


Figure 1.3: Hierarchical hybrid architecture of a UAV helicopter.

tion, there are several existing methods such as optimal control techniques, navigation function, and potential field [51], [52], [53]. Keeping the formation can be seen as a standard control problem in which the system's actual position has slightly deviated from the desired position for which many control approaches have been developed such as feedback control, rigid graph, and virtual structure [54], [55], [56], [57]. Finally, in [58], [59], and [60], different mechanisms for collision avoidance have been introduced using probabilistic methods, MILP programming, and behavioral control. Nevertheless, there is still a lack of a unified solution to address the whole process starting from reaching formation, maintaining formation while avoiding collision. To integrate all of the components of a formation mission and to capture the interactions between the subcontrollers, a proper solution is to take the advantages of the hybrid modelling and control theory.

Despite the great efforts being put in the study of hybrid supervisory control, its application to cooperative control problems and in particular in formation control problem has been hindered due to the challenges such as the existence of a finite bismulation relation, decidability issues, and etc. Nevertheless, in the literature, there are some results on the hybrid formation control which are developed for ground robots. For instance, in [61], a hybrid controller has been provided for a group of nonholonomic robots. The algorithm has two main modes for keeping the formation and obstacle avoidance. A switching strategy is provided and the stability of the overall system under the proposed switching scenario is investigated. In [62], a hybrid controller has been designed for the formation control of ground robots. The control structure has two layers by which the switching logic and the continuous low level control are separated so that the lower layer is responsible for the path tracking control of the robots and the top layer is a centralized supervisor which is responsible for decision making to manage the formation.

For the hybrid formation control of aerial robots, the results are less due to the complexity of their model and difficulties on the development of cooperative testbeds of aerial vehicles. In [40], after developing a hybrid model of a single UAV, a formation control has been implemented for two fixed wing UAVs. In [63], the formation reconfiguration problem has been addressed for a group of UAVs. Several formation manoeuvres are considered and then, for switching from each manoeuvre to another one, an optimal path is generated and stored in the library of the system. A discrete supervisor decides which formation manoeuvre should be activated. In [64] and [65] firstly, using overlapping theorem [66], the authors have decomposed the

graph of flight formation into some disjoint triangular subgraphs and have obtained a control law for the formation control of each triangular subsystem. Then, they have contracted these triangles to obtain the original graph. In fact, dealing with formation of triangles as a basic unit of a flight formation is more rational than dealing with the formation of the whole graph.

Most of the above mentioned methods focus on hybrid modeling of the system rather than providing a hybrid analysis. Moreover, the discrete and continuous dynamics of the system are still treated in a decoupled way. To take the advantage of the hybrid analysis and synthesis tools, this thesis proposes a hybrid supervisory control framework for the formation control of unmanned helicopters (Fig. 1.4). First, a new method of abstraction based on polar and spherical partitioning of the state space will be introduced, by which the original continuous system with infinite states will be abstracted to a finite state machine. Then, for the resulting abstracted system, one can take the advantages of the well-developed theory of supervisory control of Discrete Event Systems (DES) [67], and modularly design the discrete supervisors for reaching the formation, keeping the formation, and avoiding collisions. If the abstracted system is bisimilar to the original continuous system, the discrete supervisor can be applied to the continuous plant while the closed-loop system exhibits behaviors similar to the case that the discrete supervisor was applied to the discrete model of the plant. To apply the discrete supervisor to the plant, inspiring from [26], an interface layer need to be constructed which on the one hand, it translates the continuous signals to discrete symbols understandable for the discrete supervisor, and on the other hand, it converts the discrete commands of the supervisor to continuous

signals applicable to the continuous plant.

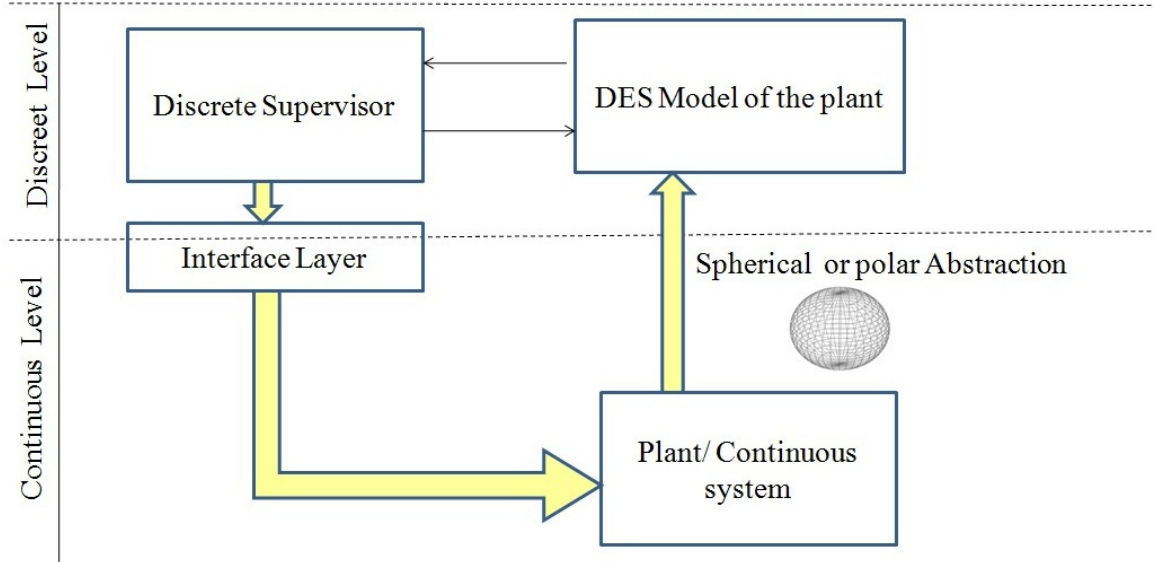


Figure 1.4: Hybrid supervisory control scheme based on polar and spherical abstraction.

Here, the key point is to develop an abstraction procedure so that at the end, the discrete model of the system and the original continuous system have a bismulation relation and exhibit the same behaviours. 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 [68], [69], [70], [71]. Recently, multi- affine vector fields, as a wider and more practical classes of hybrid systems, have been used as decidable systems under triangulation and rectangulization of the state space [72], [73]. In [73], a class of nonlinear systems has been abstracted using rectangular partitioning. In [74], it has been shown that an affine feedback over a simplex can be designed to

steer the system's trajectory to exit facets, and in [75] the method is extended to a reachability problem over a partitioned system whose elements are simplices. Despite the existing theoretical developments, so far, the use of these methods for practical robotic applications is still in its infancy, and in particular, these methods have not been used in the UAV path planning and formation control applications. Furthermore, formulating a formation problem within a rectangulized or traingulized 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 and spherical partitioning of the motion space, can be appropriately applied to the formation problem as we will discuss through the following chapters.

1.3 Organization of the Thesis

This thesis aims to develop a hybrid supervisory control framework for the formation of the UAVs. To address this problem, first it is required to model a single UAV helicopter within a hybrid framework and provide a reliable control for each of the agents involved in the formation mission. Then, a hybrid supervisory control mechanism will be developed for a team of UAV helicopters that are involved in a leader follower formation scenario. The organization of the dissertation is described as follows:

In chapter 2, the model of a UAV helicopter is discussed. Then a low-level controller for a UAV helicopter is designed which consists of two layers; the lower level that is responsible for maintaining the attitude of the UAV, and a higher level that is used to drive the UAV into the desired three-dimensional generated path in near-hover

conditions. With this control strategy, an H_∞ controller is used for the inner-loop to provide a robust stable suboptimal desired attitude, and a proportional feedback combined with a transformation block is used for the design of the outer-loop controller to compensate the system's nonlinearity and to drive the UAV to follow the desired trajectory. The proposed control structure is implemented on the NUS UAV test-bed and actual flight tests demonstrate the effectiveness of the design procedure.

Chapter 3 proposes a hybrid control structure for a single UAV helicopter. The developed hybrid controller has a hierarchical structure which consists of three layers: the regulation layer; the coordination layer; and the supervision layer. For each layer a separate hybrid model has been developed. To capture interactions between these layers and to synchronise them, a composition operator is introduced. Finally, the control structure has been implemented and several flight tests are conducted to evaluate the control performance of the proposed control structure.

In Chapter 4, a new approach of hybrid supervisory control of the UAVs is introduced which can be used 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, a new method of abstraction is introduced which uses the properties of multi-affine vector fields over a 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, a collision avoidance

mechanism has been considered in the controller structure. 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. The result is extended to a three-dimensional hybrid supervisory control for leader follower formation algorithm. To realize this controller, a spherical abstraction of the motion space is proposed and similar to the 2-d case, utilizing the properties of multi-affine functions over the partitioned space, a finite state Discrete Event System (DES) model is achieved which is bisimilar to the original partitioned system. For the obtained DES model of the plant, a supervisor has been developed to accomplish the formation mission.

Chapter 5 discusses some implementation issues and presents actual flight test results for the hybrid formation control of the UAVs, and finally Chapter 6 concludes the thesis and summarizes the contributions, and discusses some future works.

Chapter 2

Modelling and Control Design of a Unmanned Helicopter

2.1 Introduction

This chapter aims to describe the NUS cooperative testbed and develop proper model and controller for the individual members of this UAV team that are going to be used in the next chapters. Here, the testbed is a set of two similar UAV helicopters, HeLion and SheLion (Fig. 2.1) which are developed by our research group at the National University of Singapore.

Several research groups are involved in the modeling and control of UAVs [34], [5], [35]. The control methods such as the neural network approach [76], the differential geometry method [77], feedback control with decoupling approach [78], and the model predictive approach [79] have been applied for the flight control of the UAV helicopters. In this chapter, however, an analytical approach is used to design and analyze the whole system including the inner-loop and the outer-loop controllers for a small-scale UAV helicopter. Here, in the proposed multi-layer control structure, the inner-loop is responsible for the internal stabilization of the UAV in the hovering state and for the control of the linear velocities and heading angular velocity whereas



Figure 2.1: NUS Cooperative UAV test-bed.

the outer-loop is used to drive the UAV, which is already stabilized by the inner-loop, to follow a desired path while keeping the system close to the hovering state. This control strategy is an intuitive way of controlling such a complex system. However, there is another reason that compels us to employ such a control structure. Indeed, the UAV model cannot be fully linearized, since, in practice, the heading angle of the UAV could be in any direction and we cannot expect it to be restricted to a small range of variation. This will impose some kinds of nonlinearity on the system, which can be modeled by a simple transformation. To handle this semi-linearized model of the UAV, the linear and nonlinear parts are separately controlled in the inner-loop and the outer-loop.

In this control structure, for the inner-loop, an H_∞ controller is used to both stabilize the system and suboptimally achieve the desired performance of the UAV

attitude control. Assuming that the inner-loop has already been stabilized by an H_∞ controller, a proportional feedback controller combined with a transformation block have been used in the outer-loop to bring the UAV into the desired position with desired heading angle.

Although designing a proportional feedback controller for SISO systems is straightforward, the situation for MIMO systems is different. This is due to the fact that in MIMO systems, it is not easy to use the popular tools, such as the Nyquist stability theorem or the root-locus approach, that are well-established for SISO systems. The current approaches employed for MIMO systems are rather complicated and are mostly extensions of the existing results for SISO systems [80]. In this chapter, a design method of a decentralized P-controller for MIMO systems is introduced that although conservative, it can be effectively used in practical problems, particularly for the case that the system is close to a decoupled system. The approach is an extension of the Nyquist theorem to MIMO systems, and its application to the NUS UAV system provides a successful flight controlled system.

The remaining parts of this chapter are organized as follows. First, in Section 2.2 the developed cooperative testbed is explained which consists of two similar helicopters Helion and Shelion. Then, in Section 2.3, the model and structure of these helicopters are described. The model of these helicopter is composed of two decoupled subsystems for which, in Section 2.4, a two-layer controller, including an inner-loop and an outer-loop controller, is designed. Actual flight tests are presented in Section 2.5, and the chapter is concluded in Section 2.6.

2.2 Testbed Infrastructure

The members of this cooperative team are two radio-controlled bare helicopter, Raptor 90. The size of these helicopters is 1410 mm in length and 190 mm in width of the fuselage. The maximum takingoff weight is 11 kg including 5 kg as the dry weight of helicopter and 6 kg as the effective payload. Their main rotors and tail rotors have the diameter of 1,605 mm and 260 mm, respectively.

A typical UAV helicopter consists of several parts: physical parts such as engine and fuselage; ground station to monitor the flight situation and collect realtime flight data, and the avionic system to implement the control strategy to have an autonomous flight control.

Among these elements, the avionic system is in the center of our interest in this chapter and we will focus on the control structure which is embedded in its airborne computer system. Here, the avionic system consists of a a PC/104 ATHENA, as an onboard airborne computer system which has four RS-232 serial ports, a 16-pin digital to analog (D/A) port, two counters/ timers and runs at 600 MHz.

In addition, the avionic system has been equipped with some analog and digital sensors to collect the information of the current state of the UAV. The most important one is a compact fully integrated INS/GPS , NAV420, Crossbow, which is used for the navigation of the UAV and can provide three axis velocities, acceleration and angular rates in the body frame, as well as longitude, latitude, relative height and heading, pitch and roll angles. Moreover, the avionic system has a fuel level sensor as well as a magnetic RPM sensor to measure the speed of the rotor. Furthermore, it has four

servo actuators that could manipulate the helicopter to move forward and backward, up and down, to turn left and right, and to regulate the nose angle. All of these servos are controlled by a servo board as a local controller. In addition, the servo board gives the ability to put the servo system in either the manual mode or the automatic mode. In the manual mode, a pilot can drive the helicopter by a radio controller which is useful in the emergency situations; however, in the automatic mode the helicopter is under the control of the computer system and all control signals are generated by the avionic system and the computer board, autonomously.

For the reliable communication between the UAVs, and also between the UAVs and the ground station, we have used serial wireless radio modems, IM-500X008, FreeWave, with the working frequency of 2.4 GHz, which can cover a wide range up to 32 km in an open field environment.

The onboard program is implemented using QNX Neutrino real time operating system. The structure of this onboard program has utilized a multi-thread running scheme which includes several threads for flight control; reading from data acquisition board; driving the servo actuators; making dual-directional wireless communication with other UAVs or with the ground station; and logging data into an onboard compact flash card.

Furthermore, for these helicopters, a hardware-in-the-loop simulation software has been developed by integrating the developed hardware and embedded software together with the nonlinear dynamic model of the UAV helicopters. 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. Consequently, the simulation results of this simulator is very close to the actual flight tests, and it can provide a safe and reliable environment for the pre-evaluation of control algorithms.

2.3 Modeling and Structure of the UAV Helicopter

Using some basic physical principles, one can obtain a general nonlinear UAV model. These principles will result in several equations that represent the effects of different factors such as gravity, the main rotor, and tail rotor forces and moments. The model equations will be obtained in two coordinate systems: the body frame and the ground frame. The body frame is located at the center of gravity of the UAV, and the ground frame is an NED (North - East - Down) coordinate system [81] with a fixed origin at the starting point of the UAV flight. The moment and force equations in the UAV model must be derived in the body frame, whereas to obtain the net displacement of the UAV, we need to use the ground frame.

Neglecting the gyroscopic effect of the engine-driven train, the equations of the helicopter motion in the body frame are obtained as follows:

$$\dot{\vec{V}}_b = -\vec{\omega}_b \times \vec{V}_b + B_b \vec{g} + m^{-1} \vec{F} \quad (2.1)$$

$$\dot{\vec{\omega}}_b = -J^{-1} \omega_b \times J \vec{\omega}_b + J^{-1} \vec{M} \quad (2.2)$$

where in these equations, \times denotes the cross product of the vectors and concate-

nation of two matrices or vectors stands for normal matrix multiplication; $\vec{V}_b = (V_{xb}, V_{yb}, V_{zb})'$ and $\vec{\omega}_b = (\omega_{xb}, \omega_{yb}, \omega_{zb})'$ are the velocity and the angular velocity in the body frame, respectively; $\vec{g} = [0, 0, g]'$ is acceleration due to the earth gravity and g is assumed to be a constant; m is the mass of the helicopter; J is the inertia matrix of the aircraft, and \vec{F} and \vec{M} are the resultant force and moment in the body frame, including those generated from the main rotor, tail rotor and the fuselage.

The Euler angles that show the orientation of the body frame relative to the ground frame are as follows:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \tan \theta \sin \phi & \tan \theta \cos \phi \\ 0 & \cos \phi & -\sin \phi \\ 0 & \frac{\sin \phi}{\cos \theta} & \frac{\cos \phi}{\cos \theta} \end{bmatrix} \dot{\vec{\omega}} \quad (2.3)$$

where $(\phi, \theta, \psi)'$ is a vector containing the Euler angles which describe the attitude of the helicopter with respect to the NED frame.

The relation between the UAV position in the ground frame and the UAV velocity in the body frame is:

$$\dot{\vec{P}}_g = B_b' \vec{V}_b \quad (2.4)$$

where \vec{P}_g is the position of the UAV in the ground frame and B_b is the transformation matrix from the ground frame to the body frame:

$$B_b = \begin{bmatrix} \cos \theta \cos \psi & \cos \theta \sin \psi & -\sin \theta \\ -\cos \phi \sin \psi + \sin \phi \sin \theta \cos \psi & \cos \phi \cos \psi + \sin \phi \sin \theta \sin \psi & \sin \phi \cos \theta \\ \sin \phi \sin \psi + \cos \phi \sin \theta \cos \psi & -\sin \phi \cos \psi + \cos \phi \sin \theta \sin \psi & \cos \phi \cos \theta \end{bmatrix} \quad (2.5)$$

The details of this UAV model are described in [78]. This nonlinear model of the UAV is identified using in-flight data which was collected by injecting perturbed

input signals to the flying helicopter in the manual mode. Then, the achieved model is linearized at the hovering state in which $\vec{V}_b = 0$, $\vec{\omega}_b = 0$, $\theta = 0$, $\phi = 0$. The linearized model of the UAV is as follows:

$$\dot{x}_{in} = Ax_{in} + Bu + Ew \quad (2.6)$$

where $x_{in} = [V_{x_b}(m/s) \ V_{y_b}(m/s) \ \omega_{x_b}(rad/s) \ \omega_{y_b}(m/s) \ \phi(rad) \ \theta(rad) \ \tilde{a}_1(rad) \ \tilde{b}_1(rad) \ V_{z_b}(m/s) \ \omega_{z_b}(rad/s) \ w_{zf}(rad/s)]'$ is the internal state of the system. Here, V_{x_b} , V_{y_b} , and V_{z_b} are linear velocities; ω_{x_b} , ω_{y_b} , and ω_{z_b} are angular velocities; ϕ , θ , and ψ are Euler angles; \tilde{a}_1 and \tilde{b}_1 are flapping angles, and w_{zf} is the state variable of the rate gyro to describe the first order differential equation of δ_{pedal} [82]. Furthermore, the control input u includes commands to the servos embedded for the control of the helicopter blades as $u = [\delta_{pedal}(rad) \ \delta_{roll}(rad) \ \delta_{col}(rad) \ \delta_{pitch}(rad)]'$ where δ_{roll} and δ_{pitch} are cyclic commands, δ_{pedal} is the control input to the yaw channel servo, and δ_{col} is the pedal command. Finally, $w = (u_{wind}, v_{wind}, w_{wind})$ is the wind gust disturbance where u_{wind} , v_{wind} , w_{wind} affect the UAV velocities in the x, y and z directions, respectively. The state and input matrices A and B of the corresponding linearized model, and the disturbance matrix E are as follows:

$$A = \begin{bmatrix} A_2 & 0_{8 \times 3} \\ 0_{3 \times 8} & A_1 \end{bmatrix}, B = \begin{bmatrix} B_2 & 0_{8 \times 2} \\ 0_{3 \times 2} & B_1 \end{bmatrix}, E = \begin{bmatrix} E_2 & 0_{8 \times 1} \\ 0_{3 \times 2} & E_1 \end{bmatrix}$$

where

$$\begin{aligned}
A_1 &= \begin{bmatrix} -0.6821 & -0.1070 & 0 \\ -0.1446 & -5.5561 & -36.6740 \\ 0 & 2.7492 & -11.1120 \end{bmatrix}, \\
B_1 &= \begin{bmatrix} 15.6491 & 0 \\ 1.6349 & -58.4053 \\ 0 & 0 \end{bmatrix}, \\
E_1 &= \begin{bmatrix} -0.5995 \\ -1.3832 \\ 0 \end{bmatrix}, \quad B_2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0.0496 & 2.6224 \\ 2.4928 & 0.1740 \end{bmatrix}, \quad E_2 = \begin{bmatrix} -0.1778 & 0 \\ 0 & -0.3104 \\ -0.3326 & -0.2051 \\ 0.0802 & -0.2940 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \\
A_2 &= \begin{bmatrix} -0.1778 & 0 & 0 & 0 & 0 & -9.7807 & -9.7808 & 0 \\ 0 & -0.3104 & 0 & 0 & 9.7807 & 0 & 0 & 9.7807 \\ -0.3326 & -0.5353 & 0 & 0 & 0 & 0 & 75.7640 & 343.86 \\ -0.1903 & -0.2940 & 0 & 0 & 0 & 0 & 172.620 & -59.958 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & -8.1222 & 4.6535 \\ 0 & 0 & -1 & 0 & 0 & 0 & -0.0921 & -8.1222 \end{bmatrix}.
\end{aligned}$$

Remark 1 *In the linearized model described by (2.6), the saturation level for the servos is $|\delta_{\star_{max}}| = 0.5$. We need to provide a control law such that the resulting*

control signals always remain within the linear unsaturated range.

Although (2.6) describes the relation between the control input and the state variable x_{in} , it still does not describe the whole dynamics of the system, and particularly, the position, $\vec{P}_g = (x, y, z)'$ and the heading angle ψ are not reflected in the model. Thus, considering (2.4) and (2.5), a more complete model containing \vec{P}_g and ψ is as follows:

$$\begin{cases} \dot{x}_{in} = Ax_{in} + Bu + Ew \\ \dot{x}_{out} = \Omega'(\Theta)Cx_{in} \end{cases} \quad (2.7)$$

where $x_{out} = [x(m) \ y(m) \ z(m) \ \psi(rad)]'$. Here, x , y , and z describe the position of the UAV in the ground frame, ψ is its heading angle, and $\Theta = [\phi, \theta, \psi]^T$ is the orientation vector. Matrix C and the block $\Omega(\Theta)$ are as follows:

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \quad \Omega(\Theta) = \begin{bmatrix} B_b & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix},$$

Remark 2 Matrix B_b includes some time-dependent terms. Therefore, matrix B_b can not be considered as a constant term and it is not simple to integrate both sides of the second equation of (2.7) in order to obtain the position in the ground frame. This is due to the fact that the body frame is a moving coordinate system. Hence, to obtain the displacement, it is necessary to first obtain the velocities in a fixed coordinate system such as the ground frame. Then, the displacement can be calculated by integrating the velocity vector in the fixed coordinate system.

To make the design control problem easier, we can further simplify the model. Indeed, matrix B_b in (2.5), which introduces some nonlinear terms to the model, can be linearized at the hovering state. In practice, the heading angle of the helicopter can take any arbitrary value; however, the roll and pitch angles are usually kept close to the hovering condition. Therefore, linearizing matrix B_b at the hovering state will result in:

$$B_b = \begin{bmatrix} \cos \psi & \sin \psi & 0 \\ -\sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R & \mathbf{0}_{2 \times 1} \\ \mathbf{0}_{1 \times 2} & 1 \end{bmatrix} \quad (2.8)$$

The physical interpretation is that by keeping $\vec{\theta}$ and $\vec{\phi}$ close to zero, the Euler rotation in a three-dimensional space will be converted into a simple rotation in a two-dimensional space with respect to ψ . In this case, the rotation matrix is:

$$R = \begin{bmatrix} \cos \psi & \sin \psi \\ -\sin \psi & \cos \psi \end{bmatrix} \quad (2.9)$$

In the next section, the control structure for this model of the UAV will be described.

2.4 Controller Design

We will construct a two-layer controller for this UAV (Fig. 2.2). In this framework, the system is stabilized by the inner-loop, and then it is driven to track a desired trajectory by the outer-loop.

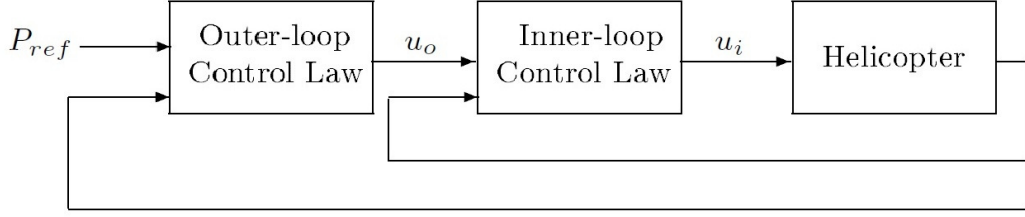


Figure 2.2: Schematic diagram of the flight control system

In this control architecture, the reference for the inner-loop controller is $u_i = [V_{x_b}, V_{y_b}, V_{z_b}, \omega_{z_b}]'$ where $(V_{x_b}, V_{y_b}, \text{ and } V_{z_b})$ are the linear velocity references, and ω_{z_b} is the yaw rate reference, which all should be provided by the outer-loop. The outer-loop, however, is responsible for the control of the position and heading angle of the UAV and will guide the UAV to follow a desired trajectory. Therefore, the references for the outer-loop are the position (X_r, Y_r, Z_r) and the yaw angle ψ_r . In other words, the UAV will follow the desired path by controlling the position and the yaw angle in the outer-loop, and the linear velocity and the angular rate in the inner-loop.

Looking at matrices A , B , and E in (2.6), we can see that, the model is a decoupled system with two independent parts. Therefore, (2.6) can be rewritten into two separate subsystems as follows:

$$\dot{x}_1 = A_1 x_1 + B_1 u_1 + E_1 w_1 \quad (2.10)$$

$$\dot{x}_2 = A_2 x_2 + B_2 u_2 + E_2 w_2 \quad (2.11)$$

where $x_1 = [V_{z_b}(m/s) \ \omega_{z_b}(rad/s) \ w_{zf}(rad/s)]'$, $u_1 = [\delta_{col} \ \delta_{pedal}]'$, $x_2 = [V_{x_b}(m/s) \ V_{y_b}(m/s) \ \omega_{x_b}(rad/s) \ \omega_{y_b}(rad/s) \ \phi(rad) \ \theta(rad) \ \tilde{a}_s(rad) \ \tilde{b}_s(rad)]'$, and $u_2 = [\delta_{roll}(rad)$

$\delta_{pitch}(rad) \]'$.

Considering (2.7), (2.8), (2.10), and (2.11), the control structures for these two subsystems are shown in Fig. 2.3 and Fig. 2.4. In these figures, subscripts g , b , and r stand for ground frame, body frame, and reference, respectively. Moreover, matrices C_1 and C_2 are:

$$C_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, C_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (2.12)$$

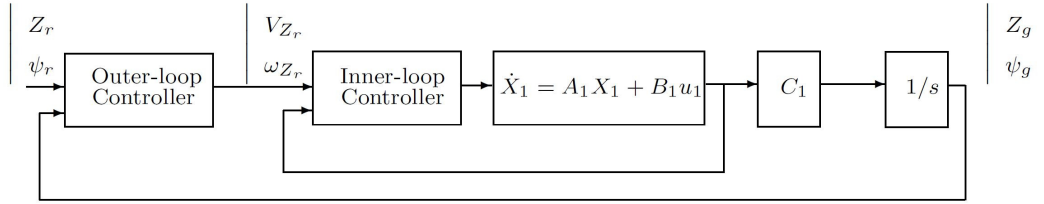


Figure 2.3: Control schematic for Subsystem 1.

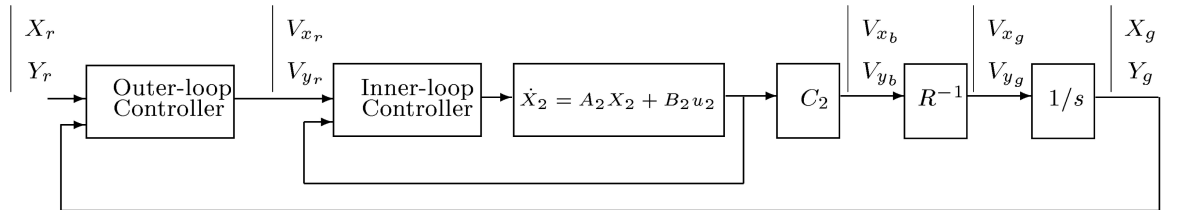


Figure 2.4: Control schematic for Subsystem 2.

Due to the special structure of the linearized form of B_b , Subsystem 1 is a fully linearized model. However, in the outer-loop of Subsystem 2, the term R^{-1} appears as a nonlinear element, and thus Subsystem 2 is more complicated than Subsystem 1. In the following parts, we will describe the control design for both subsystems.

2.4.1 Designing the Controller for Subsystem 1

Subsystem 1 is a fully linearized model and hence, we can take the advantages of linear design tools. We will use the H_∞ control design technique for the inner-loop and a P-controller for the outer-loop.

2.4.1.1 Inner-loop Controller

Using an H_∞ controller for the inner-loop, both robust stability and proper performance of the system can be achieved, simultaneously. To design a H_∞ controller, using notation analogous with [83], we define the measurement output simply as the state feedback in the form of $y_1 = C_{11}x_1$ with $C_{11} = I$. Also, we define the controlled output h_1 in the form of $h_1 = C_{12}x + D_{12}u$, where

$$C_{12} = \begin{bmatrix} & 0_{2 \times 3} & \\ 3.1623 & 0 & 0 \\ 0 & 3.1623 & 0 \\ 0 & 0 & 1.7321 \end{bmatrix}, D_{12} = \begin{bmatrix} 44.7214 & 0 \\ 0 & 28.2843 \\ 0_{3 \times 2} \end{bmatrix} \quad (2.13)$$

The nonzero entries of C_{12} and D_{12} are used for tuning the controller. Here, they are determined experimentally to achieve the desired performance. Meanwhile, the H_∞ design guarantees internal stability and robustness of the system. Indeed, H_∞ control design reduces the effect of the wind gust disturbance on the control performance, by minimizing the H_∞ norm of the closed-loop transfer matrix from the disturbance w to the controlled output h_1 , denoted by T_1 . The H_∞ norm of the

transfer function T_1 is defined as follows:

$$\|T_1\|_\infty = \sup_{0 \leq \omega < \infty} \sigma_{\max}[T_1(j\omega)] \quad (2.14)$$

where $\sigma_{\max}[*]$ denotes the maximum singular value of the matrix $*$.

Having the matrices C_{12} and D_{12} , one can find γ_∞^* which is the optimal H_∞ performance for the closed-loop system from the disturbance input w to the controlled output h_1 over all the possible controllers that internally stabilize the system. As practically, γ_∞^* is not achievable, we will try to reach γ_∞ which is slightly larger than γ_∞^* .

With this choice of the control parameters, D_{11} and D_{12} are full rank and the quadruples $(A_1, B_1, C_{12}, D_{12})$ and $(A_1, E_1, C_{11}, D_{11})$ are left invertible and are free of invariant zeros. Therefore, we have a so-called regular problem, for which we can use the well-established H_∞ control theory [83]. As it was mentioned, the resulting closed loop system suboptimality minimizes the H_∞ norm of the transfer function from the disturbance w to the controlled output h_1 . To design this controller we consider the control law in the following form:

$$u_1 = F_1 x_1 + G_1 r_1 \quad (2.15)$$

where $r_1 = (V_{z_r}, \omega_{z_r})'$ is the reference signal generated by the outer-loop controller, $G_1 = -(C_1(A_1 + B_1 F_1)^{-1} B_1)^{-1}$ is the feedforward gain, and F_1 is the H_∞ control gain that can be achieved as follows:

$$F_1 = -(D'_{12} D_{12})^{-1} (D'_{12} C_{12} + B'_1 P_1) \quad (2.16)$$

where matrix P_1 is the positive semi-definite solution of the following H_∞ algebraic

Riccati equation:

$$\begin{aligned} & A_1' P_1 + P_1 A_1 + C_{12}' C_{12} + P_1 E_1 E_1' P_1 / \gamma^2 - \\ & (P_1 B_1 + C_{12}' D_{12})(D_{12}' D_{12})^{-1}(D_{12}' C_{12} + B_1' P_1) = 0 \end{aligned} \quad (2.17)$$

For this system and these control parameters values, the value of γ_∞^* is 1.4516.

Choosing $\gamma_\infty = 1.4616$, matrices F_1 and G_1 are obtained as follows:

$$F_1 = \begin{bmatrix} -0.0935 & -0.0005 & 0.0027 \\ 0.0008 & 0.0364 & -0.0481 \end{bmatrix}, \quad G_1 = \begin{bmatrix} 0.1371 & 0.0066 \\ -0.0020 & -0.2748 \end{bmatrix} \quad (2.18)$$

To evaluate the controller performance and its effect on the disturbance attenuation, we simulated the closed loop system with an initial state of $x_1(0) = [1.5 \ 0 \ 0]'$, and also we injected wind gust disturbance for 20 sec (Fig. 2.5). The injected disturbance has a maximum amplitude of 3 m/s along the z axis (the other directions do not affect the dynamics of Subsystem 1). The controlled system reaches the steady hovering state after 3.5 sec, and the disturbance effect is reduced to less than 25%. The control inputs are within the unsaturated region.

2.4.1.2 Outer-loop Controller

In the outer-loop of Subsystem 1, we use a P-controller K_{P_1} (Fig. 2.6). We can redraw this system as shown in Fig. 2.7, in which $G_{in_1} = \frac{1}{s} C_1 (SI - (A_1 + B_1 F_1))^{-1} B_1 G_1$. It can be shown that G_{in_1} is a 2×2 multi-variable system. In general, designing a P-controller for a MIMO system is difficult. However, if we consider K_{P_1} in diagonal form as $K_{P_1} = k_{p_1} I_{2 \times 2}$, we can apply the generalized Nyquist theorem [84] to design

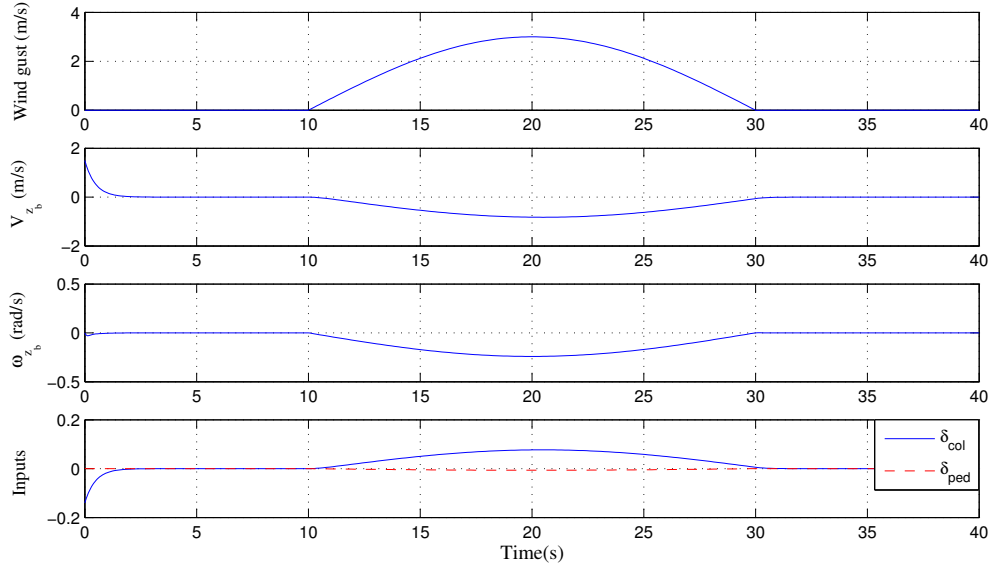


Figure 2.5: Simulation of the inner-loop of Subsystem 1.

k_{p1} to stabilize the system.

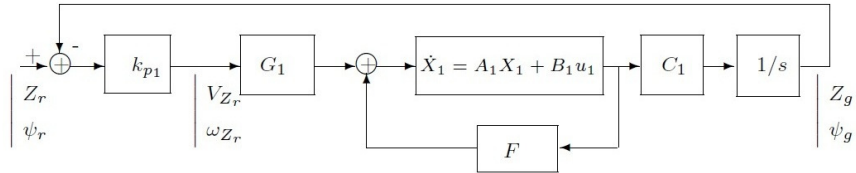


Figure 2.6: Control structure of Subsystem 1.

2.4.1.3 Stability Analysis

The characteristic loci of G_{in_1} are shown in Fig. 2.8, where the dash-dot lines correspond to the infinite values. In Subsystem 1, Fig. 2.7, the inner-loop has already been stabilized, using an H_∞ controller. Therefore, due to the presence of the integral

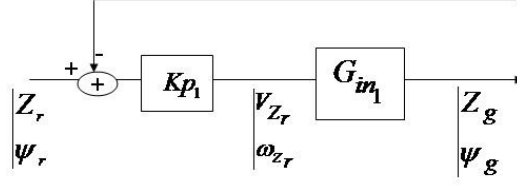


Figure 2.7: Redrawing the control structure of Subsystem 1.

term, G_{in_1} has two poles at the origin and the remaining poles are in the LHP plane. Hence, G_{in_1} has no pole in the Nyquist contour. It follows from the form of the characteristic loci of G_{in_1} in Fig. 2.8, that $k_{p_1} \in (0, \infty)$ will keep the entire system stable. However, in practice, we are subjected to the selection of small values of k_{p_1} to avoid saturation of the actuators. For this experiment we chose $k_{p_1} = 1.5$.

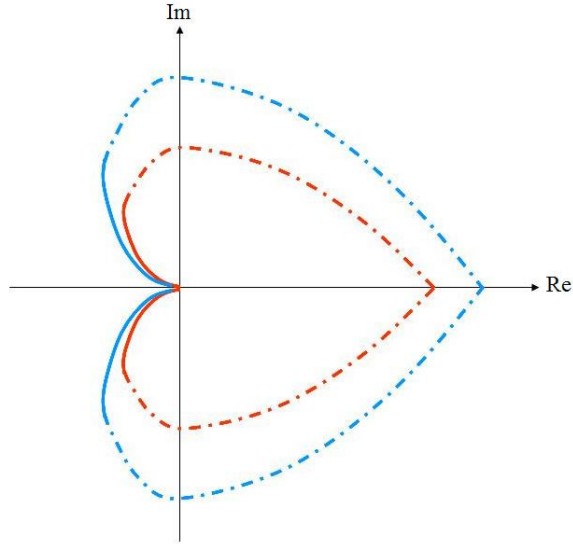


Figure 2.8: Characteristic loci of G_{in_1} .

2.4.1.4 Tuning the Controller

With the above outer-loop controller, the stability of the whole system has been achieved; however, the controller in the form of $K_{p_1} = k_{p_1} \times I_{2 \times 2}$ with only one control parameter is not an appropriate choice. We need to have more degrees of freedom to tune the controller and achieve better performance. By considering the proportional feedback gain in the form of $K_{p_1} = \text{diag}\{K_{p_{11}}, K_{p_{12}}\}$, we have more degrees of freedom and can control each of the output channels in a decentralized manner, while keeping the system decoupled.

Uncertainty analysis usually is used to investigate the effect of the plant uncertainty. Here, we borrow this idea to analyze the effect of deviation of the diagonal entries of the matrix $K_{p_1} = k_{p_1} I_{2 \times 2}$ in the controller part. Alternatively, we can define K_{p_1} as follows:

$$K_{p_1} = \begin{bmatrix} K_{p_{11}} & 0 \\ 0 & K_{p_{12}} \end{bmatrix} = k_{p_1} I_{2 \times 2} + \begin{bmatrix} \Delta_{K_{p_{11}}} & 0 \\ 0 & \Delta_{K_{p_{12}}} \end{bmatrix} \quad (2.19)$$

The objective is to design $\Delta = \text{diag}\{\Delta_{K_{p_{11}}}, \Delta_{K_{p_{12}}}\}$ such that it does not affect the stability of the system. In fact, Δ is the tuning range (Fig. 2.9).

Following from Fig. 2.9, one can extract the internal model of the system as:

$$\begin{cases} y = G_{in_1} k_{p_1} (I + G_{in_1} k_{p_1})^{-1} v + (I + G_{in_1} k_{p_1})^{-1} G_{in_1} z \\ x = (I + G_{in_1} k_{p_1})^{-1} v - (I + G_{in_1} k_{p_1})^{-1} G_{in_1} z \end{cases} \quad (2.20)$$

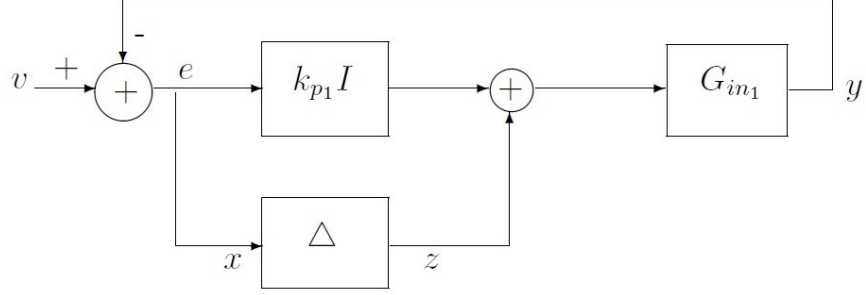


Figure 2.9: Robust system diagram.

To simplify the notation, (2.20) can be rewritten as:

$$\begin{cases} y = G_{11}v + G_{12}z \\ x = G_{21}v + G_{22}z \end{cases} \quad (2.21)$$

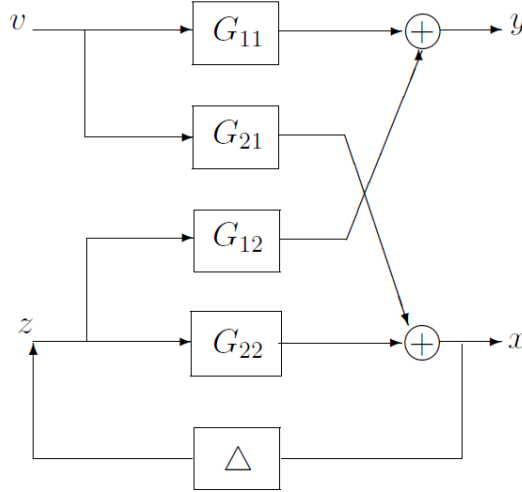


Figure 2.10: Redrawing the Subsystem 1 for robust analysis.

Therefore, Fig. 2.9 can be redrawn as it is shown in Fig. 2.10. In the new diagram, since the nominal system with $\Delta = 0$ is stable, all G_{ij} are stable. G_{11} , G_{12} , and G_{21} are outside the uncertain loop and cannot be affected by block Δ ;

however, the loop includes G_{22} and Δ may affect the internal stability of the system due to perturbations of the elements of Δ . Since G_{22} and Δ are stable, according to the generalized Nyquist theorem, the characteristic loci of the loop transfer function should not encircle the point $(-1+j0)$. Equivalently, we should have $|\lambda_i(-G_{22}\Delta)| < 1$. To satisfy this condition, since $|\lambda_i(G_{22}\Delta)| \leq \bar{\sigma}(G_{22}\Delta) \leq \sup_{\omega}(\bar{\sigma}(G_{22}\Delta)) = \|G_{22}\Delta\|_{\infty}$, it is sufficient that $\|G_{22}\Delta\|_{\infty} < 1$. Using norm properties, we have:

$$\|G_{22}\Delta\|_{\infty} \leq \|G_{22}\|_{\infty} \|\Delta\|_{\infty} \quad (2.22)$$

Therefore, the sufficient condition for the stability of the system is:

$$\|G_{22}\|_{\infty} \|\Delta\|_{\infty} < 1 \quad (2.23)$$

For these values of the controller and plant parameters and for a frequency range of $(0, 10000)$, we obtained $\|G_{22}\|_{\infty} = 0.6986$. Therefore, the perturbation of K_{p1} should be such that $\|\Delta\|_{\infty} \leq 1.4315$. Recall that Δ has a diagonal structure, and hence, all diagonal entries of K_{p1} should have less than a 1.4315-unit deviation from their nominal value. In fact, using this approach, we first obtained a nominal controller that provides the stability of the system, and then, we attempted to tune the controller to improve the performance, while keeping the system stable. After tuning the controller, the value of $K_{p1} = \text{diag}\{0.5, 0.7\}$ was selected as an appropriate value that satisfies the above mentioned condition and gives a satisfactory performance. The method is conservative as Δ is structured and real, but applying to the UAV plant it has provided sufficient degree of freedom for tuning the controller and improving the performance.

To simulate the resulting system, let the outer-loop reference be $(Z_r, \psi_r) = (-2, 0.5)$ and the current position and heading angle be $(Z_g, \psi_g) = (0, 0)$. The system will reach its target after approximately 8 sec as shown in Fig. 2.11.

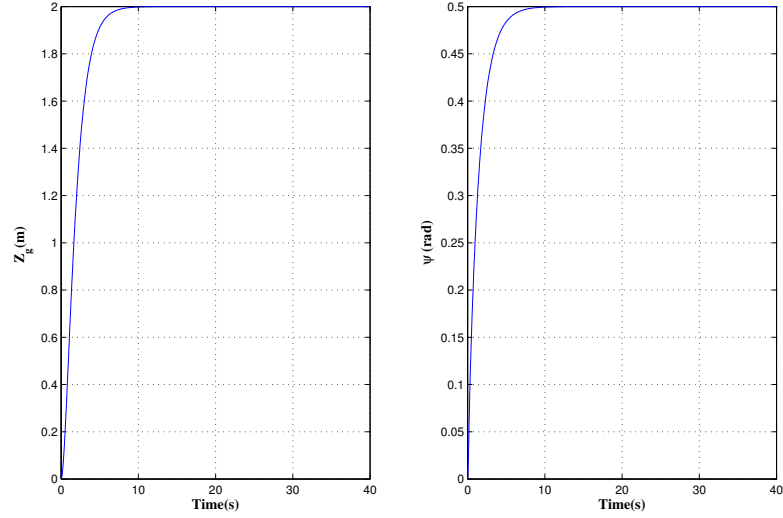


Figure 2.11: Simulation of the outer-loop of Subsystem 1.

2.4.2 Designing the Controller for Subsystem 2

2.4.2.1 Inner-loop Controller

For Subsystem 2, described by (2.11), we use an H_∞ controller for the inner-loop controller design. Analogous with Subsystem 1, we define h_2 as $h_2 = C_{22}x_2 + D_{22}u_2$, where

$$C_{22} = \begin{bmatrix} & & & 0_{2 \times 8} & & & & \\ & 0.3162 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0.3162 & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0.3162 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0.3162 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$D_{22} = \begin{bmatrix} 5.4772 & 0 \\ 0 & 5.4772 \\ 0_{8 \times 2} \end{bmatrix}.$$

With these parameters, we will obtain $\gamma_\infty^* = 0.0731$. Choosing $\gamma_\infty = 0.0831$, we will have:

$$F_2 = \begin{bmatrix} 0.0017 & -0.1683 & -0.0486 & 0.0081 & -1.9336 & -0.1974 & -0.3227 & -2.1444 \\ 0.0815 & -0.0461 & -0.0087 & -0.0535 & -0.3908 & -1.0690 & -1.1712 & -0.4659 \end{bmatrix}.$$

Moreover, $G_2 = -(C_2(A_2 + B_2 F_2)^{-1} B_2)^{-1}$, is the feedforward gain for Subsystem 2 and can be calculated as $G_2 = \begin{bmatrix} -0.0029 & 0.2335 \\ -0.0978 & 0.0632 \end{bmatrix}$.

The simulation of the system is shown in Fig. 2.12. In this figure, the initial state of the system is $x_2(0) = [1.5 \ 0 \ 0 \ 0 \ 0.17 \ 0 \ 0 \ 0]'$. The injected disturbance has a

maximum amplitude of 10 m/s along the x and y axes (the z direction does not affect the dynamics of Subsystem 2). The controlled system reaches the steady hovering state after 3.5 sec, and the disturbance effect is reduced to less than 25%. In addition, the control inputs are within the unsaturated region.

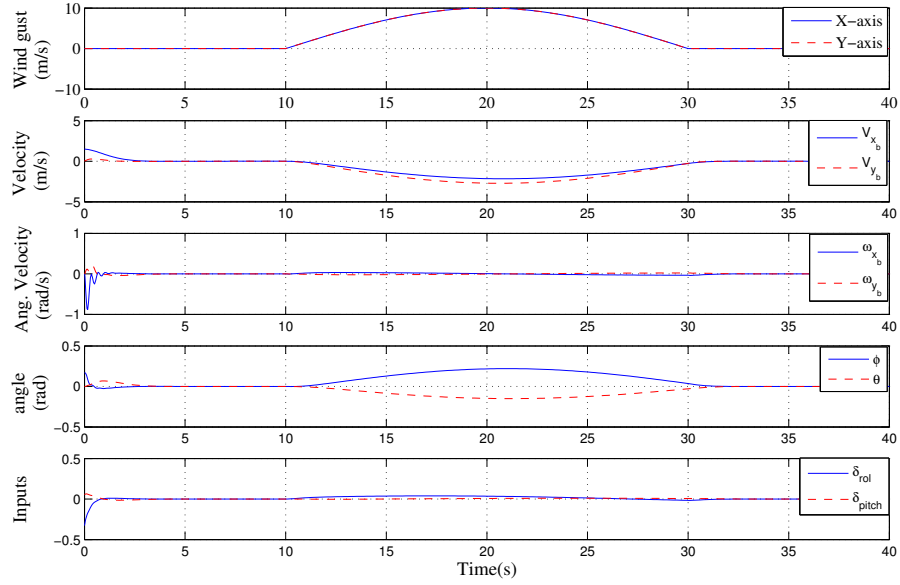


Figure 2.12: Simulation of the inner-loop of Subsystem 2.

2.4.2.2 Outer-loop Controller

Although the outer-loop of Subsystem 2 is similar to the outer-loop of Subsystem 1, the main difference lies in the presence of the nonlinear term, R^{-1} , in the outer-loop of Subsystem 2, as shown in Fig. 2.4. In this structure, the error signal is the difference between the actual position and the reference position, which both are in the ground frame. Therefore, the resulting control signal, which is going to be

passed to the inner-loop as the reference, is obtained in the ground frame; however, the inner-loop dynamics is derived in the body frame. Hence, it is reasonable that we transform the control signal to the body frame before delivering it to the inner-loop. To implement this idea, we can use the transformation term, R , to obtain a control signal in the body frame. The new structure is shown in Fig. 2.13, in which $G_{in_2} = C_2(SI - (A_2 + B_2F_2))^{-1}B_2G_2$ is a 2×2 multi-variable system. In Fig. 2.14, it is shown that the inner-loop block G_{in_2} is very close to a decoupled system with equal diagonal elements. Indeed, Subsystem 2 corresponds to the dynamics of the helicopter for the $x - y$ plane movement and, we expect the dynamics of the UAV in the x and y directions to be similar and decoupled. Using this concept, we can take the block G_{in_2} out so that the two rotation matrices R and R^{-1} will cancel each other.

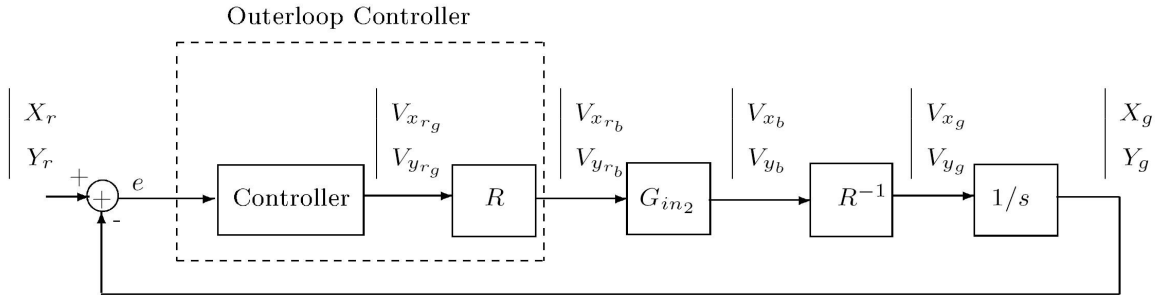


Figure 2.13: Control diagram of Subsystem 2.

The remaining job is simple, and we can repeat the procedure of designing the outer-loop controller for Subsystem 1 and design a P-controller in the form of $K_{p_2} = \text{diag}\{K_{p_{21}}, K_{p_{22}}\}$ that stabilizes Subsystem 2 (Fig. 2.15). As an appropriate choice of control parameters, we can select $K_{p_2} = \text{diag}\{0.3, 0.3\}$. Rationally, $K_{p_{21}}$ and $K_{p_{22}}$ should be the same, since we expect a similar behavior of the UAV system in the x

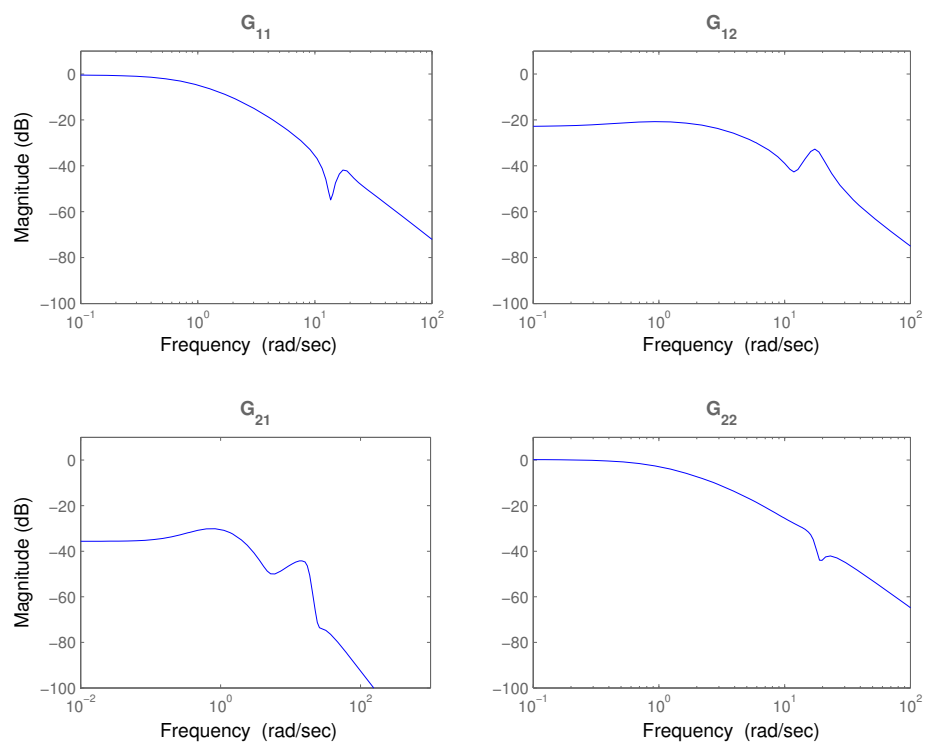


Figure 2.14: Bode plot of entries of G_{in_2}

and y directions.

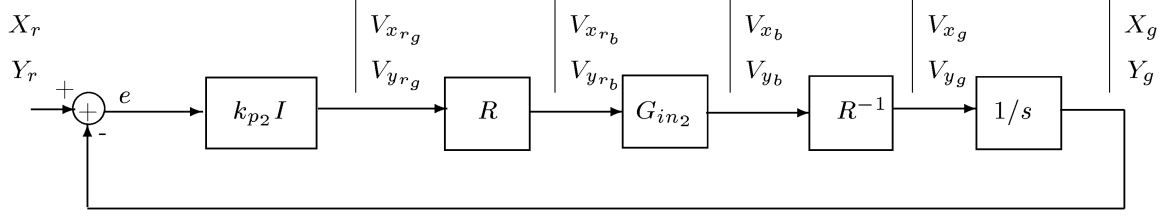


Figure 2.15: Redrawing the control diagram of Subsystem 2.

For an outer-loop reference at $(x_r, y_r) = (2, 2)$ and the current UAV position at $(x_g, y_g) = (0, 0)$, the simulation results are shown in Fig. 2.16, in which the UAV reaches to the desired position after approximately 10 sec, smoothly and without overshooting.

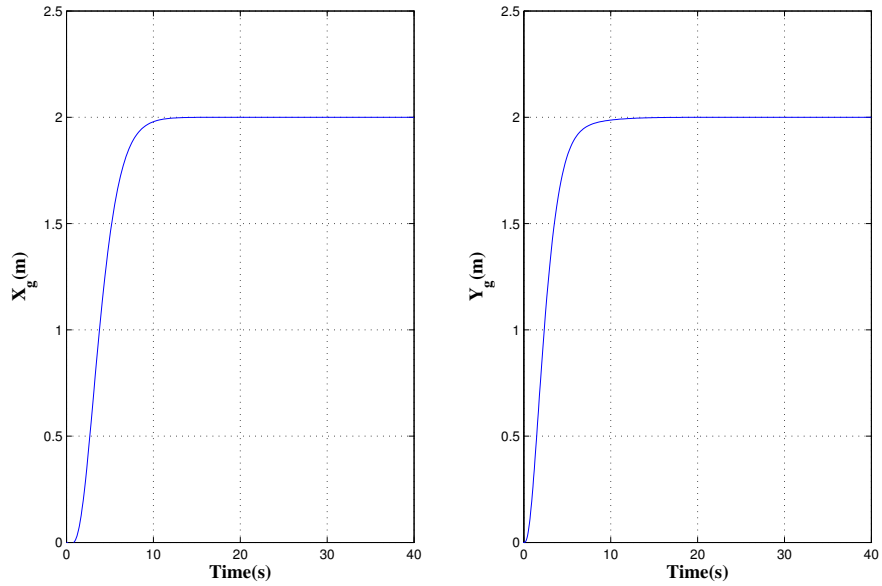


Figure 2.16: Simulation of the outer-loop of Subsystem 2.

2.5 Experimental Results

Before using the designed controller in an actual flight test, we first evaluate it through a hardware-in-the-loop simulation platform [85]. In this platform, the nonlinear dynamics of the UAV has been replaced with its nonlinear model, and all software and hardware components that are involved in a real flight test remain active during the simulation. Using the hardware-in-the-loop simulation environment, the behavior of the system is very close to the real experiments.

Then, we conducted actual flight tests to observe the in-flight behavior of the helicopter. First, we used the UAV in the hovering state for 80 sec. Figure 2.17 shows the state variables in the hovering experiment at $(x, y, z, \psi) = (-16, -34, 10, -1.5)$. To evaluate the hovering control performance, the position of the UAV is depicted in a 2D $x - y$ plane in Fig. 2.18. As it can be seen, the position of the UAV has at most a 1-meter deviation from the desired hovering position, which is quite satisfactory. The control inputs are also shown in Fig. 2.19. All of the control inputs are within the unsaturated region.

Next, we used the UAV to follow a circle with a diameter of 20 meters as a given trajectory. This trajectory determines the reference $(x(t), y(t), z(t), \psi(t))$ for the system. With this trajectory, the UAV should complete the circle within 70 sec. In Fig. 2.20, it is shown that the UAV is able to follow this trajectory successfully. The UAV path tracking in the $x - y$ plane is shown in Fig. 2.21. Moreover, to have a better insight of the system behavior, the states variables of the UAV and the control inputs are represented in Fig. 2.22 and Fig. 2.23, respectively. These results show

that the UAV is able to track the desired trajectory in situations close to the hovering state. The small deviations in the hovering mode or path tracking mode could be due to environmental effects such as wind disturbances and the GPS signal inaccuracy. Here, as measured by a handheld anemometer the wind disturbances is 2 to 3 m/s in the horizontal plane, the steady-state accuracy of the heading angle is 2.5° , and the position accuracy of the GPS signal is $3m(1\sigma)$. Videos of the hovering experiment and circle path tracking are available at <http://uav.ece.nus.edu.sg/video/hover.mpg> and <http://uav.ece.nus.edu.sg/video/circle.mpg> , respectively.

2.6 Conclusion

In this chapter, we presented a systematic approach for the flight control design of a small-scale UAV helicopter. In the proposed two-layer control structure, the lower level aims at the stabilization of the system, and the upper level focuses on the reference tracking. For the disturbance attenuation and stabilization of the UAV, we used an H_∞ controller in the inner-loop of the system. Due to the presence of some nonlinear terms in the outer-loop of the system, we first compensated for the nonlinearity by an inverse rotation; then, we used a decentralized P-controller to enable the UAV to follow a desired trajectory. We also proposed a new method of designing a P-controller for MIMO systems that was successfully applied to the UAV system. The simulations and actual flight tests show the efficacy of the control structure.

In the next chapter, we will use this structure to accomplish more complex mis-

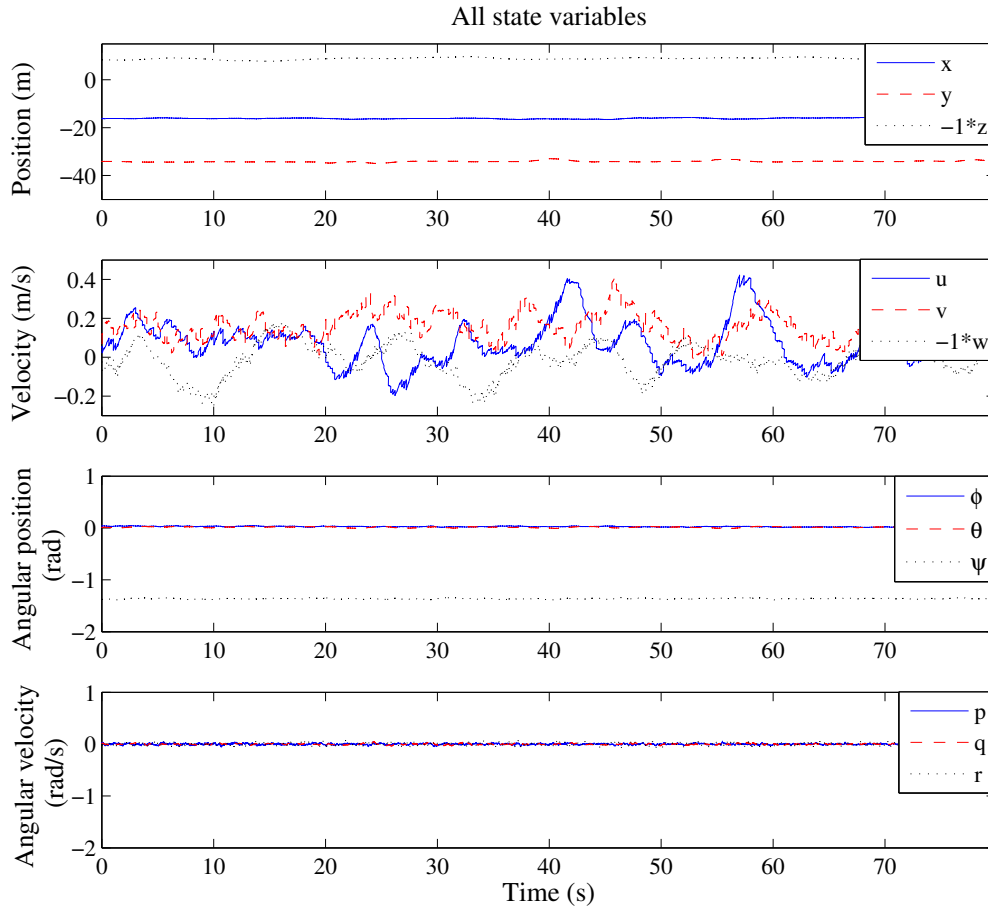


Figure 2.17: State variables of the UAV for the hovering.

sions which consist of several tasks. Such missions will require an embedded decision-making unit to support the tasks and to switch between the controllers.

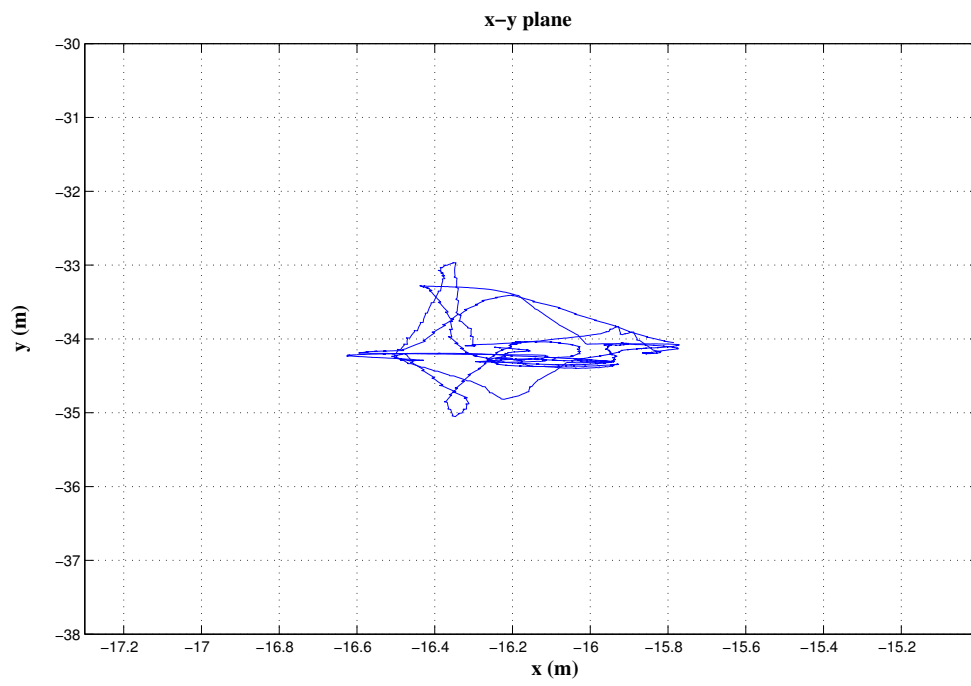


Figure 2.18: UAV position in $x - y$ plane at hovering

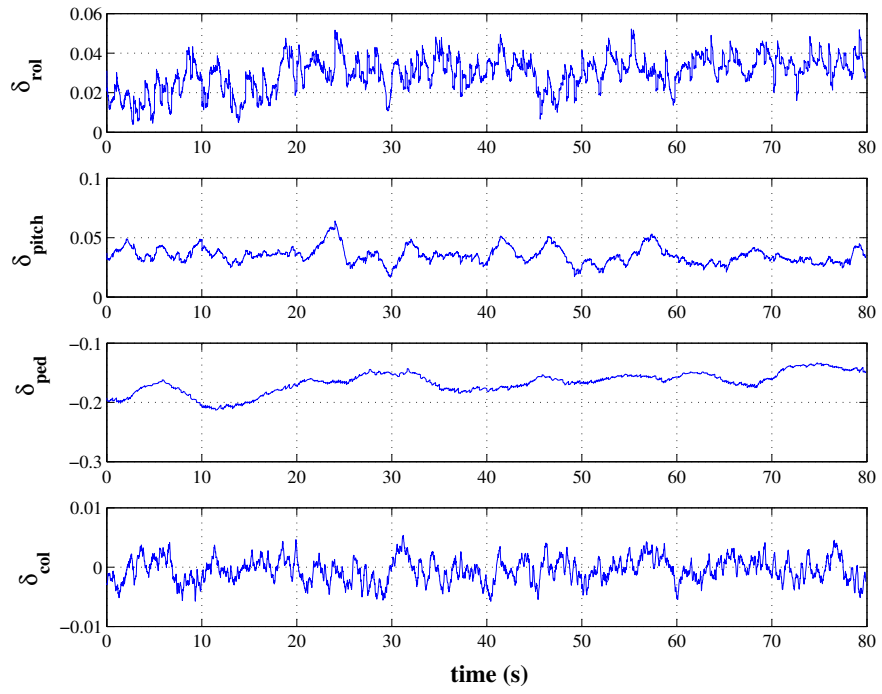


Figure 2.19: Control signals at hovering

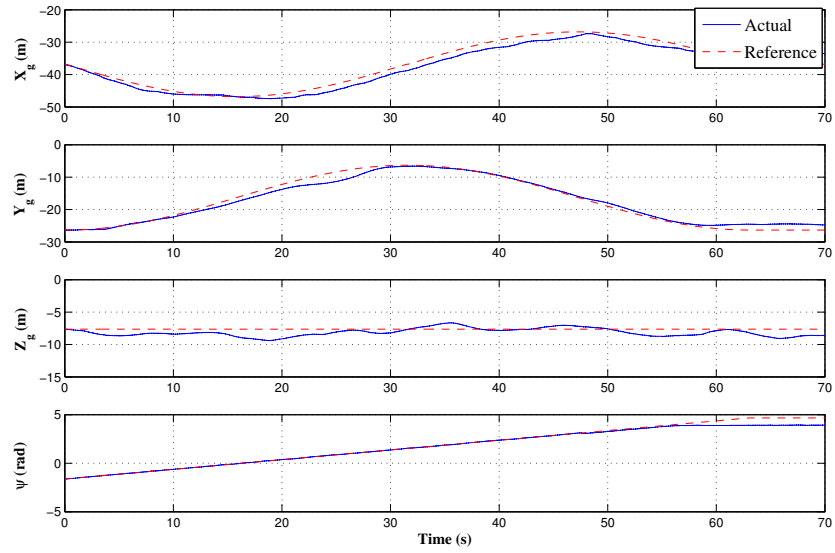


Figure 2.20: Tracking a desired path.

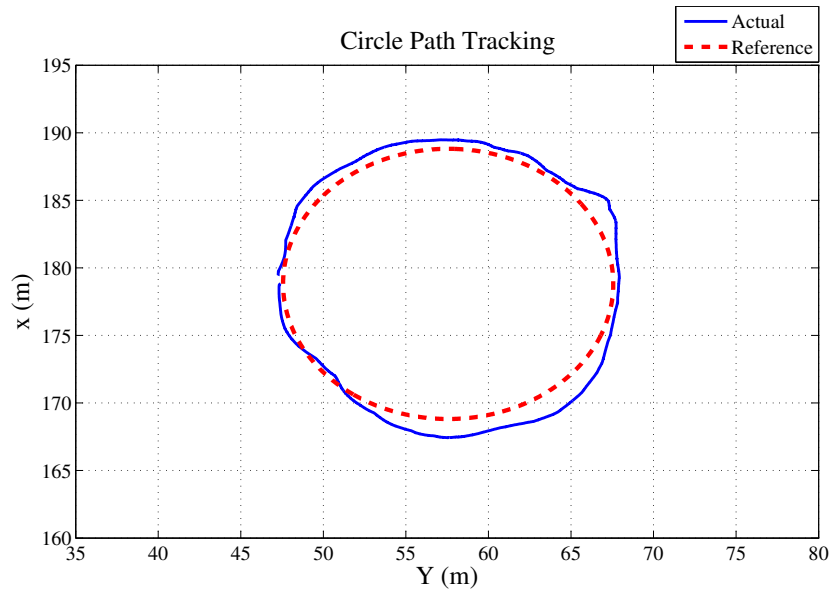


Figure 2.21: Circle path tracking in $x - y$ plane.

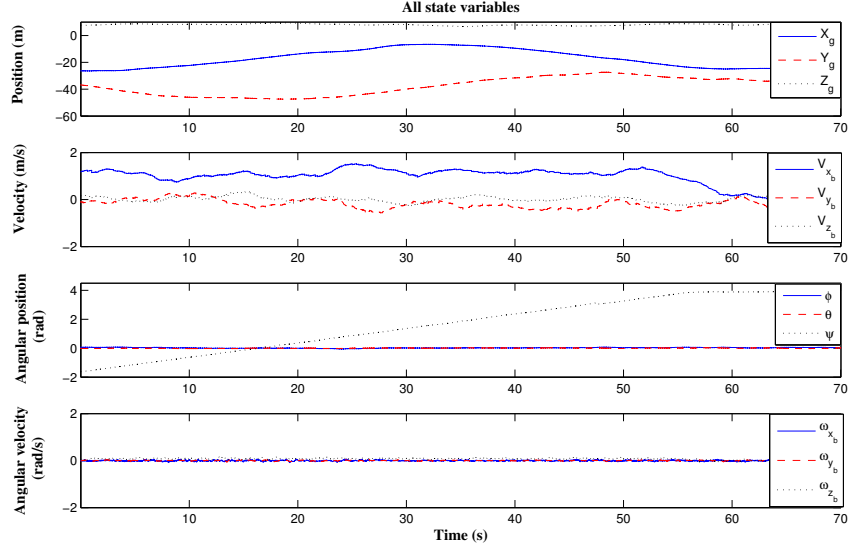


Figure 2.22: States of the UAV in the circle path tracking behavior.

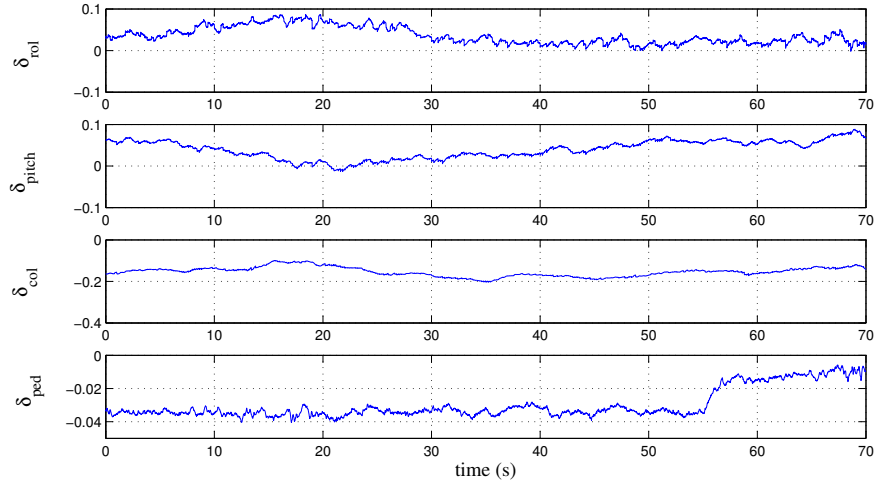


Figure 2.23: Control inputs in the circle path tracking behavior.

Chapter 3

Hybrid Modeling and Control of an Unmanned Helicopter

3.1 Introduction

In this chapter, our aim is to propose a framework for the hybrid control of a UAV helicopter so that it can autonomously accomplish the assigned mission including several tasks. To reduce the complexity of the system, we have developed a hierarchical control structure to distribute the control tasks among the layers. The proposed control system has three layers: the regulation layer, the coordination layer, and the supervision layer. Each layer has a hybrid structure and is responsible to do a specific task. The layers have been modelled with a hybrid Input/Output automaton [50]. Then, a composition operator is introduced to synchronize the layers and capture the interactions between them. Figure 3.1 shows the overall picture of this hierarchical system and describes the nature and objectives of each layer. The philosophy behind this hierarchy is that the lower levels are involved in more details such as reference tracking and stability analysis, while the higher levels mostly manage and coordinate the control scenarios. The advantage of this structure is that it simplifies the design procedure so that each layer can be developed to accomplish a particular part of the

control task.

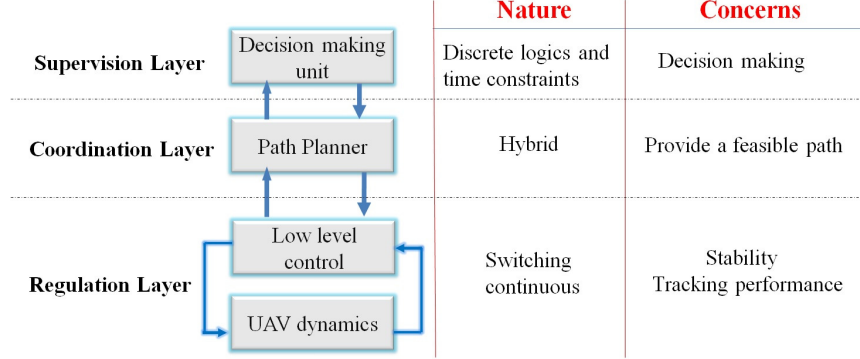


Figure 3.1: Hierarchical hybrid control structure of an autonomous UAV Helicopter.

Furthermore, the designed controller is implemented on the NUS UAV helicopter, and real flight tests are conducted to evaluate the proposed hybrid control structure. The flight test results show that the designed control system can be effectively involved in a complex mission composed of several tasks.

The remaining parts of this chapter are organized as follows. Section 3.2 describes the regulation layer which is responsible for the low level control of the system, Section 3.3 explains the coordination layer which is responsible for generating a path to be followed by the regulation layer, and Section 3.4 describes the supervision layer which is responsible for managing the switching scenario to perform a mission, autonomously. In Section 3.5, the composition operator is discussed. The experimental results are presented in Section 3.6, and finally, the paper is concluded in Section 3.7.

3.2 The Regulation Layer

The regulation layer is the heart of the control structure of the UAV, which is directly connected to the UAV avionic system and can manipulate the actuators and gather the sensors reading for the control process. It also receives task scheduling commands from the coordination layer to activate proper control modes. The model of the NUS UAV helicopter was given in Part 2.7. For different velocities and situations, different controllers can be designed. For example, in [86], several controllers have been designed for different modes of operation of the NUS UAV helicopter. Then, the higher layers are responsible to manage the switching between these control modes. To elaborate the idea of hierarchical control, without loss of generality, here we consider two control modes for the regulation layer of the UAV: the velocity control mode and the position control mode, and then will give hybrid model for this layer.

3.2.1 Velocity Control Mode

In the velocity control mode, vc , one can stabilize the attitude of the helicopter and control the UAV to move with the desired velocity vector (v_{xr}, v_{yr}, v_{zr}) and the desired yaw rate, w_{zr} . Considering the model of the UAV, described by (2.6), the control law can be chosen in the form of $u = Fx_{in} + Gr$. The control diagram is shown in Fig. 3.2. Here, matrix $F = \begin{bmatrix} F_2 & 0 \\ 0 & F_1 \end{bmatrix}$ is obtained through the robust H_∞ design technique [83], [87], and $G = \begin{bmatrix} G_2 & 0 \\ 0 & G_1 \end{bmatrix}$ is the feedforward gain. The values of

matrices F_1 , F_2 , G_1 , and G_2 were calculated in Part 2.4. It was also shown that with these control parameters, both stability and satisfactory robust performance of the system are achieved.

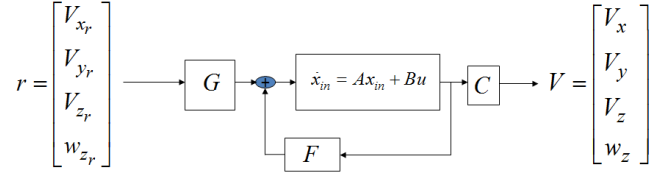


Figure 3.2: The controller for the velocity-control of the UAV.

3.2.2 Position Control Mode

The control objective in the position control mode, pc , is to drive the UAV to follow the generated path. In other words, the state variable x_{out} should track the given reference r . The control law for this operation mode is $u = Fx_{in} + G\Omega(\Theta)K_p(r - x_{out})$,

where $K_p = \begin{bmatrix} K_{p2} & 0 \\ 0 & K_{p1} \end{bmatrix}$. The values of K_{p1} and K_{p2} were calculated in Part 2.4.

It was also discussed that this controller consists of two layers: the inner-loop and the outer-loop. The inner-loop controller stabilizes the attitude of the UAV, and its parameters, F and G , are selected as the same as the velocity control mode. The outer-loop controller, however, smoothly drives the UAV towards the desired position.

In the outer-loop, the block Ω is used to compensate for the effect of the transformation matrix Ω' , and K_p is a P-controller.

3.2.3 Hybrid Model of the Regulation Layer

Now, we can present the hybrid model of the regulation layer based on what was explained for each control mode. Both control modes have the same plant dynamics $\dot{x}_{in} = Ax_{in} + Bu$; however, the control law in the velocity control mode is $u = Fx_{in} + Gr$, and in the position control mode is $u = Fx_{in} + G\Omega(\Theta)K_p(r - x_{out})$.

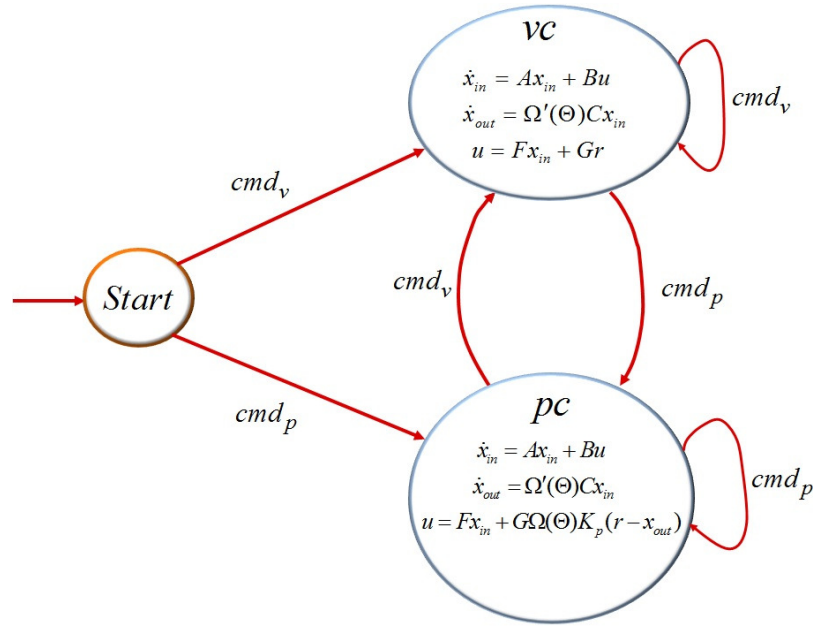


Figure 3.3: The graph representation of the hybrid automaton H_R for the regulation layer.

The graph representation of the hybrid model of the regulation layer is shown in Fig. 3.3. Formally, this hybrid model of the regulation layer can be described by a hybrid automaton ([50], [88]) $H_R = (V_R, X_R, U_R, Y_R, f_R, Init_R, Inv_R, E_R, Guard_R, Reset_R, h_R)$, where

- $V_R = \{start, vc, pc\}$ is the set of discrete states. Here, vc and pc stand for the

velocity control mode and the position control mode, respectively. The *start* mode is used for the initialization of the system to choose either of the modes. These discrete states are considered as the vertices in the graph representation of the regulation layer (Fig. 3.3).

- $X_R = [x_{in}, x_{out}]'$ is the continuous state of the system.
- $U_R = U_{D_R} \times U_{C_R}$ is the input space, where $U_{C_R} = r \subseteq \mathbb{R}^4$ is the continuous control input, and $U_{D_R} = \{cmd_V, cmd_P\}$ is the set of discrete inputs. The subscripts denote the corresponding ending discrete states in Fig. 3.3. For instance, cmd_P is the command that fires a transition to the position control mode.
- $Y_R = Y_{D_R} \times Y_{C_R}$ is the system output, where here, $Y_{C_R} = x_{out}$ and $Y_{D_R} = V_R$ feedback the current continuous output of the system to the coordination layer to be able to generate appropriate reference signals.
- $f_R : V_R \times X_R \times U_R \rightarrow X_R$ is the vector field description of the system which is defined as follows:

$$\dot{x} = f_R(v, x, u) = f_R(v, x, u_d, r) = \begin{cases} 0 & \text{if } u_d = start \\ \begin{bmatrix} (A + BF)x_{in} + BGr \\ \Omega'(\Theta)Cx_{in} \end{bmatrix} & \text{if } u_d = vc \\ \begin{bmatrix} (A + BF)x_{in} - BG\Omega(\Theta)K_px_{out} + BG\Omega(\Theta)K_pr \\ \Omega'(\Theta)Cx_{in} \end{bmatrix} & \text{if } u_d = pc \end{cases}$$

- $Init_R = \{(start, 0)\} \subseteq V_R \times X_R$ is the set of initial states of the UAV.

- $Inv_R \subseteq V_R \times X_R \times U_R$ is the invariant condition. Here, it is required that for both discrete modes, $z > 0$, $V_{xb}, V_{yb}, V_{xb} < 3.5m/s$, $\omega_z < 15deg/s$ and $a, b, \theta, \phi < \frac{\pi}{6}$.
- $E_R \subseteq V_R \times V_R$ is the set of discrete transitions. Here, $E = \{(Start, vc), (start, pc), (pc, vc), (vc, pc), (pc, pc), (vc, vc)\}$ is the set of discrete transitions which are shown by the edges in the graph representation of the regulation layer in Fig. 3.3.
- $Guard_R : E_R \rightarrow 2^{X_R \times U_R}$ describes the guard conditions for the discrete transitions. For each discrete transition from the vertex v to v' , the continuous state of the system and the control input should belong to $Guard(v, v')$. In the graph representation, the continuous part of the guard condition is shown near the beginning of the corresponding edge while its discrete part is depicted at the middle of the edge. For instance, in Fig. 3.3, when the system is in mode vc , the control input cmd_P can cause a transition to the mode pc . For the guard map of this transition, no condition has been considered on the continuous state of the system, and only the discrete control input is used for the guard condition.
- $Reset_R : E_R \times X_R \times U_R \rightarrow 2^{X_R}$ describes the reset map. For instance, $z' \in Reset(v, v', z, w)$ shows that for $(v, v') \in E$, $z \in X$, and $w \in U$, there is a transition for which the continuous state of the system will be reset to z' . In the graph representation, the reset map is usually shown near the end of the corresponding edge. In the hybrid model of the regulation layer, the reset map

is an identity map as there is no jump on the continuous state of the system.

When the reset map is an identity map, it is not shown in the graph.

- $h_R : V_R \times X_R \rightarrow Y_R$ is the output map. Here we have $h(v, x) = x_{out}$.

3.3 Coordination Layer

Based on the feedbacked information received from the regulation layer, the coordination layer can activate the corresponding control mode in the regulation layer and can generate proper control references in the form of a feasible path for the regulation layer. The path generation mechanism could be done in an off-line manner or through a dynamic path planning mechanism. In Chapter 4, after proposing the formation algorithm, an online path planning mechanism will be introduced. But, in this chapter we will talk about the off-line path planning mechanism in which based on the problem requirements and constraints, a path can be generated and stored in the library of the system. As an example, we explain a coordination layer that has been used in our flight tests using off-line path generation mechanism. The hybrid automaton for this model of the coordination layer is $H_{P_1} = (V_{P_1}, X_{P_1}, U_{P_1}, Y_{P_1}, f_{P_1}, Init_{P_1}, Inv_{P_1}, E_{P_1}, Guard_{P_1}, Reset_{P_1}, h_{P_1})$, where $X_{P_1} = (r_x, r_y, r_z, r_\psi)$ is the continuous state of the coordination layer and indeed, it is the generated reference that is going to be given to the regulation layer. The discrete state is $V_{P_1} = \{Start_p, PathZ_p, PathC_p, Ascend_p, Hover_p, Vel_p, Descend_p, Emergency_p\}$, where $Start_p$, $PathZ_p$, $PathC_p$, $Ascend_p$, $Hover_p$, Vel_p , $Descend_p$, and $Emergency_p$ stand for starting the task, zigzag path tracking, circle path track-

ing, ascending, hovering, generating velocity references, descending, and emergency mode, respectively. Here, the control signal is $U_{P_1} = U_{C_{P_1}} \times U_{D_{P_1}}$, where $U_{C_{P_1}} = Y_{C_R}$ is the current continuous output that is feedbacked from the regulation layer, and $U_{D_{P_1}} = \{cmd_{PathZ}, cmd_{PathC}, cmd_{Ascend}, cmd_{Hover}, cmd_{Vel}, cmd_{Descend}, cmd_{Emergency}\}$ is the set of commands that can be received from the supervision layer. When the coordination layer receives one of these commands, it switches to the corresponding discrete mode. The layer output is $Y_{P_1} = Y_{D_{P_1}} \times Y_{C_{P_1}}$, where $Y_{C_{P_1}} = X_{P_1}$ is the continuous part which informs the supervision layer about the current state of the coordination layer and also, it will be given to the regulation layer as the generated reference to be tracked. $Y_{D_{P_1}} = Y_{D_{pr_1}} \times Y_{D_{ps_1}}$ is the discrete output signal, where $Y_{D_{ps_1}} = V_{P_1}$ is given to the supervisor to inform about the current discrete mode of the coordination layer, and $Y_{D_{pr_1}} = \{cmd_p, cmd_v\}$ is the command that activates the proper control mode in the regulation layer:

$$Y_{D_{pr_1}} = \begin{cases} cmd_p & \text{for } V_{P_1} = PathC_p, PathZ_p, Ascend_p, Descend_p, Hover_p \\ cmd_v & \text{for } V_{P_1} = Vel_p, Emergency_p \end{cases}$$

The dynamics of the coordination layer is

$$\dot{X}_{P_1}(v) = [\dot{x}_r \quad \dot{y}_r \quad \dot{z}_r \quad \dot{\psi}_r]^T = \begin{cases} (0, 0, f_{z_a}(t), 0) & v = Ascend_p \quad f_z(t) > 0 \\ (0, 0, f_{z_d}(t), 0) & v = Descend_p \quad f_z(t) < 0 \\ (0, 0, 0, 0) & v = Hover_p, Emergency_p \\ (f_{x_{pc}}(t), f_{y_{pc}}(t), f_{z_{pc}}(t), f_{\psi_{pc}}(t)) & v = PathC_p \\ (f_{x_{pz}}(t), f_{y_{pz}}(t), f_{z_{pz}}(t), f_{\psi_{pz}}(t)) & v = PathZ_p \\ (f_{x_v}(t), f_{y_v}(t), f_{z_v}(t), f_{\psi_v}(t)) & v = Vel_p \end{cases}$$

Here, f_* is the path generation mechanism. In the graph representation for the hybrid model of the coordination layer, the system starts from the mode $Start_p$, all discrete states are connected, and the command cmd_* can fire a transition to the state $*$. There is no guard condition and jump for the discrete transitions. As this graph is tedious, we have not shown it here.

3.4 Supervision Layer

This layer is responsible for the decision making and task scheduling for the mission that should be performed by the UAV. The supervision layer can be presented by a purely discrete automaton [67] or a timed automaton [89] which are subclasses of hybrid systems.

As an example, a supervision layer has been designed for a typical mission shown in Fig. 3.4. This mission starts with 8 meters ascending, followed by 15 sec hovering, 60 sec zigzag path tracking, 35 sec velocity control, 42 sec circle path tracking, 20 sec hovering, and 8 meters descending. The mission ends with hovering. For safety issues, when the measured signals are out of range, the fuel level sensor alarms, or other possible problems occur, a fault signal is generated which leads the system to the emergency mode. The hybrid model for this supervisor can be described by a hybrid automaton $H_{S_1} = (V_{S_1}, X_{S_1}, U_{S_1}, Y_{S_1}, f_{S_1}, Init_{S_1}, Inv_{S_1}, E_{S_1}, Guard_{S_1}, Reset_{S_1}, h_{S_1})$, where $X_{S_1} = t$ to capture the time. The discrete states and corresponding discrete outputs are shown in Fig. 3.4. These discrete outputs are commands that activate a control mode in the coordination layer. The input space of this layer is in the form of

$U_{S_1} = U_{C_{S_1}} \times U_{D_{S_1}}$, where $U_{C_{S_1}} = Y_{C_{ps_1}} = X_{P_1}$ is the current state of the path planner, and $U_{D_{S_1}} = U_{D_{se_1}} \times U_{D_{sp_1}}$. Here, $U_{D_{sp_1}} = Y_{D_{ps_1}} = V_{p_1}$ is the information about the current discrete mode of the coordination layer, and $U_{D_{se_1}} = \{Cmd_{StartMission}, Fault\}$ is the set of external events generated by the other sources such as the ground station.

The graph representation for this supervisor is shown in Fig. 3.4.

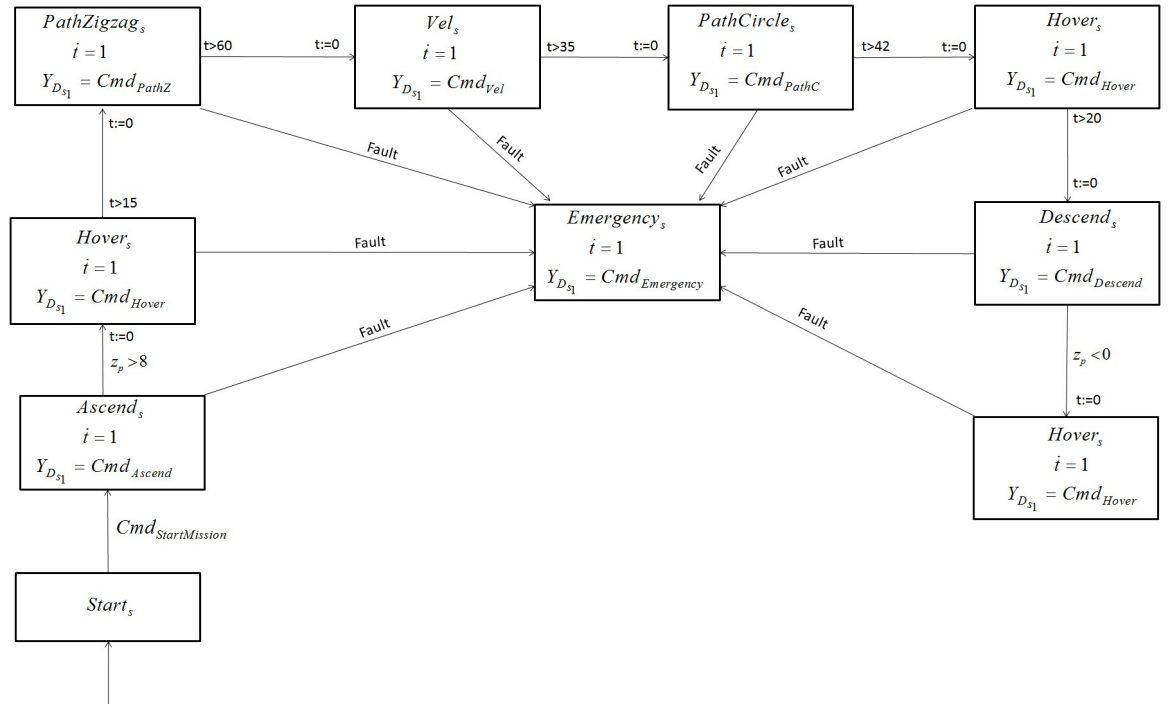


Figure 3.4: The graph representation of the automaton H_{S_1} as the hybrid model for supervision layer for a mission with successive tasks.

3.5 The Composed Hybrid System

So far the layers of the control structure are described. However, to establish such a hierarchy, it is required to introduce a composition operator to synchronize the layers

of the control hierarchy and to capture their relation. In [90], a definition of parallel composition for fully connected hybrid systems is introduced. The resulting closed loop system for such a system is an autonomous unit which cannot be extended to a multi-agent scenario or a multi-layer structure. In [91] and [92], a more general definition of composition of hybrid systems has been given in which the components need not to be fully connected. However in this method, the elements are only coexist in the combined system and there is no refinement on the transitions and states of the closed loop system. In contrast, here, we give a new definition of the composition operator for hybrid systems that can be used for hybrid-multi-agent systems or a multi-layer hybrid system. Furthermore, it considers a treatment on the discrete transitions and states of the composed system which leads to a more simplified system. First, we need to define the composability condition:

Definition 1 *Composability of hybrid automata*

Hybrid automata H_1, H_2, \dots, H_n are composable if:

1. $Y_i \cap Y_j = \emptyset, V_i \cap V_j = \emptyset, X_i \cap X_j = \emptyset$ for all $i \neq j$ and $i, j = 1, \dots, n$.
2. $U_i \setminus Y_i = \emptyset$ for all $i = 1, \dots, n$.

The first condition avoids the conflict between the components and the second condition guarantees the causality condition.

Definition 2 *Composition of hybrid automata*

Consider two composable hybrid automata $H_1 = (V_1, X_1, U_1, Y_1, f_1, Init_1, Inv_1, E_1, Guard_1, Reset_1, h_1)$ and $H_2 = (V_2, X_2, U_2, Y_2, f_2, Init_2, Inv_2, E_2, Guard_2, Reset_2, h_2)$.

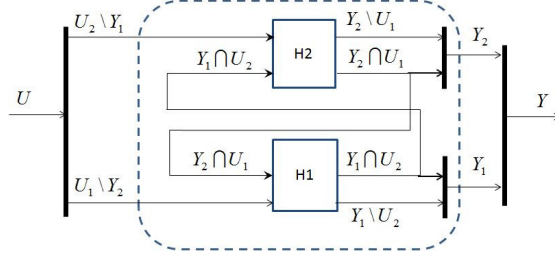


Figure 3.5: Input and output channels for two composed systems.

The composition of H_1 and H_2 , denoted by $H_1 \parallel H_2$, is the automaton $H = (V, X, U, Y, f, \text{Init}, \text{Inv}, E, \text{Guard}, \text{Reset}, h)$, where:

- $V = V_1 \times V_2$ and $X = X_1 \times X_2$.
- $U = (U_1 \setminus Y_2) \times (U_2 \setminus Y_1)$ and $Y = Y_1 \times Y_2$ (See Fig. 3.5).
- $h : V \times X \rightarrow Y$, where $h = \begin{bmatrix} h_1 : V_1 \times X_1 \rightarrow Y_1 \\ h_2 : V_2 \times X_2 \rightarrow Y_2 \end{bmatrix}$.
- $f : V \times X \times U \rightarrow X$, and $f(v, x, u) = \begin{bmatrix} f_1(v_1, x_1, u_1) \\ f_2(v_2, x_2, u_2) \end{bmatrix} = \begin{bmatrix} f_1(v_1, x_1, (u_1 \setminus y_2, u_1 \cap y_2)) \\ f_2(v_2, x_2, (u_2 \setminus y_1, u_2 \cap y_1)) \end{bmatrix} = \begin{bmatrix} f_1(v_1, x_1, (u_{11}, h_{21}(v_2, x_2))) \\ f_2(v_2, x_2, (u_{22}, h_{12}(v_1, x_1))) \end{bmatrix}$, where $v = (v_1, v_2)$, $x = (x_1, x_2)$, $u = (u_1 \setminus y_2, u_2 \setminus y_1) = (u_{11}, u_{22})$, $h_{12} : V_1 \times X_1 \rightarrow Y_1 \cap U_2$, and $h_{21} : V_2 \times X_2 \rightarrow Y_2 \cap U_1$.
- $\text{Init} = \{((v_1, v_2), (x_1, x_2)) \mid (v_1, x_1) \in \text{Init}_1 \wedge (v_2, x_2) \in \text{Init}_2\}$.
- $\text{Inv} = \{((v_1, v_2), (x_1, x_2), (u_{11}, u_{22})) \mid \exists u_1, u_2 \text{ s.t. } (v_1, x_1, u_1) \in \text{Inv}_1, (v_2, x_2, u_2) \in \text{Inv}_2, u_1 = (u_{11}, u_{12}), u_2 = (u_{22}, u_{21}), u_{11} = u_1 \setminus y_2, u_{22} = u_2 \setminus y_1, u_{12} = u_1 \cap y_2 = h_{21}(v_2, x_2), u_{21} = u_2 \cap y_1 = h_{12}(v_1, x_1)\}$.

- $E = \{e = ((v_1, v_2), (v'_1, v'_2)) \in V \times V | (v_1, v'_1) \in E_1, (v_2, v'_2) \in E_2 \text{ and } \text{Guard}(e) \neq \emptyset\} \cup \{e = ((v_1, v_2), (v'_1, v'_2)) \in V \times V | (v_1, v'_1) \in E_1, v_2 = v'_2 \text{ and } \text{Guard}(e) \neq \emptyset\} \cup \{e = ((v_1, v_2), (v'_1, v'_2)) \in V \times V | v_1 = v'_1, (v_2, v'_2) \in E_2 \text{ and } \text{Guard}(e) \neq \emptyset\}$.
- $\text{Guard} : E \rightarrow 2^{X \times U}$ which can be described as $\text{Guard}((v_1, v_2), (v'_1, v'_2)) = \{((x_1, x_2), (u_{11}, u_{22})) \in X \times U | (v_1, v'_1) \in E_1, (v_2, v'_2) \in E_2, \exists u_1, u_2 \text{ s.t. } (x_1, u_1) \in \text{Guard}_1(v_1, v'_1), (x_2, u_2) \in \text{Guard}_2(v_2, v'_2)\} \cup \{((x_1, x_2), (u_{11}, u_{22})) \in X \times U | v_1 = v'_1, (v_2, v'_2) \in E_2, \exists u_1, u_2 \text{ s.t. } (v_1, x_1, u_1) \in \text{Inv}_1, (x_2, u_2) \in \text{Guard}_2(v_2, v'_2)\} \cup \{((x_1, x_2), (u_{11}, u_{22})) \in X \times U | (v_1, v'_1) \in E_1, v_2 = v'_2, \exists u_1, u_2 \text{ s.t. } (x_1, u_1) \in \text{Guard}_1(v_1, v'_1), (v_2, x_2, u_2) \in \text{Inv}_2\}$, where $u_1 = (u_{11}, u_{12}), u_2 = (u_{22}, u_{21}), u_{11} = u_1 \setminus y_2, u_{22} = u_2 \setminus y_1, u_{12} = u_1 \cap y_2 = h_{21}(v_2, x_2), u_{21} = u_2 \cap y_1 = h_{12}(v_1, x_1)$.
- $\text{Reset} : E \times X \times U \rightarrow 2^X$ where for the composed system is defined as $\text{Reset}(((v_1, v_2), (v'_1, v'_2)), (x_1, x_2), (u_{11}, u_{22})) = \{(x'_1, x'_2) \in X | \exists u_1 = (u_{11}, u_{12}), u_2 = (u_{22}, u_{21}) \text{ s.t. } ((x_1, x_2), (u_{11}, u_{22})) \in \text{Guard}((v_1, v_2), (v'_1, v'_2)), x'_1 \in \text{Reset}_1((v_1, v'_1), x_1, u_1), x'_2 \in \text{Reset}_2((v_2, v'_2), x_2, u_2)\} \cup \{(x'_1, x'_2) \in X | \exists u_1 = (u_{11}, u_{12}), u_2 = (u_{22}, u_{21}) \text{ s.t. } ((x_1, x_2), (u_{11}, u_{22})) \in \text{Guard}((v_1, v_2), (v'_1, v'_2)), v_1 = v'_1, x'_1 = x_1, x'_2 \in \text{Reset}_2((v_2, v'_2), x_2, u_2)\} \cup \{(x'_1, x'_2) \in X | \exists u_1 = (u_{11}, u_{12}), u_2 = (u_{22}, u_{21}) \text{ s.t. } ((x_1, x_2), (u_{11}, u_{22})) \in \text{Guard}((v_1, v_2), (v'_1, v'_2)), x'_1 \in \text{Reset}_1((v_1, v'_1), x_1, u_1), v_2 = v'_2, x'_2 = x_2\}$, where $u_{11} = u_1 \setminus y_2, u_{22} = u_2 \setminus y_1, u_{12} = u_1 \cap y_2 = h_{21}(v_2, x_2), u_{21} = u_2 \cap y_1 = h_{12}(v_1, x_1)$.

Using this hybrid composition operator, the layers of the control hierarchy can be combined and their discrete transitions can be synchronized. For instance, the regu-

lation layer H_R with the coordination layer H_{P_1} , and the supervision H_{S_1} , together form a control hierarchy for which the control layers and the data flow between the layers are shown in Fig. 3.6. The discrete transitions of the layers are synchronized using the composition operator, and the composed system is shown in Fig. 3.7. This composed system helps to have more insight into the closed loop behavior by providing a detailed picture of the continuous evolution and discrete transitions of the system. Also, since most of the hybrid tools are developed for a single layer hybrid system, for this composed hybrid model of the system we can apply hybrid analysis tools such as model checking [93] and verification [94].

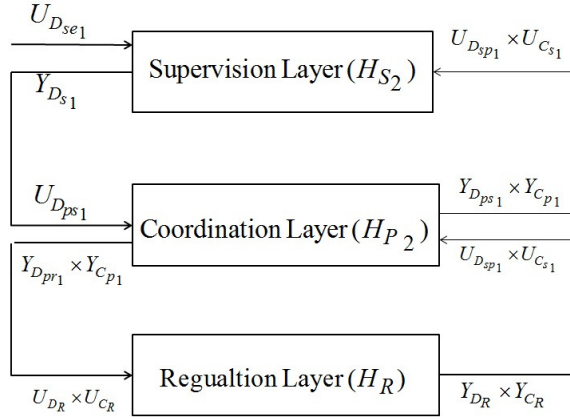


Figure 3.6: The layers of the control hierarchy.

3.6 Implementation and Experimental Results

The proposed control structure is implemented in the avionic system of the NUS UAV, and the supervision layer, H_{S_1} , together with its corresponding coordination layer,

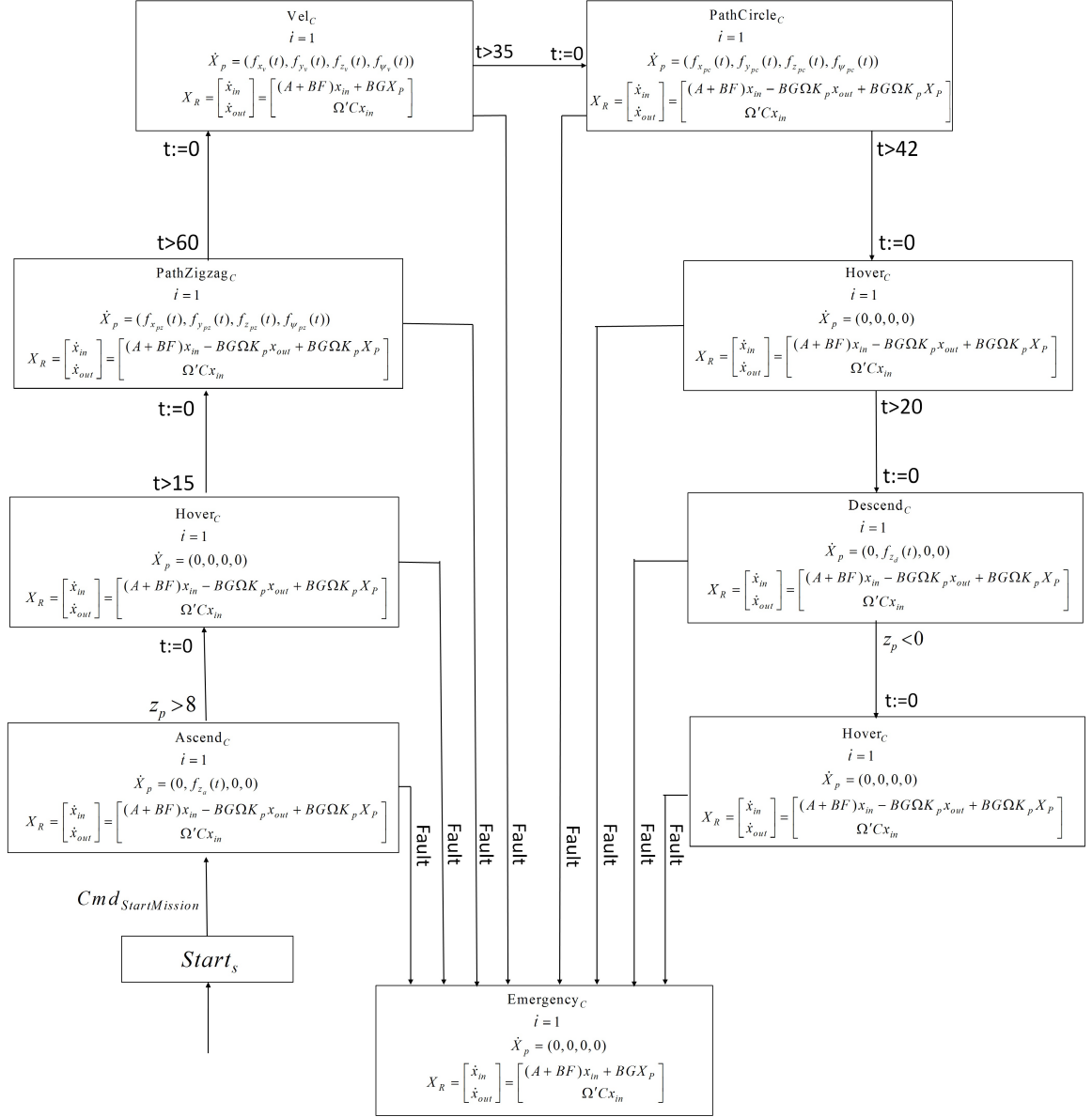


Figure 3.7: The composed system.

H_{P_1} , have been used to conduct a flight test. The assigned mission in this experiment is composed of several successive tasks. The mission starts with ascending, followed by hovering, zigzag path tracking, velocity control, circle path tracking, hovering, descending, and hovering. The state variables of the UAV are shown in Fig. 3.8. The control signals recorded in the flight test are shown in Fig. 3.9. To have a better sense of the system performance, the reference signals and actual flight test data in *Zigzag Path Tracking*, *Velocity Control*, and *Circle Path Tracking* modes are presented in Fig. 3.10. As it can be seen in this figure, the system is able to follow the given trajectory. Small deviations from the reference path could be due to the wind disturbances (around 2 to 3 m/s in the horizontal plane) and GPS signal errors as the position accuracy of GPS is 3m CEP. The video of this flight test is available at <http://uav.ece.nus.edu.sg/video/hybridswitching2.avi>.

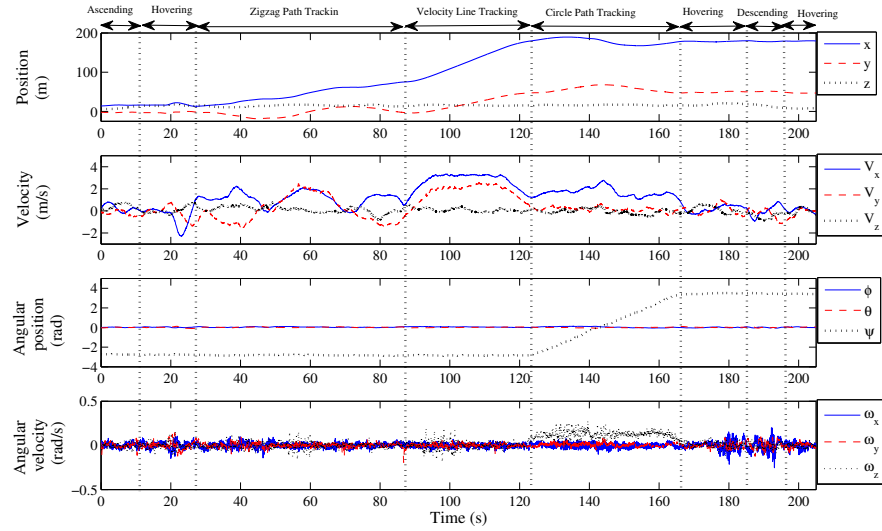


Figure 3.8: State variables of the UAV in a mission with successive tasks.

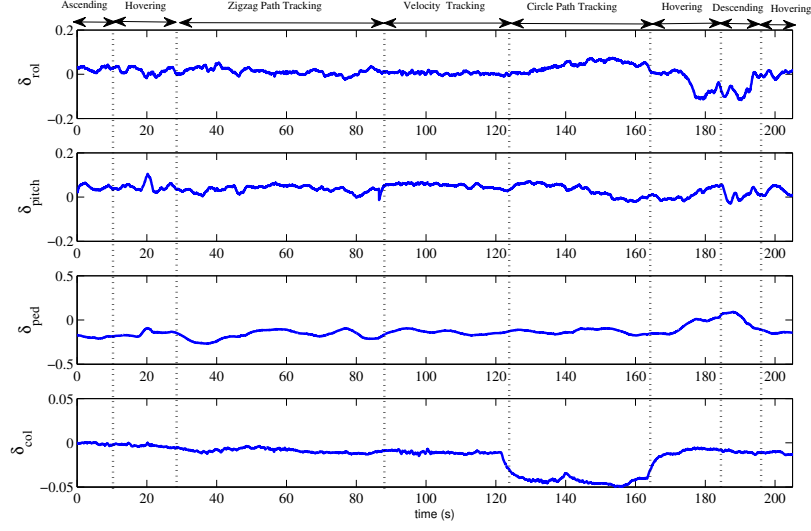
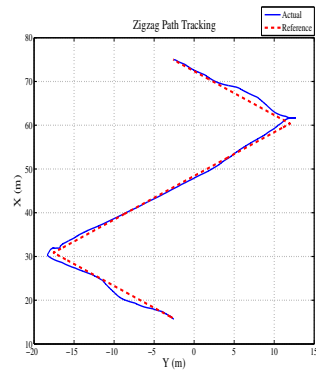


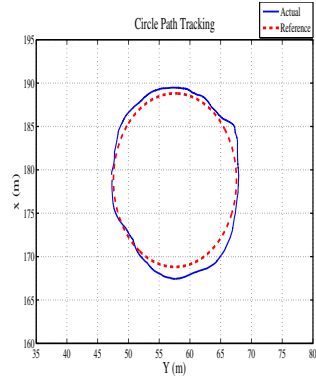
Figure 3.9: Control signals of the UAV in a mission with successive tasks.

3.7 Conclusion

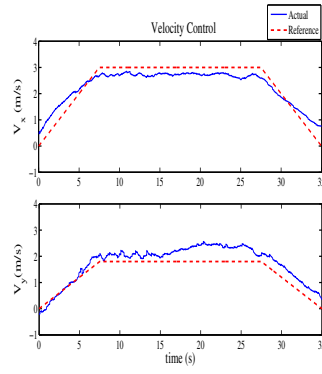
In this chapter we developed a hierarchical hybrid control structure for a UAV helicopter. This hierarchy consists of three layers: the regulation layer which is responsible for reference tracking; the coordination layer which is responsible for the path planning, and the supervision layer which is responsible for the task scheduling and decision making. Each layer was modelled by an hybrid Input/Output automaton and the discrete transitions and continuous dynamics of the system were simultaneously captured within the hybrid framework. Then, a composed hybrid operation was proposed to synchronize the layers of the control hierarchy and to obtain the whole closed-loop system. An experiment was done to verify the proposed approach in which the UAV was involved in a mission composed of several successive tasks. The experiment was successfully implemented and the actual flight test results showed the



(a)



(b)



(c)

Figure 3.10: (a) Zigzag Path Tracking (b) Circle Path Tracking (c) Velocity Control.

effectiveness of the control structure.

With the provided hybrid model although we have very detailed information from inside of the system, the closed loop system is too complicated to be used for further analysis. Therefore, a more effective computational method should be associated with this method to be able to be used for more complex tasks such as formation control. Next chapter is devoted to develop a hybrid formation algorithm which uses some techniques to abstract the motion space to simplify the system analysis.

Chapter 4

Hybrid Formation Control of Unmanned Helicopters

4.1 Introduction

In this chapter, a *hybrid supervisory control* mechanism is proposed for the UAV helicopters, involved in a leader follower formation scenario. Hybrid modeling and control [95], is a powerful framework that can capture both discrete and continuous dynamics of the system, simultaneously and collectively and provides a comprehensive analysis for interactions between the discrete part and the continuous evolution of the system. To propose a formal method for the hybrid supervisory control for a formation mission, we first introduce a new method of abstraction based on polar partitioning of the motion space. Then, we utilize 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, separate supervisors are designed for reaching the formation, keeping the formation, and collision avoidance, modularly.

After designing the DES supervisor, it is shown that due to the bisimulation relation between the plant and the abstracted system, the designed DES supervisor for the abstracted system can be applied to the original hybrid model of the plant so that the closed loop behaviour does not change. The result is then extended to a 3-D space using spherical partitioning of the motion space.

The rest of the chapter is organized as follows. First, the problem of formation control is formulated in Section 4.2, and then, Section 4.3 describes the principles of the polar partitioning of the motion space and utilizes the properties of multi-affine functions over the partitioned motion space. Using this method, the partitioned model can be abstracted into a finite state machine. In Section 4.4, the DES model of the system has been developed and then, a discrete supervisor has been designed modularly. The simulation results are presented in Section 4.5. In Section 4.6, the algorithm is extended to the 3-D space. The chapter is concluded in Section 4.7.

4.2 Problem Formulation

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 [96].

As discussed in chapter 2, the control structure of a UAV helicopter is a two-layer controller whose inner-loop stabilizes the system using H_∞ control design techniques

and its outer-loop is used to drive the system towards the desired position (Fig. 4.1). Assuming that the inner-loop is fast enough to track the given references, the outer loop dynamics can be approximately described as follows:

$$\dot{x} = u \quad x \in \mathbb{R}^2, u \in U \subseteq \mathbb{R}^2 \quad (4.1)$$

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:

$$V_{follower} = V_{leader} + V_{rel} \quad (4.2)$$

where the follower should reach and keep the formation by tuning the relative velocity V_{rel} . Alternatively, one can consider a relatively fixed frame (Fig. 4.2), 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 follower UAV as (4.1) and its velocity in the form of (4.2), 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 collision between the leader and the follower. Moreover, after reaching the formation, the follower UAV should remain at the desired position to keep the formation.*

To tackle this problem, we will propose the polar partitioning mechanism in which, as we will see, the direct path to the desired position is applicable. We also will utilize

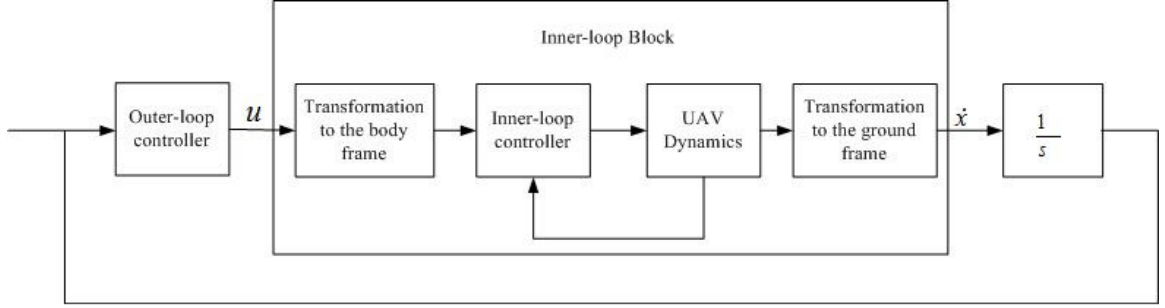


Figure 4.1: Control Structure of the UAV.

the properties of multi-affine functions over the polar partitioned space that will result in a hybrid system. This partitioned system 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 will apply the resulting supervisor to the original hybrid model of the plant, as the bisimulation relation guarantees equivalent behaviors for the DES model and its original model [68].

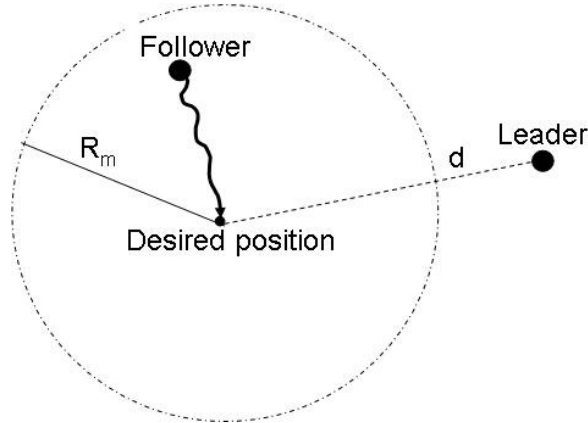


Figure 4.2: Relative frame; the follower should reach the desired position starting from any point inside the control horizon.

4.3 Polar Abstraction of the Motion Space

4.3.1 Polar Partitioning

In the polar coordinate system with $0 \leq \theta < 2\pi$, consider a circle C_{R_m} , with the radius of R_m , which has been partitioned by the curves $\{r = r_i \mid 0 \leq r_i \leq R_m, \text{ for } i < j : r_i < r_j, i = 1, \dots, n_r, r_1 = 0, r_{n_r} = R_m\}$ and $\{\theta = \theta_j \mid 0 \leq \theta_j \leq 2\pi, \text{ for } j < k : \theta_j < \theta_k, j = 1, \dots, n_\theta, \theta_1 = 0, \theta_{n_\theta} = 2\pi\}$. As an example, $\{r_i = \frac{R_m}{n_r-1}(i-1), i = 1, \dots, n_r\}$ and $\{\theta_j = \frac{2\pi}{n_\theta-1}(j-1), j = 1, \dots, n_\theta\}$ are such curves that we will use through our further derivations. Clearly, choosing large partitions reduces the maneuverability of the UAVs in the partitioned space. Therefore, it is desired to have smaller partitions. Theoretically we can choose very small partitions for the system with a mass point model, but it increases the computation cost due to the increase in the number of discrete states and also it may cause collision between two UAVs that are in adjacent regions. Therefore, the size of partitions should be bigger than the size of helicopters.

Using this notation, we will have $(n_r - 1)(n_\theta - 1)$ partitioning elements. In this partitioned space, the region $\bar{R}_{i,j} = \{p = (r, \theta) \mid r_i \leq r \leq r_{i+1}, \theta_j \leq \theta \leq \theta_{j+1}\}$ is a subset of the circle C_{R_m} surrounded by the above curves. We use the notation $R_{i,j}$ to denote the interior of the region $\bar{R}_{i,j}$. Fig. 4.3, illustrates an example of such partitioned space with $n_r = 3$ and $n_\theta = 9$.

The intersection between the element $\bar{R}_{i,j}$ and the partitioning curves generates the *vertices* and the *edges* (Fig. 4.4). We use $V(*)$ to denote the set of vertices that belong to $*$ ($*$ can be an edge, a region $R_{i,j}$, or the circle C_{R_m}), and $E(v)$ for the set

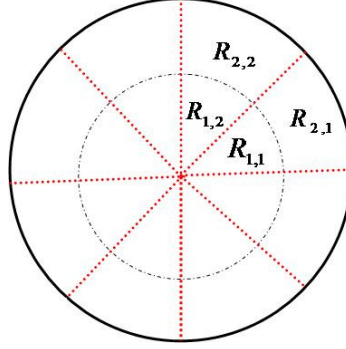


Figure 4.3: Partition labels.

of edges that vertex v belongs to them.

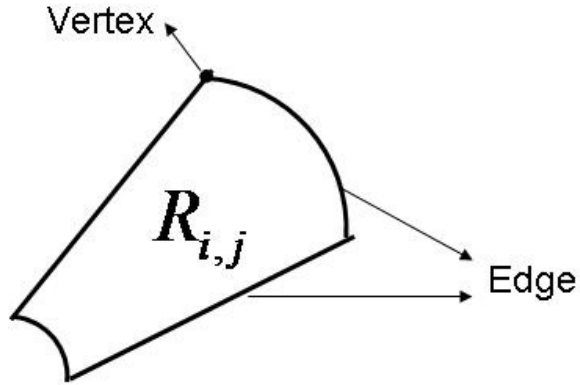


Figure 4.4: Vertices and Edges in the element $R_{i,j}$.

The vertices of the element $R_{i,j}$ are arranged as shown in Fig. 4.5. To label the vertices, consider v_m with $m = (m_\theta, m_r)_2$, where m_θ and m_r are the binary indices refer to the partitioning curves that have generated the vertex v_m . For example, if $m_r = 1$, it shows that the vertex v_m of the region $R_{i,j}$ touches the curve r_{i+1} , and if

$m_r = 0$, it touches the curve r_i . Therefore, we will have v_0 , v_1 , v_2 , and v_3 as follows:

$$\begin{cases} v_0 = v_{00} & r = r_i, \theta = \theta_j \\ v_1 = v_{01} & r = r_{i+1}, \theta = \theta_j \\ v_2 = v_{10} & r = r_i, \theta = \theta_{j+1} \\ v_3 = v_{11} & r = r_{i+1}, \theta = \theta_{j+1} \end{cases} \quad (4.3)$$

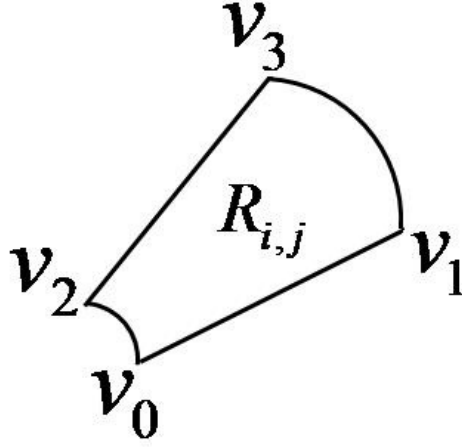


Figure 4.5: Vertices of the element $R_{i,j}$.

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. 4.6).

Remark 3 The element with $r_i = 0$ (Fig. 4.7), 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.

In the circle C_{R_m} , let's define \bar{C} as the circle's perimeter, and V as the set of all vertices. Also, consider the *detection element* $d([i, j], [i', j']) = \bar{R}_{i,j} \cap \bar{R}_{i',j'} - V$, which is defined for two regions $R_{i,j}$ and $R_{i',j'}$ that are adjacent in a common edge.

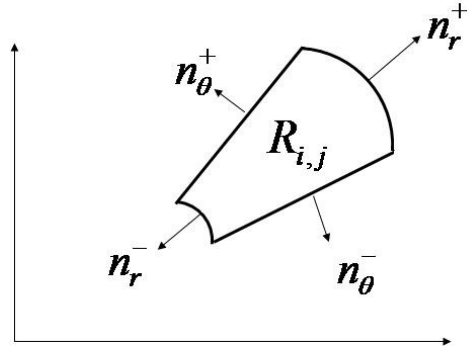


Figure 4.6: Outer normals of the element $R_{i,j}$.

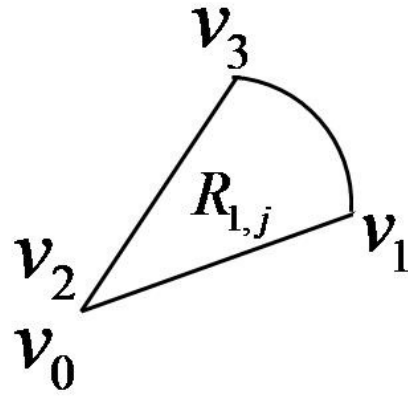


Figure 4.7: $R_{1,j}$ is a special case of the element $R_{i,j}$.

Indeed, the detection elements are the edges in which the vertices are excluded. With this procedure, the circle C_{R_m} has been partitioned into $V \cup R_{i,j} \cup d([i, j], [i', j']) \cup C$, where $1 \leq i, i' \leq n_r - 1$, $1 \leq j, j' \leq n_\theta - 1$ and $C = \bar{C} - V$. Correspondingly, consider \tilde{V} , $\tilde{R}_{i,j}$, $\tilde{d}([i, j], [i', j'])$, and \tilde{C} as the labels for these partitioning elements, where $\mathfrak{S}(\tilde{r}) = r$ relates the label \tilde{r} to the set r . This partitioned space can be captured by the equivalence relation $Q = \{(x_1, x_2) | \exists \tilde{r} \in \{\tilde{V}, \tilde{R}_{i,j}, \tilde{d}([i, j], [i', j']), \tilde{C}\} \text{ s.t. } x_1, x_2 \in \mathfrak{S}(\tilde{r}) \text{ and } 1 \leq i, i' \leq n_r - 1, 1 \leq j, j' \leq n_\theta - 1\}$. Correspondingly, the projection map $\pi_Q(x)$ shows the partitioning element that x belongs to it: $\pi_Q(x) = \tilde{r} \in \{\tilde{V}, \tilde{R}_{i,j}, \tilde{d}([i, j], [i', j']), \tilde{C}\} \text{ s.t. } x \in r \text{ and } \mathfrak{S}(\tilde{r}) = r$, where $1 \leq i, i' \leq n_r - 1$, $1 \leq j, j' \leq n_\theta - 1$.

4.3.2 Properties of Multi-affine Functions over the Partitioned Space

In this section we explain the properties of multi-affine vector fields over the polar partitioned space. Multi-affine functions are defined as follows:

Definition 3 *Multi-affine function* [27]

A function $g = (g_1, g_2, \dots, g_m) : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is said to be multi - affine, if for all $g_i : \mathbb{R}^n \rightarrow \mathbb{R}$, $i = 1, \dots, m$, and for every $a_1, a_2 \geq 0$ satisfying $a_1 + a_2 = 1$, the following equality holds:

$$g_i(x_1, \dots, (a_1 x_{k_1} + a_2 x_{k_2}), \dots, x_n) = a_1 g_i(x_1, \dots, x_{k_1}, \dots, x_n) + a_2 g_i(x_1, \dots, x_{k_2}, \dots, x_n).$$

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 theorem, a multi-affine function over the element $R_{i,j}$ can be uniquely expressed in terms of the values of the function at the vertices of $R_{i,j}$.

Theorem 1 *For a multi-affine function $g(x) : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ over the region $R_{i,j}$, the following property always holds true:*

$$\forall x = (r, \theta) \in \bar{R}_{i,j} : g(x) = \sum_{m=0}^3 \lambda_m g(v_m) \quad (4.4)$$

where $v_m \in V(R_{i,j})$, $m = 0, 1, 2, 3$, are the vertices of the element $R_{i,j}$ and λ_m , $m = 0, 1, 2, 3$, are obtained as follows:

$$\lambda_m = \lambda_r^{m_r} (1 - \lambda_r)^{1-m_r} \lambda_\theta^{m_\theta} (1 - \lambda_\theta)^{1-m_\theta} \quad (4.5)$$

where m_r and m_θ are the corresponding binary digits of the index m , 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: $r_i \leq r \leq r_{i+1}$ and $\theta_j \leq \theta \leq \theta_{j+1}$. Hence, r and θ can be written affinely as follows:

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

Now, consider a trajectory starting from the vertex v_0 to the point x , only along the polar directions. As an example, $v_0 = (r_i, \theta_j) \xrightarrow{\text{step1}} x_1 = (r, \theta_j) \xrightarrow{\text{step2}} x_2 = x = (r, \theta)$ 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 of multi-affine functions, since in Step 1, the parameter θ is fixed and only r changes:

$$\begin{aligned} g(x_1) &= g_\theta|_{\theta_j}(r) = g_\theta|_{\theta_j}((1 - \lambda_r)r_i + \lambda_r r_{i+1}) = (1 - \lambda_r)g_\theta|_{\theta_j}(r_i) + \lambda_r g_\theta|_{\theta_j}(r_{i+1}) = \\ & (1 - \lambda_r)g(r_i, \theta_j) + \lambda_r g(r_{i+1}, \theta_j) \end{aligned} \quad (4.6)$$

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

$$\begin{aligned} g(x_2) &= g_r|_r(\theta) = g_r|_r((1 - \lambda_\theta)\theta_j + \lambda_\theta \theta_{j+1}) = (1 - \lambda_\theta)g_r|_r(\theta_j) + \lambda_\theta g_r|_r(\theta_{j+1}) = (1 - \lambda_\theta) \\ & g(r, \theta_j) + \lambda_\theta g(r, \theta_{j+1}) = (1 - \lambda_\theta)g_\theta|_{\theta_j}(r) + \lambda_\theta g_\theta|_{\theta_{j+1}}(r) \end{aligned} \quad (4.7)$$

In (4.6), 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 (4.7), we will obtain $g(x_2)$ as follows:

$$\begin{aligned} g(x_2) &= (1 - \lambda_\theta)[(1 - \lambda_r)g(r_i, \theta_j) + \lambda_r g(r_{i+1}, \theta_j)] + \lambda_\theta[(1 - \lambda_r)g(r_i, \theta_{j+1}) + \\ & \lambda_r g(r_{i+1}, \theta_{j+1})] = (1 - \lambda_\theta)(1 - \lambda_r)g(v_0) + (1 - \lambda_\theta)\lambda_r g(v_1) + \lambda_\theta(1 - \lambda_r)g(v_2) + \lambda_\theta\lambda_r g(v_3) \end{aligned} \quad (4.8)$$

which is equivalent to (4.4). ■

Remark 4 *It can be verified that for all $x = (r, \theta) \in R_{i,j}$, the resulting coefficients λ_m , $m = 0, \dots, 3$, have the property that $\lambda_m \geq 0$ and $\sum_{m=0}^3 \lambda_m = 1$.*

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:*

$$\forall x = (r, \theta) \in E_q^s : g(x) = \sum_{v_m \in V(E_q^s)} \lambda_m g(v_m), \quad v_m \in V(E_q^s) \quad (4.9)$$

where λ_m can be obtained as follows:

- For the edges E_r^+ and E_r^- : $\lambda_m = \lambda_\theta^{m_\theta} (1 - \lambda_\theta)^{m_\theta}$
- For the edges E_θ^+ and E_θ^- : $\lambda_m = \lambda_r^{m_r} (1 - \lambda_r)^{1-m_r}$

Proof: This is a special case of Theorem 1 and for the proof, it just needs to follow Step 1 of the proof of the previous Theorem. ■

In the following proposition, we will show that the coefficients in Theorem 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 $g(v_m)$ at the vertices of $R_{i,j}$.

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

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

which contradicts with the assumption. ■

4.3.3 Control over the Partitioned Space

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 deterministically leave through a particular edge (*Exit edge*).

4.3.3.1 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 4 (*Invariant region*)

In the circle C_{R_m} and the vector field $\dot{x} = g(x)$, $g : \mathbb{R}^2 \rightarrow \mathbb{R}^2$, the region $R_{i,j}$ is said to be invariant region (Fig. 4.8), if $\forall x(0) \in \text{int}(R_{i,j})$, $x(t) \in R_{i,j}$ for $t \geq 0$.

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

Theorem 2 (*Sufficient condition for $R_{i,j}$ to be an invariant region*) For a continuous multi-affine vector field $\dot{x} = g(x)$, $g : \mathbb{R}^2 \rightarrow \mathbb{R}^2$, the element $R_{i,j}$ is an

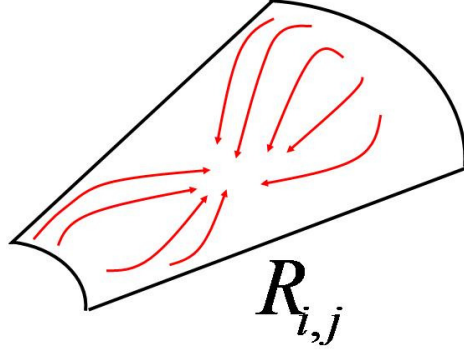


Figure 4.8: Invariant region.

invariant region if for each edge E_q^s and its corresponding outer normal n_q^s , $q \in \{r, \theta\}$ and $s \in \{+, -\}$, we have:

$$n_q^s(y)^T \cdot g(v_m) < 0 \quad \forall v_m \in V(E_q^s), \text{ and } \forall y \in F_q^s \quad (4.10)$$

Proof: According to the Corollary 1, for each E_q^s , we have the property that $\forall y = (r, \theta) \in E_q^s : g(y) = \sum \lambda_m g(v_m)$, $v_m \in V(E_q^s)$, where $\lambda_m \geq 0$ and $\sum \lambda_m = 1$. Hence, knowing that $n_q^s(y)^T \cdot g(v_i) < 0$ for all $v_m \in V(E_q^s)$ will result in $n_q^s(y)^T \cdot g(y) < 0$, for all $y \in E_q^s$. Since $g(x)$ is continuous, it will be concluded that $n_q^s(y)^T \cdot g(y) < 0$, for the neighborhood of all $y \in E_q^s$. Therefore, the trajectories of the system cannot touch the edge E_q^s . In other words, 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 $\dot{x} = h(x, u(x)) = g(x)$, $g : \mathbb{R}^2 \rightarrow \mathbb{R}^2$, $R_{i,j}$ is an invariant region if there

exists a controller $u : \mathbb{R}^2 \rightarrow U \subseteq \mathbb{R}^2$, such that for each vertex v_m , $m = 0, 1, 2, 3$, with incident edges $E_q^s \in E(v_m)$, and corresponding outer normals n_q^s , $q \in \{r, \theta\}$ and $s \in \{+, -\}$:

$$U_m(\text{Inv}(R_{i,j})) = U \cap \left\{ u \in \mathbb{R}^2 \mid n_q^s(y)^T \cdot g(v_m) < 0, \text{ for all } E_q^s \in E(v_m), \text{ and } \forall y \in E_q^s \right\} \neq \emptyset \quad (4.11)$$

where the convex set U represents the velocity bounds. ■

Corollary 2 is the direct implication of Theorem 2. 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 $U_m \neq \emptyset$ for $m = 0, 1, 2, 3$, this controller is feasible. We will use the notation C_0 to label this controller.

Remark 5 In the case that all $U_m \neq \emptyset$, it is sufficient to pick an arbitrary value for $u(v_m)$ from the set U_m . In this situation, $g(x) = h(x, u(x))$, as a multi-affine function, satisfies the conditions of Theorem 2 and makes $R_{i,j}$ an invariant region. In addition, using Theorem 1, if we construct u 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 point $x \in R_{i,j}$ is uniquely determined by the value of u at the vertices of the region $R_{i,j}$.

4.3.3.2 Controller for an Exit Edge

When all of the trajectories of the system leave the region from a specific edge, this edge is called the exit edge and formally can be defined as follows:

Definition 5 (*Exit edge*)

In the circle C_{R_m} and the vector field $\dot{x} = g(x)$, $g : \mathbb{R}^2 \rightarrow \mathbb{R}^2$, the edge E_q^s , $q \in \{r, \theta\}$ and $s \in \{+, -\}$, is said to be an exit edge (Fig. 4.9), if $\forall x(0) \in \text{int}(R_{i,j})$, there exist $\tau(\text{finite}) > 0$ and $\tau_d > 0$ satisfying:

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

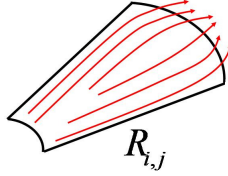


Figure 4.9: Exit edge.

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

Theorem 3 (*Sufficient condition for an exit edge*) For a continuous multi-affine vector field $\dot{x} = g(x)$, $g(x) : \mathbb{R}^2 \rightarrow \mathbb{R}^2$, the edge E_q^s with the outer normal n_q^s , is an exit edge if :

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

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 2. The second requirement is to derive the trajectories of the system out through the edge E_q^s . According to Theorem 1, for the multi-affine function g , there exist λ_m such that $\forall x \in \bar{R}_{i,j} : g(x) = \sum_m \lambda_m g(v_m)$, $m = 0, 1, 2, 3$. Since $\lambda_m \geq 0$ and $\sum_m \lambda_m = 1$, then $n_q^s(y)^T \cdot \lambda_m g(v_m) > 0$ for all v_m . This will lead to have $n_q^s(y)^T \cdot g(x) > 0$ for all $x \in \bar{R}_{i,j}$, which 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 $\dot{x} = h(x, u(x)) = g(x)$, $g : \mathbb{R}^2 \rightarrow \mathbb{R}^2$, 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 $u : \mathbb{R}^2 \rightarrow U \subseteq \mathbb{R}^2$, such that for each vertex v_m , $m = 0, 1, 2, 3$, the following property holds true:

$$U_m(Ex(E_q^s(R_{i,j}))) = U \cap \{n_q^s(y)^T \cdot g(v_m) > 0 \forall v_m \in R_{i,j}, \text{ and } \forall y \in E_q^s\} \cap \{n_{q'}^{s'}(y)^T \cdot g(v_m) < 0 \forall E_{q'}^{s'} \neq E_q^s, \forall y \in E_{q'}^{s'}, \text{ and } \forall v_m \in V(E_{q'}^{s'})\} \neq \emptyset \quad (4.12)$$

where the convex set U represents the velocity bounds. ■

Corollary 3 is the direct implication of Theorem 3 for which we check the feasibility of the feedback controller, applying the velocity constraints U . Clearly, if $U_m \neq \emptyset$, $m = 1, \dots, 3$, the controller is feasible and we will use C_q^s to label it. 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.

Here, we will present some results about the properties of the exit edge controller that will be used through our further derivations.

Lemma 1 *For a multi-affine vector field $\dot{x} = g(x)$, $g : C_{R_m} \rightarrow \mathbb{R}^2$, in region $R_{i,j}$ with the exit edge E_q^s , constructed by Theorem 3, following properties are concluded:*

1. *The trajectories that leave the region do not return back any more.*
2. *The points on the exit edge are not reachable from other points on the edge.*
3. *The trajectory that has reached the exit edge, leaves it immediately.*

proof: As observed in the proof of Theorem 3, respecting the second condition of this theorem leads to have $n_q^s(y)^T \cdot g(x) > 0$ for any $x \in \bar{R}_{i,j}$ and any $y \in E_q^s$. In particular this is true for the points on Edge E_q^s : $n_q^s(y)^T \cdot g(y) > 0, \forall y \in E_q^s$. This strictly positive inequality guarantees that the trajectory reaches the exit edge, leaves it upon reaching the edge so that it can neither move along the edge nor return back. ■

Lemma 2 *For a multi-affine vector field $\dot{x} = g(x)$, $g : C_{R_m} \rightarrow \mathbb{R}^2$, in region $R_{i,j}$ with the exit edge E_q^s constructed by Theorem 3, the trajectories that leave the region only can pass through the detection elements.*

proof: Strictly negative inequalities in Condition 1 of Theorem 3 guarantees that the trajectory of the system cannot pass through the non exit edges. Therefore, the trajectory of the system cannot pass through the vertices common between the exit edge and non exit edges. Hence, the only way is that the trajectory leaves the region through the detection elements. ■

Proposition 2 *For a multi-affine vector field $\dot{x} = g(x)$, $g : C_{R_m} \rightarrow \mathbb{R}^2$, in region $R_{i,j}$ with the exit edge E_q^s constructed by Theorem 3, all $y \in E_q^s$ are reachable from a point inside region $R_{i,j}$.*

proof: As the system dynamics is continuous Lipschitz, and since any $y \in E_q^s$ is not reachable from the adjacent region (Part 1, Lemma 1) or from another point on E_q^s (Part 2, Lemma 1), then there is a point inside region $R_{i,j}$ on the neighborhood of y from which y is reachable. ■

4.3.4 Abstraction of the Motion Space

Now, consider the original system which is defined over the partitioned space. The equivalence relation Q , defined in Section 4.3.1, describes this partitioned space. The system over the partitioned space can be captured by a transition system $T_Q = (X_Q, X_{Q_0}, U_Q, \rightarrow_Q, Y_Q, H_Q)$, where

- $X_Q = V \cup R_{i,j} \cup d([i, j], [i', j']) \cup S$ is the set of system states, where $1 \leq i, i' \leq n_r - 1, 1 \leq j, j' \leq n_\theta - 1$.
- X_{Q_0} is the set of initial states. Assuming that the system initially starts from inside one of the regions $R_{i,j}$, $X_{Q_0} = \bigcup R_{i,j}$, where $1 \leq i, i' \leq n_r - 1, 1 \leq j, j' \leq n_\theta - 1$.
- $U_Q = U_a \cup U_d$, where
 - $U_a = \{C_r^+, C_r^-, C_\theta^+, C_\theta^-, C_0\}$ is the set of labels corresponding to the controllers that can make the region $R_{i,j}$ an invariant region or can make

one of its edges an exit edge. For these control labels, as discussed in Section 4.3.3, the sets of control actions that can be activated in this region are : $r(C_q^s) = \{u(x) | u(x) = \sum_m \lambda_m u(v_m), m = 0, 1, \dots, 3, v_m \in V(R_{i,j}), u(v_m) \in U_m(Ex(E_q^s))\}$, and $r(C_0) = \{u(x) | u(x) = \sum \lambda_m u(v_m), v_m \in V(R_{i,j}), u(v_m) \in U_m(Inv(R_{i,j}))\}$, where λ_m can be obtained by (4.5).

– $U_d = U_c \cup U_e$ is the set of the detection events, where $U_c = \{\hat{d}([i, j], [i', j']) | 1 \leq i, i' \leq n_r - 1, 1 \leq j, j' \leq n_\theta - 1\}$. Here, $\hat{d}([i, j], [i', j'])$ is an event that shows the detection element $d([i, j], [i', j'])$ has been crossed and U_e is the set of external events such as entering an alarm zone of collision.

- $(x, x', v) \in \rightarrow_Q$, denoted by $x \xrightarrow{v}_Q x'$, if and only if one of the following conditions holds true:

1. Actuation:

– Exit edge: $v \in \{C_q^s | q \in \{r, \theta\}, s \in \{+, -\}\}$; $\pi_Q(x) \neq \pi_Q(x')$; $\exists \tilde{R}_{i,j}$ and $\tilde{d}([i, j], [i', j'])$ such that $\pi_Q(x) = \tilde{R}_{i,j}$ and $\pi_Q(x') = \tilde{d}([i, j], [i', j'])$; $\exists \tau(\text{finite})$ and $\tau_d > 0$ such that $\psi(t) : [0, \tau + \tau_d] \rightarrow \mathbb{R}^2$ is the solution of $\dot{x} = h(x, r(v))$, $\psi(0) = x$; $\psi(\tau) = x'$, $\pi_Q(\psi(t)) = \pi_Q(x)$ for $t \in [0, \tau)$, and $\pi_Q(\psi(t)) \neq \pi_Q(x)$ for $t \in [\tau, \tau + \tau_d]$. Here, $r(v)$ is the continuous controller corresponding to the control label v , which can be constructed as discussed above.

– Invariant region: $v = C_0$; $\exists \tilde{R}_{i,j}$ such that $\pi_Q(x) = \pi_Q(x') = \tilde{R}_{i,j}$; $\psi(t) : \mathbb{R}^+ \rightarrow \mathbb{R}^2$ is the solution of $\dot{x} = h(x, r(v))$, $\psi(0) =$

x , $\psi(\tau) = x'$, and $\pi_Q(\psi(t)) = \pi_Q(x)$ for all $t \geq 0$.

2. Detection:

- Crossing a detection element: $v \in U_c$; $\pi_Q(x) \neq \pi_Q(x')$;
 $\exists \tilde{R}_{i,j}, \tilde{R}_{i',j'}, \tilde{d}([i,j], [i',j'])$ such that $\pi_Q(x) = \tilde{d}([i,j], [i',j'])$ and
 $\pi_Q(x') = \tilde{R}_{i',j'}$; $\exists \varepsilon > 0$, $\tau_d > 0$ and $\exists w \in \{C_q^s | q \in \{r, \theta\}, s \in \{+, -\}\}$ such that $\psi(t) : [0, \tau_d + \varepsilon] \rightarrow \mathbb{R}^2$ is the solution of $\dot{x} = h(x, r(w))$, $\psi(\varepsilon) = x$; $\psi(\tau_d + \varepsilon) = x'$, $\pi_Q(\psi(t)) = \tilde{R}_{i,j}$ for $t \in (0, \varepsilon)$,
and $\pi_Q(\psi(t)) = \tilde{R}_{i',j'}$ for $t \in (\varepsilon, \varepsilon + \tau_d]$.
- External events: $v \in U_e$, and $x = x'$. In this case, x is the value of the system state at the time instant that the event v appears. External events do not affect the system dynamics.

- $Y_Q = X_Q$ is the output space.
- $H_Q : X \rightarrow Y_Q$ is the output map. Here, we have chosen $H_Q(x) = \pi_Q(x)$.

Although T_Q contains only important transitions that either cross the boundaries or remain inside the regions, still it has infinite number of states and the analysis of such a system might be difficult. Abstraction [68], is the technique that can reduce the complexity and can lead to a finite state machine for which the DES supervisory control tools can be used for the system analysis and control synthesis. For this purpose, each partitioning element, can be considered one state in the abstracted model. Hence, the abstract model is a tuple $T_\xi = (X_\xi, X_{\xi_0}, U_\xi, \rightarrow_\xi, Y_\xi, H_\xi)$, where

- $X_\xi = \{\tilde{R}_{i,j} | 1 \leq i \leq n_r - 1, 1 \leq j \leq n_\theta - 1\} \cup \{\tilde{d}([i,j], [i',j']) | 1 \leq i, i' \leq n_r - 1, 1 \leq j, j' \leq n_\theta - 1\}$, where $\tilde{R}_{i,j}$ and $\tilde{d}([i,j], [i',j'])$ are the labels for

the regions $R_{i,j}$ and $d([i,j],[i',j'])$, respectively. Note that since the system starts from a point inside the regions $R_{i,j}$ and never crosses the vertices (see Lemma 2), the set V does not need to be considered in the abstracted system. Moreover, as the circle C_{R_m} is the control horizon, its perimeter, C , should not be crossed.

- $X_{\xi_0} = \{\tilde{R}_{i,j} \mid 1 \leq i, i' \leq n_r - 1, 1 \leq j, j' \leq n_\theta - 1\}$.
- $U_\xi = U_a \cup U_d$ is like what we defined in T_Q .
- $(r, r', v) \in \rightarrow_\xi$, denoted by $r \xrightarrow{v}_\xi r'$, if $\exists v \in U_\xi, x \in \mathfrak{S}(r), x' \in \mathfrak{S}(r')$ such that $x \xrightarrow{v}_Q x'$.
- $Y_\xi = X_\xi$.
- $H_\xi(r) = r$ is the output map, which is selected as an identity map.

In general, the abstract model contains all of the behaviors of the partitioned system, however, the converse might not be always true. If the converse is also true, we say that they are bisimilar. A bisimulation relation between two transition systems can be formally defined as follows:

Definition 6 [68] *Given $T_i = (Q_i, Q_i^0, U_i, \rightarrow_i, Y_i, H_i)$, ($i = 1, 2$), R is a bisimulation relation between T_1 and T_2 , denoted by $T_1 \approx_R T_2$, iff:*

1. $\forall q_1 \in Q_1^0$ then $\exists q_2 \in Q_2^0$ that $(q_1, q_2) \in R$. Also, $\forall q_2 \in Q_2^0$ then $\exists q_1 \in Q_1^0$ that $(q_1, q_2) \in R$.
2. $\forall q_1 \rightarrow_1 q'_1$, and $(q_1, q_2) \in R$ then $\exists q'_2 \in Q_2$ such that $q_2 \rightarrow_2 q'_2$ and $(q'_1, q'_2) \in R$.
Also, $\forall q_2 \rightarrow_2 q'_2$, and $(q_1, q_2) \in R$ then $\exists q'_1 \in Q_1$ such that $q_1 \rightarrow_1 q'_1$ and

$$(q'_1, q'_2) \in R.$$

Theorem 4 *The original partitioned system, T_Q , and the abstract model, T_ξ , are bisimilar.*

Proof: See the Appendix for the proof. ■

This bisimulation relation ensures us that the abstract model, T_ξ , and the original system, T_Q , behave exactly the same so that for the control synthesis, we can use the abstract model with finite number of states instead of the original partitioned system with infinite states, as we will do in the next sections.

4.4 Hybrid Supervisory Control of the Plant

4.4.1 DES Model of the Plant

The finite state machine T_ξ can be formally presented by an automaton $G = (X, \Sigma, \alpha, X_0, X_m)$, where $X = X_\xi$ is the set of states; $X_0 = X_{\xi_0} \subseteq X$ is the set of initial states; $X_m = \{\tilde{R}_{1,j} | 1 \leq j \leq n_\theta - 1\}$ is the set of final (marked) states. Σ is the finite set of events. The sequence of these events forms a string. We use ε to denote an empty string, while Σ^* is as the set of all possible strings over the set Σ including ε . The language of the automaton G , denoted by $L(G)$, is the set of all strings that can be generated by G , starting from the initial states. The marked language, $L_m(G)$, is the set of strings that belong to $L(G)$ and end with the marked states. $L(G(x_0))$ is the set of strings that belong to $L(G)$ and start from the initial state x_0 . \bar{L} is the set of all prefixes to the strings that belong to the language L .

Here, the event set Σ consists of the actuation events $U_a = \{C_q^s | q \in \{r, \theta\}, s \in \{+, -\}\} \cup \{C_0\}$, the crossing detection events $U_c = \{\hat{d}([i, j], [i', j']) | 1 \leq i, i' \leq n_r - 1, 1 \leq j, j' \leq n_\theta - 1\}$, and the external event set U_e . The set $U_e = \{C\}$ contains the external event C which alarms the UAV that has entered the alarm zone and there is a danger of collision. The event set Σ consists of the controllable event set $\Sigma_c = U_a$ and uncontrollable event set $\Sigma_{uc} = U_d = U_c \cup U_e$. The uncontrollable events are those that cannot be affected by the supervisor. In automaton G , $\alpha : X \times \Sigma \rightarrow X$ is the transition function, which is a partial function and determines the possible transitions in the system caused by an event. This function is corresponding to \rightarrow_ξ in T_ξ , so that for any $r \xrightarrow{v}_\xi r'$ we have $\alpha(r, v) = r'$. Based on the definition of T_ξ and the constructed controllers $C_0, C_r^+, C_r^-, C_\theta^+, C_\theta^-$, we have:

$$\alpha(\tilde{R}_{i,j}, \sigma) = \begin{cases} \tilde{R}_{i,j} & \sigma = C_0 \\ \tilde{R}_{i,j} & \sigma \in U_e \text{ for } i \neq 1 \\ \tilde{d}([i, j], [i+1, j]) & \sigma = C_r^+ \text{ for } i \neq n_r - 1 \\ \tilde{d}([i, j], [i-1, j]) & \sigma = C_r^- \text{ for } i \neq 1 \\ \tilde{d}([i, j], [i, j+1]) & \sigma = C_\theta^+ \text{ for } j \neq n_\theta - 1 \\ \tilde{d}([i, j], [i, 1]) & \sigma = C_\theta^+ \text{ for } j = n_\theta - 1 \\ \tilde{d}([i, j], [i, j-1]) & \sigma = C_\theta^- \text{ for } j \neq 1 \\ \tilde{d}([i, j], [i, n_\theta - 1]) & \sigma = C_\theta^- \text{ for } j = 1 \end{cases}$$

$$\alpha(\tilde{d}([i, j], [i', j']), \sigma) = \tilde{R}_{i',j'} \quad \sigma = \hat{d}([i, j], [i', j'])$$

Some parts of the graph representation of the system automaton are shown in Fig.

4.10. In this automaton, the arrows starting from one state and ending to another state represent the transitions, labeled by the events belong to Σ . The states with an entering arrow stand for the initial states. As it is shown in Fig. 4.10, the system could start from any of the states $\tilde{R}_{i,j}$.

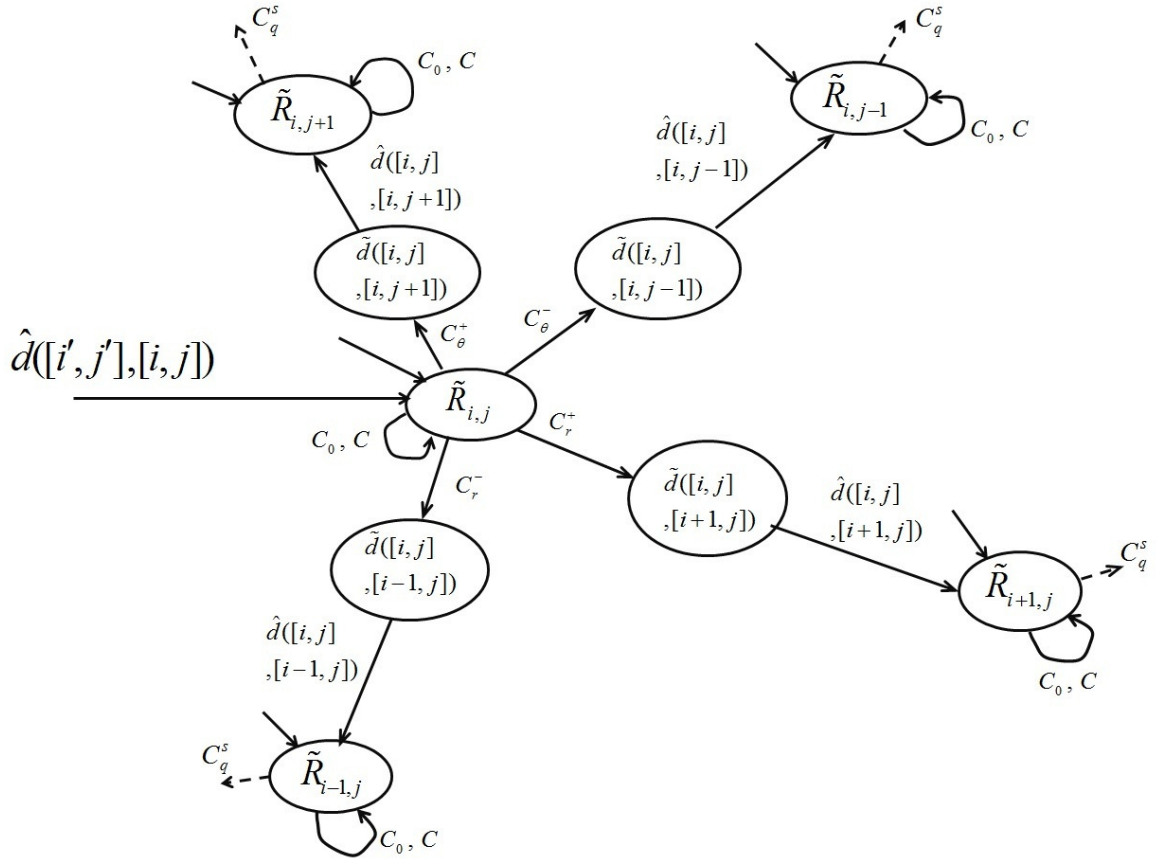


Figure 4.10: DES model of the plant.

In this model, when the system is in the state $\tilde{R}_{1,j}$, the command C_r^- leads the system to an unknown region and the system would become nondeterministic. Hence, as it is reflected in the expression of α , the command C_r^- cannot be activated in this region. Moreover, in the region $\tilde{R}_{n_r-1,j}$ the command C_r^+ cannot be activated

as it leads the system outwards the control horizon. These restrictions are already considered in the definition of α .

4.4.2 Design of the Supervisor

The logical behavior of the system can be modified by a discrete supervisor to achieve a desired order of events. Indeed, the supervisor, S , observes the executed events of the plant G and disables the undesirable controllable strings. Here, we assume that all of the events are observable. The language and marked language of the closed-loop system, $L(S/G)$ and $L_m(S/G)$, can be constructed as follows:

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

where s is the string that has been generated so far, and σ is an event which the supervisor should decide whether keep it active or not.

Within this framework, we can use the parallel composition to facilitate the control synthesis. Parallel composition is a binary operation between two automata. Here, the parallel composition is used to combine the plant discrete model and the supervisor.

Definition 7 (*Parallel Composition*) [97] *Given $G = (X_G, \Sigma_G, \alpha_G, x_{0_G}, X_{m_G})$ and $S = (X_S, \Sigma_S, \alpha_S, x_{0_S}, X_{m_S})$, $G_{cl} = G \parallel S = (X_{cl}, \Sigma_{cl}, \alpha_{cl}, x_{0_{cl}}, X_{m_{cl}})$ is said to be the parallel composition of G and S with $X_{cl} = X_G \times X_S$, $\Sigma_{cl} = \Sigma_G \cup \Sigma_S$, $x_{0_{cl}} = (x_{0_G}, x_{0_S})$, $X_{m_{cl}} = X_{m_G} \times X_{m_S}$, and $\forall x = (x_1, x_2) \in X_{cl}, \sigma \in \Sigma_{cl}$, then $\alpha_{cl}(x, \sigma) =$*

$$\left\{ \begin{array}{l} \bullet(\alpha_G(x_1, \sigma), \alpha_S(x_2, \sigma)) \\ \quad \text{if } \alpha_G(x_1, \sigma)! \text{ and } \alpha_S(x_2, \sigma)! \text{ and } \sigma \in \Sigma_G \cap \Sigma_S \\ \bullet(\alpha_G(x_1, \sigma), x_2) \quad \text{if } \alpha_G(x_1, \sigma)! \text{ and } \sigma \in \Sigma_G - \Sigma_S \\ \bullet(x_1, \alpha_S(x_2, \sigma)) \quad \text{if } \alpha_S(x_2, \sigma)! \text{ and } \sigma \in \Sigma_S - \Sigma_G \\ \bullet \text{undefined} \quad \text{otherwise} \end{array} \right.$$

where $\alpha_*(x, \sigma)!$ shows the existence of a transition from the state x by the event σ in system $*$. In this definition, the initial conditions of these automata were assumed to be the states x_{0_G} and x_{0_S} . Extending this definition to the case that the automata G and S have the initial state sets X_{0_G} and X_{0_S} , the initial state set of the composed system will be $X_{0_{cl}} = \Upsilon(X_{0_G}, X_{0_S}) \subseteq X_{0_G} \times X_{0_S}$, where the relation Υ describes the initial states in G and S that are coupled to synchronously generate a string in the composed system.

In fact, parallel composition synchronizes operand systems on their common events; however, their private events can transit independently. Next lemma and Corollary use the parallel composition of the plant and the supervisor to obtain the closed-loop system.

Lemma 3 [\[97\]](#)

Let $G = (X, \Sigma, \alpha, x_0, X_m)$, be the plant automaton with the initial state of x_0 and $K \subseteq \Sigma^$ be a desired language. There exists a nonblocking supervisor S such that $L(S/G) = L(S||G) = K$ if $\emptyset \neq K = \bar{K} \subseteq L(G)$ and K is controllable. In this case, S could be any automaton with $L(S) = L_m(S) = K$.*

Using the above lemma and following the definition of the parallel composition, the result can be extended to a plant with several initial states:

Corollary 4 *Let $G = (X, \Sigma, \alpha, X_0, X_m)$ be the plant with the initial state set $X_0 = \{x_0^1, x_0^2, \dots\}$ and $K = \bigcup K_i \subseteq \Sigma^*$ be a desired language where K_i is the desired language that should be generated starting from x_0^i . If $\emptyset \neq K_i = \bar{K}_i \subseteq L(G(x_0^i))$ and K_i is controllable for all $i = 1, 2, \dots, |X_0|$, there exists a nonblocking supervisor S such that $L(S/G) = L(S||G) = K$. In this case, S could be any automaton that has the initial state set $S_0 = \{s_0^1, s_0^1, \dots, s_0^m\}$, $m \leq |x_0|$, and for any x_0^i there exists a s_0^j , $(x_0^i, s_0^j) \in \Upsilon$, which satisfies $L_m(S(s_0^j)) = L(S(s_0^j)) = K_i$ where Υ is the coupling relation between the supervisor S and the plant G .*

Now, deploying the parallel composition and the above corollary, we will design the supervisor for reaching the formation, keeping the formation, and collision avoidance modularly.

4.4.2.1 Design of the Supervisor for Reaching and Keeping the Formation

For reaching the formation, it is sufficient to drive the UAV directly towards one of regions $R_{1,j}$, $1 \leq j \leq n_\theta - 1$, located in the first circle. This specification, K_F , is realized in Fig. 4.11. Since, there is another module to handle the collision avoidance, when the event C appears, the formation supervisor should not do any thing to handle the collision avoidance and lets this alarm to be treated by the collision avoidance supervisor. It can be seen that K_F is controllable as it does not disable any uncontrollable event.

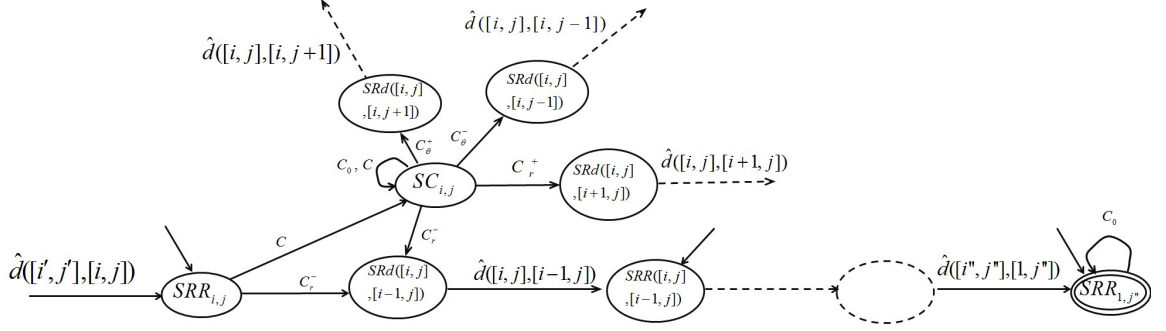


Figure 4.11: Realization of reaching and keeping the formation specification.

Based on Corollary 4, there exists a supervisor that can control the plant to achieve this specification. The supervisor is the realization of the above specification in which all states are marked. Marking all states of the supervisor, lets the closed loop marked states to be solely determined by the plant marked states. The supervisor for reaching the formation and keeping the formation is denoted by S_F . The closed loop system can be obtained using parallel composition: $G_{cl} = S_F/G = S_F||G$. Here, the coupling relation is $\Upsilon = \{(R_{i,j}, SRR_{i,j}) | 1 \leq i, i' \leq n_r - 1, 1 \leq j, j' \leq n_\theta - 1\}$. All of the events are common between the plant and the supervisor. Moreover, it can be seen that $L(S_F) \subseteq L(G)$ which results in

$$L(S_F/G) = L(G||S_F) = L(G) \cap L(S_F) = L(S_F) = K_F \quad (4.13)$$

4.4.2.2 Design of the Supervisor for Collision Avoidance Supervisor

When the follower UAV is going to reach the desired position, it might be happened the case that the follower collide with the leader. More precisely, when the follower UAV is in region $R_{i,j}$ and the leader UAV is located in region $R_{i',j}$, and $i' < i$, then the

collision alarm will be generated. If we look at this problem from the relative frame point of view, the leader UAV has a fixed position in this frame. Therefore, when the follower detects that the leader is located on its path towards the desired position (center of circle), it suffices that the follower turns to change its azimuth angle, θ , and then it can resume reaching the formation task. In other words, the supervisor should lead the follower to region $R_{i,j+1}$ and then resume the mission. This specification, K_C is shown in Fig. 4.12. The supervisor for this task, S_C , is the realization of the collision avoidance specification in which all states are marked. Here, the coupling relation is $\Upsilon = \{(R_{i,j}, SCR_{i,j}) | 1 \leq i, i' \leq n_r - 1, 1 \leq j, j' \leq n_\theta - 1\}$. Also, all the events are common between the plant and the supervisor. Hence, $L(S_C) \subseteq L(G)$ which leads to:

$$L(S_C/G) = L(G||S_C) = L(G) \cap L(S_C) = L(S_C) = K_C \quad (4.14)$$

4.4.2.3 The Closed Loop System

For prefix closed languages K_F and K_C , we can apply modular synthesis [97], by the composition of the plant, the reaching and keeping the formation supervisor, and the collision avoidance supervisor: $G_{cl} = G||S_F||S_C$. Hence, the closed-loop system's language can be achieved as: $L(G||S_F||S_C) = L(G) \cap L(S_F) \cap L(S_C) = L(S_F) \cap L(S_C) = K_F \cap K_C$. The closed-loop automaton, G_{cl} , is shown in Fig. 4.13.

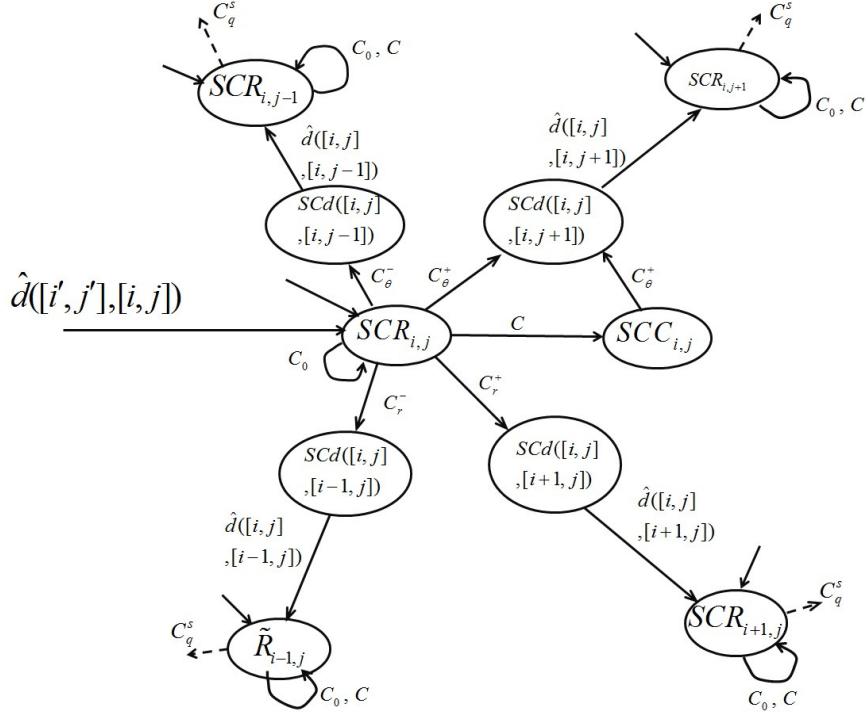


Figure 4.12: Realization of the specification for collision avoidance.

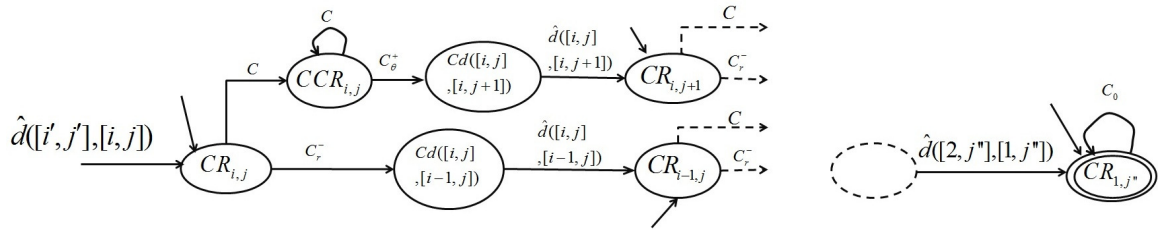


Figure 4.13: The closed loop system.

4.5 Simulation Results

This control architecture has been applied to a leader- follower formation problem. Let $R_m < d$. Assume that the follower dynamics is given by (4.1). The control horizon is 20 meters, and we have selected $n_r = 5$ and $n_\theta = 13$. Using Corollaries 2 and 3, the sets $U_m(Inv(R_{i,j}))$ and $U_m(Ex(E_r^-(R_{i,j})))$ for the control labels C_0 and C_r^- can be achieved. Then, it is sufficient to pick up appropriate values for $u(v_m)$ at each vertex from the sets U_m , while respecting the velocity bounds.

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

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

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

The controller $u(x)$, corresponding to the control label C_0 for the region $R_{i,j}$, is in the form of $u(x) = \sum_{m=0}^3 \lambda_m u(v_m)$ where $\lambda_m(x)$ can be obtained through Theorem 2. We can follow the same procedure to construct the controller for the control label C_r^- .

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 Fig. 4.14 and Fig. 4.15, respectively. These generated velocities should be given to the lower level of control architecture as the references to be tracked.

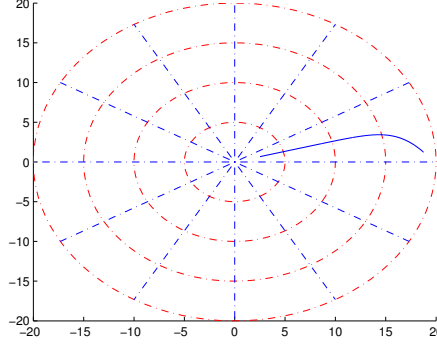


Figure 4.14: Simulation of the system for an initial state inside the region $R_{4,1}$.

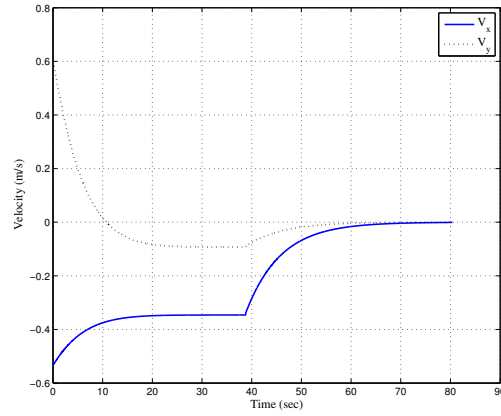


Figure 4.15: Generated velocities V_x and V_y for an initial state inside the region $R_{4,1}$.

The relative distance between the UAV and the desired position is shown in Fig. 4.16. It has finally reached the first circle, i.e., one of the regions $R_{1,j}$, and the global

specification has been achieved;

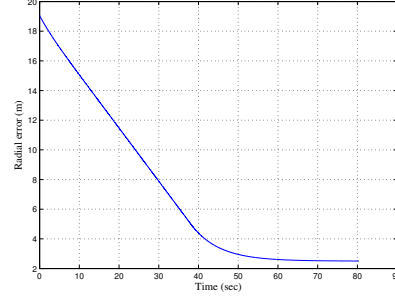


Figure 4.16: Absolute distance from the desired position.

As 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. 4.17. The control mechanism was explained in Section 4.4.2.2. To avoid collision, the follower has first turned to the region $R_{4,2}$ and then, it has resumed the reaching the formation.

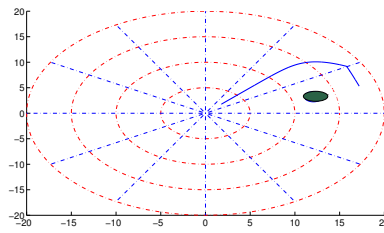


Figure 4.17: Collision avoidance mechanism.

4.6 Extension of the Algorithm to a 3-D Space

In previous parts, the controller was designed for a 2-D formation case under the assumption that the UAV's attitude remains unchanged and irrelevant, but in fact, practical applications demand to address the formation problem in a 3-D space. Therefore, in this section, the previous result is extended to a 3-D space and the spherical partitioning of the state space is provided. Here, the control horizon is a sphere, S_{R_m} , with the radius of R_m that is centered at the desired position, and the model of the UAV is

$$\dot{x} = u \quad x \in \mathbb{R}^3, \quad u \in U \subseteq \mathbb{R}^3, \quad (4.15)$$

where x is the position of the UAV; u is the UAV's velocity reference generated by the formation algorithm, and U is the velocity constraint set, which is a convex set.

The formation problem can be stated as follows:

Problem 2 *Given the dynamics of the follower UAV as (4.15) and its velocity in the form of $V_{follower} = V_{leader} + V_{rel}$, 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.*

4.6.1 Spherical Partitioning

Consider a plant with continuous dynamics of $\dot{x} = g(x) = h(x, u(x))$, defined over the sphere S_{R_m} , with the radius of R_m , where $u(x)$ is the control value computed based on feedbacked position of the system. The sphere S_{R_m} can be partitioned in the spherical coordinate system with $r \geq 0$, $0 \leq \theta < 2\pi$, and $0 \leq \phi \leq \pi$ (Fig. 4.18(a)). The curves $\{r = r_i | 0 \leq r_i \leq R_m, \text{ for } i < j : r_i < r_j, i, j = 1, \dots, n_r, r_1 = 0, r_{n_r} = R_m\}$, $\{\theta = \theta_i | 0 \leq \theta_i \leq 2\pi, \text{ for } i < j : \theta_i < \theta_j, i, j = 1, \dots, n_\theta, \theta_1 = 0, \theta_{n_\theta} = 2\pi\}$, and $\{\phi = \phi_i | 0 \leq \phi_i \leq \pi, \text{ for } i < j : \phi_i < \phi_j, i, j = 1, \dots, n_\phi, \phi_1 = 0, \phi_{n_\phi} = \pi\}$, with $n_r, n_\theta, n_\phi \geq 2$, partition the control horizon S_{R_m} . Equivalently partitioning, we will use $\{r_i = \frac{R_m}{n_r-1}(i-1), i = 1, \dots, n_r\}$, $\{\theta_j = \frac{2\pi}{n_\theta-1}(j-1), j = 1, \dots, n_\theta\}$, and $\{\phi_k = \frac{\pi}{n_\phi-1}(k-1), k = 1, \dots, n_\phi\}$ as the partitioning curves.

In the partitioned space, the region $\bar{R}_{i,j,k} = \{x = (r, \theta, \phi) | r_i \leq r \leq r_{i+1}, \theta_j \leq \theta \leq \theta_{j+1}, \phi_k \leq \phi \leq \phi_{k+1}\}$ is a subset of S_{R_m} surrounded by the above curves. We use the term $R_{i,j,k}$ to denote the interior of the region $\bar{R}_{i,j,k}$. The intersection between the region $\bar{R}_{i,j,k}$ and the partitioning curves is called a *face* and could be 0-dimensional, 1-dimensional, or 2-dimensional, which are named as *vertex*, *edge*, and *facet*, respectively (Fig. 4.18(b)). Each region $R_{i,j,k}$ has eight vertices, v_m , $m = (m_\phi m_\theta m_r)_2$, where m_ϕ , m_θ , and m_r are the binary indices refer to the partitioning curves that have generated the vertex. For example, if $m_r = 1$, it shows that the vertex v_m of the region $R_{i,j,k}$ touches the curve r_{i+1} , and if $m_r = 0$, it touches the curve r_i . The set $V(*)$ stands for the vertices that belong to $*$ ($*$ can be a facet, a region $R_{i,j,k}$, or the sphere S_{R_m}), and $F(v_m)$ is the set of facets that the vertex v_m belongs to them. Furthermore, the

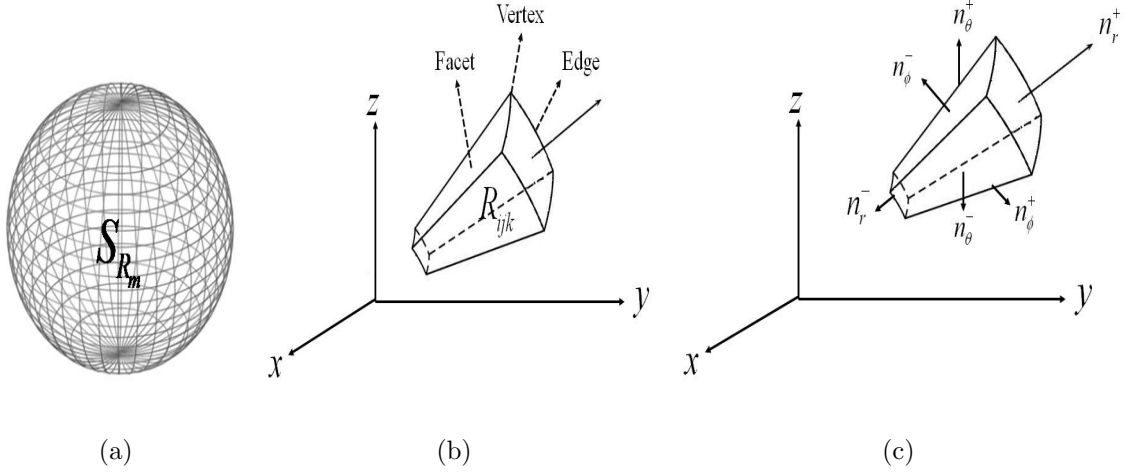


Figure 4.18: (a) The partitioned sphere (b) Vertices, edges, and facets of the element $R_{i,j,k}$, and (c) Outer normal vectors of the element $R_{i,j,k}$.

element $R_{i,j,k}$ has six facets $\{F_r^+, F_r^-, F_\theta^+, F_\theta^-, F_\phi^+, F_\phi^-\}$ and correspondingly, six outer normal vectors $\{n_r^+, n_r^-, n_\theta^+, n_\theta^-, n_\phi^+, n_\phi^-\}$ as shown in Fig. 4.18(c). The exception is when the region $R_{i,j,k}$ touches the origin or the z axis. In this case, some of the vertices are coincident.

In the sphere S_{R_m} , let's define \bar{S} as the sphere surface, and E as the set of all edges and vertices. Also, consider the *detection element* $d([i, j, k], [i', j', k']) = \bar{R}_{i,j,k} \cap \bar{R}_{i',j',k'} - E$, which is defined for two regions $R_{i,j,k}$ and $R_{i',j',k'}$ that are adjacent in a common facet. Indeed, the detection elements are the facets in which the edges and the vertices are excluded. With this procedure, the sphere S_{R_m} has been partitioned into $E \cup R_{i,j,k} \cup d([i, j, k], [i', j', k']) \cup S$, where $1 \leq i, i' \leq n_r - 1$, $1 \leq j, j' \leq n_\theta - 1$, $1 \leq k, k' \leq n_\phi - 1$ and $S = \bar{S} - E$. Correspondingly, consider \tilde{E} , $\tilde{R}_{i,j,k}$, $\tilde{d}([i, j, k], [i', j', k'])$, and \tilde{S} as the labels for these partitioning elements, where $\Im(\tilde{r}) = r$ relates the label \tilde{r} to the set r .

Using the properties of multi-affine functions we will have:

Theorem 5 *For a multi - affine function $g(x) : S_{R_m} \rightarrow \mathbb{R}^3$, the following property holds:*

$$\forall x = (r, \theta, \phi) \in \bar{R}_{i,j,k} : g(x) = \sum_m \lambda_m g(v_m), \quad m = 0, 1, 2, \dots, 7, \quad (4.16)$$

where $v_m \in V(R_{i,j,k})$ are the vertices of the element $R_{i,j,k}$ and λ_m can be obtained uniquely as follows:

$$\lambda_m = \lambda_r^{m_r} (1 - \lambda_r)^{1-m_r} \lambda_\theta^{m_\theta} (1 - \lambda_\theta)^{1-m_\theta} \lambda_\phi^{m_\phi} (1 - \lambda_\phi)^{1-m_\phi}, \quad (4.17)$$

where m_r, m_θ, m_ϕ are the corresponding binary digits of the index m , and

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

Proof: The proof is similar to the proof of Theorem 1. ■

4.6.2 Control over the Spherical Partitioned Space

Extending Theorem 2 and Theorem 3 to a spherical partitioned space, one can design the control signals at the vertices so that a region $R_{i,j,k}$ becomes an invariant region or one of its facet becomes an exit facet:

Theorem 6 *Sufficient condition for $R_{i,j,k}$ to be an invariant region:* For a multi - affine vector field $\dot{x} = g(x)$, $g : S_{R_m} \rightarrow \mathbb{R}^3$, $R_{i,j,k}$ is an invariant region if for each facet F_q^s and its corresponding outer normal n_q^s , $q \in \{r, \theta, \phi\}$ and $s \in \{+, -\}$ the following inequality holds:

$$n_q^s(y)^T \cdot g(v_m) < 0, \quad \forall v_m \in V(F_q^s), \quad \forall y \in F_q^s \quad (4.18)$$

Theorem 7 Sufficient condition for an Exit facet: For a multi - affine vector field $\dot{x} = g(x)$, $g : S_{R_m} \rightarrow \mathbb{R}^3$, the facet F_q^s with the outer normal n_q^s , $q \in \{r, \theta, \phi\}$ and $s \in \{+, -\}$, is an exit facet if :

1. $n_{q'}^{s'}(y)^T \cdot g(v_m) < 0 \quad \forall v_m \in V(F_{q'}^{s'}), \forall y \in F_{q'}^{s'}, q' \neq q, \text{ or } s' \neq s$
2. $n_q^s(y)^T \cdot g(v_m) > 0 \quad \forall v_m \in V(R_{i,j,k}), \text{ for all } y \in F_q^s$

Therefore, to make the region $R_{i,j,k}$ an invariant region, the value of the control signal at the vertices, $u(v_m)$, should be chosen such that $g(v_m) = h(v_m, u(v_m))$ falls in the set $U_m(Inv(R_{i,j,k})) = Inv_m(R_{i,j,k}) \cap U$, for $m = 0, \dots, 7$, where $Inv_m(R_{i,j,k})$ is the eligible set for the vertex v_m so that $g(v_m)$ satisfies the conditions of Theorem 6, and U is the velocity bound, which comes from the practical limitations. If $U_m(Inv(R_{i,j,k})) \neq \emptyset$, for all $m = 0, \dots, 7$, then, making the region $R_{i,j,k}$ an invariant region is feasible. Based on Theorem 5, having the value of the control function u at the vertices of the region, it is possible to construct the multi-affine controller $u(x)$ for all $x \in \bar{R}_{i,j,k}$. We will use the notation C_0 to label this controller.

To make the facet F_q^s an exit facet, similar to the invariant controller, it is sufficient to choose the values of $u(v_m)$ such that $g(v_m) = h(v_m, u(v_m))$ falls in the set $U_m(Ex(F_q^s(R_{i,j,k}))) = Ex_m(F_q^s(R_{i,j,k})) \cap U$, where $Ex_m(F_q^s(R_{i,j,k}))$ is the eligible set for the vertex v_m that satisfies the exit facet condition for $F_q^s(R_{i,j,k})$ as explained in Theorem 7, and U is the velocity constraint. Therefore, for the region $R_{i,j,k}$, if all of $U_m(Ex(F_q^s(R_{i,j,k}))) \neq \emptyset$, $m = 0, \dots, 7$, then corresponding to each of its facets, F_r^+ , F_r^- , F_θ^+ , F_θ^- , F_ϕ^+ , F_ϕ^- , there are controllers that can make them exit facet. We label these controllers as C_r^+ , C_r^- , C_θ^+ , C_θ^- , C_ϕ^+ , C_ϕ^- , respectively.

Using this set of controllers over the partitioned space, the plant can be abstracted into a finite state machine and can be presented by an automaton $G = (X, \Sigma, \alpha, X_0, X_m)$, where $X = \{\tilde{R}_{i,j,k} | 1 \leq i \leq n_r - 1, 1 \leq j \leq n_\theta - 1, 1 \leq k \leq n_\phi - 1\} \cup \{\tilde{d}([i, j, k], [i', j', k']) | 1 \leq i, i' \leq n_r - 1, 1 \leq j, j' \leq n_\theta - 1, 1 \leq k, k' \leq n_\phi - 1\}$, is the set of states where $\tilde{R}_{i,j,k}$ and $\tilde{d}([i, j, k], [i', j', k'])$ are the labels for the regions $R_{i,j,k}$ and $d([i, j, k], [i', j', k'])$, respectively. $X_0 = \{\tilde{R}_{i,j,k} | 1 \leq i, i' \leq n_r - 1, 1 \leq j, j' \leq n_\theta - 1, 1 \leq k, k' \leq n_\phi - 1\}$ is the set of initial states; $X_m = \{\tilde{R}_{1,j,k} | 1 \leq j \leq n_\theta - 1, 1 \leq k \leq n_\phi - 1\}$ is the set of final (marked) states. Σ is the finite set of events. Here, the event set Σ consists of the actuation events $U_a = \{C_q^s | q \in \{r, \theta, \phi\}, s \in \{+, -\}\} \cup \{C_0\}$, the crossing detection events $U_c = \{\hat{d}([i, j, k], [i', j', k']) | 1 \leq i, i' \leq n_r - 1, 1 \leq j, j' \leq n_\theta - 1, 1 \leq k, k' \leq n_\phi - 1\}$, and the external event set U_e . The set $U_e = \{C\}$ contains the event C which alarms about the collision. The event set Σ consists of the controllable event set $\Sigma_c = U_a$ and uncontrollable event set $\Sigma_{uc} = U_d = U_c \cup U_e$. In automaton G , $\alpha : X \times \Sigma \rightarrow X$ is the transition function, which is a partial function and determines the possible transitions in the system caused by an event which can be defined as follows

$$\alpha(\tilde{R}_{i,j,k}, \sigma) = \begin{cases} \tilde{R}_{i,j,k} & \sigma = C_0 \\ \tilde{R}_{i,j,k} & \sigma \in U_e \text{ for } i \neq 1 \\ \tilde{d}([i, j, k], [i+1, j, k]) & \sigma = C_r^+ \text{ for } i \neq n_r - 1 \\ \tilde{d}([i, j, k], [i-1, j, k]) & \sigma = C_r^- \text{ for } i \neq 1 \\ \tilde{d}([i, j, k], [i, j+1, k]) & \sigma = C_\theta^+ \text{ for } j \neq n_\theta - 1 \\ \tilde{d}([i, j, k], [i, j-1, k]) & \sigma = C_\theta^- \text{ for } j \neq 1 \\ \tilde{d}([i, j, k], [i, n_\theta - 1, k]) & \sigma = C_\theta^- \text{ for } j = 1 \\ \tilde{d}([i, j, k], [i, j, k+1]) & \sigma = C_\phi^+ \text{ for } k \neq n_\phi - 1 \\ \tilde{d}([i, j, k], [i, j, k-1]) & \sigma = C_\phi^- \text{ for } k \neq 1 \end{cases}$$

$$\alpha(\tilde{d}([i, j, k], [i', j', k']), \sigma) = \tilde{R}_{i',j',k'} \quad \sigma = \hat{d}([i, j, k], [i', j', k'])$$

Some parts of the graph representation of the system automaton are shown in Fig. 4.19.

Next, we ill design the discrete supervisor for this discrete model.

4.6.3 Designing the Supervisor for a Formation Mission over a the Spherically Partitioned Space.

For reaching the formation, it is sufficient to activate the event C_r^- to drive the follower UAV towards one of the sectors in the first circle. After reaching one of the regions $R_{1,j,k}$, $1 \leq j \leq n_\theta - 1$, $1 \leq k \leq n_\phi - 1$, the UAV should remain there. This specification, K_F , is realized in Fig. 4.20. When the UAV is not in the first sphere,

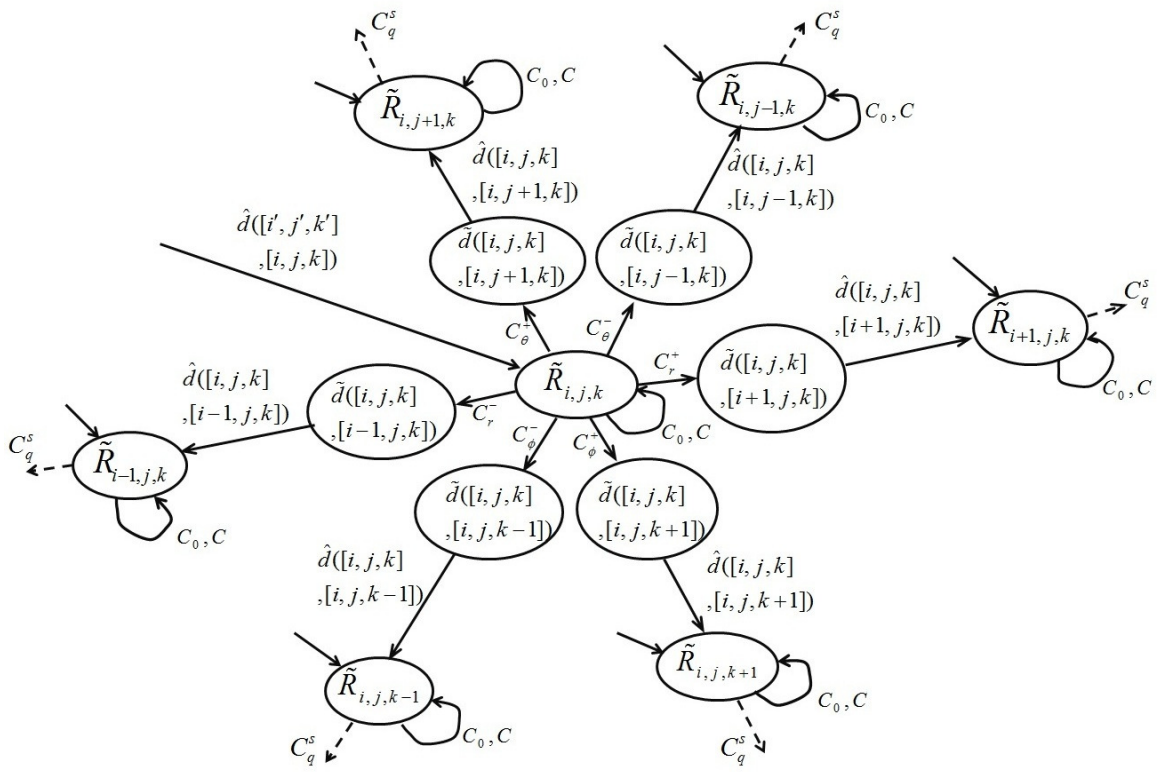


Figure 4.19: DES model of a spherically partitioned plant.

the command C_r^- will be generated to push the UAV towards the origin. Entering a new state, the event $d([i, j, k], [i', j', k'])$ will appear to show the current state of the system. This will continue until the event $d([i, j, k], [1, j', k'])$ being generated, which shows that the formation is reached. In this case, the event C_0 will be activated, which keeps the system trajectory in the region. It can be seen that K_F is controllable as it does not disable any uncontrollable event.

Based on Lemma 4, there exists a supervisor that can control the plant to achieve this specification. The supervisor is the realization of the above specification in which all states are marked. Marking all states of the supervisor allows the closed-loop marked states to be solely determined by the plant marked states. The supervisor for reaching the formation and keeping the formation is denoted by S_F . The closed-loop system can be obtained using the parallel composition: $G_{cl} = S_F/G = S_F||G$. Here, the coupling relation is $\Upsilon = \{(R_{i,j,k}, SRR_{i,j,k}) | 1 \leq i, i' \leq n_r - 1, 1 \leq j, j' \leq n_\theta - 1, 1 \leq k, k' \leq n_\phi - 1\}$. All of the events are common between the plant and the supervisor. Moreover, it can be seen that $L(S_F) \subseteq L(G)$ which leads to: $L(S_F/G) = L(G||S_F) = L(G) \cap L(S_F) = L(S_F) = K_F$.

4.6.3.1 Collision Avoidance Supervisor

When the follower UAV is going to reach the desired position, in some situations, the follower may collide with the leader. If the leader is located in the way of the follower towards the desired position and the follower enters the alarm zone, a collision alarm will be generated. More precisely, assume that the follower is in region $R_{i,j,k}$, and

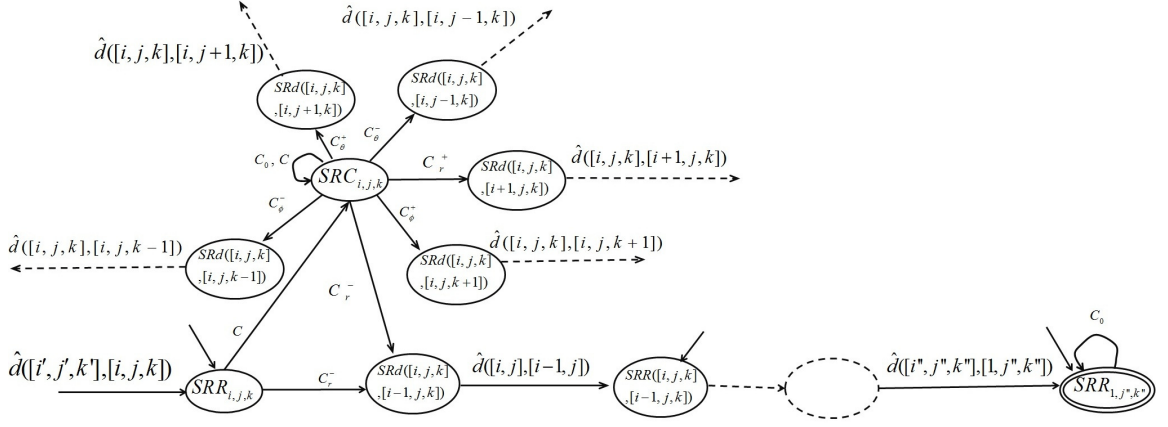


Figure 4.20: The realization of reaching and keeping the formation specification.

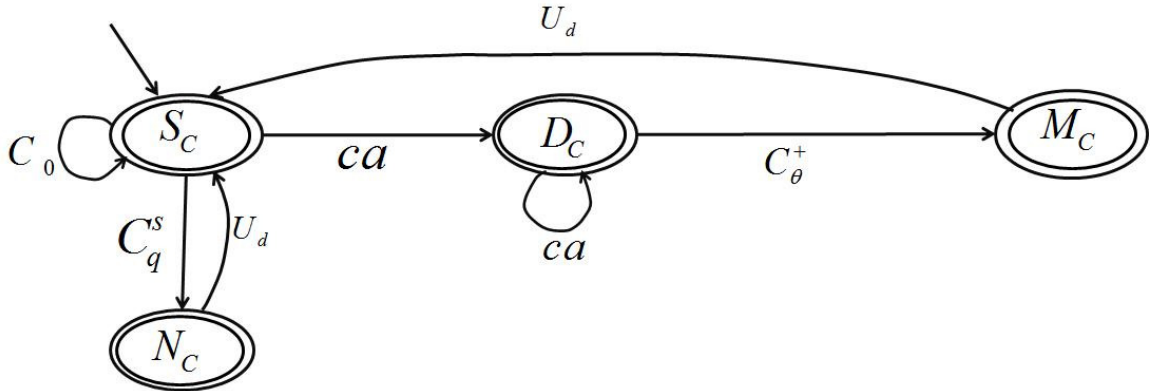


Figure 4.21: The realization of collision avoidance specification, K_C .

the leader is in region $R_{i',j,k}$ and $i' < i$, then the collision alarm will be generated. If we look at this problem from the relative frame point of view, the leader UAV has a fixed position in this frame, and therefore, for the collision avoidance, it suffices that the follower turns to change its azimuth angle, θ , and then it can resume reaching the formation task. In other words, by activating the command C_θ^+ , the supervisor should lead the follower to region $R_{i,j+1,k}$ and then resume the mission. For this purpose, the collision avoidance supervisor only changes the generatable language after happening the event C and lets the rest be treated by the formation supervisor. The collision avoidance specification, K_C , is shown in Fig. 4.21. Here, the coupling relation is $\Upsilon = \{(R_{i,j,k}, SC R_{i,j,k}) | 1 \leq i, i' \leq n_r - 1, 1 \leq j, j' \leq n_\theta - 1, 1 \leq k, k' \leq n_\phi - 1\}$. Again, all of the events are common between the plant and the supervisor, S_C . Hence, $L(S_C) \subseteq L(G)$ which leads to: $L(S_C/G) = L(G||S_C) = L(G) \cap L(S_C) = L(S_C) = K_C$, where K_C is the collision avoidance specification.

4.6.3.2 The closed-loop system

Hence, the closed-loop system's language can be achieved as: $L(G||S_F||S_C) = L(G) \cap L(S_F) \cap L(S_C) = L(S_F) \cap L(S_C) = K_F \cap K_C$. The closed-loop automaton, G_{cl} , is shown in Fig. 4.22.

4.6.4 Simulation Results

To verify the proposed algorithm, we have used a hardware-in-the-loop simulation platform [85] developed for NUS UAV helicopters [98]. In this platform, the nonlinear

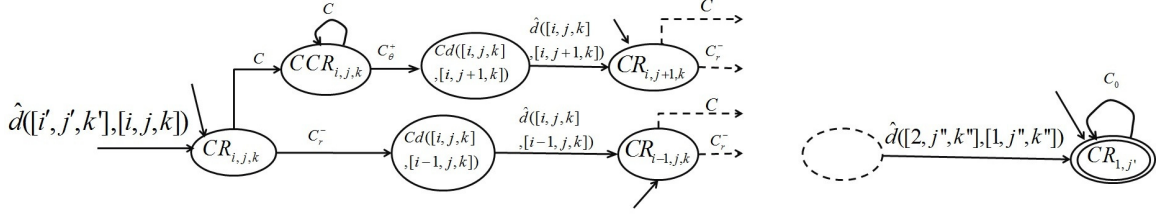


Figure 4.22: The closed loop system.

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 simulate the algorithm for the following cases:

4.6.4.1 Simulation of Reaching the Formation and Collision Avoidance

First, to monitor the reaching the formation and the collision avoidance behaviour of the UAVs, assume that the leader has a fixed position and the follower should reach the desired position with respect to the leader. The controller, $u(x)$, drives the UAV inside the spherical partitioned space. This control signal is generated using the control mechanism described in Section 4.6.2. The control horizon is a sphere of diameter 50m. The partitioning parameters are selected as $n_r = 15$, $n_{\theta} = 20$, and $n_{\phi} = 10$. To construct the controllers C_0 , C_r^{+} , C_r^{-} , C_{θ}^{+} , C_{θ}^{-} , C_{ϕ}^{+} , and C_{ϕ}^{-} , we can apply Theorem 6 and 7, respectively. Now, assume that the relative distance between the follower and the desired position is $(dx, dy, dz) = (-17, -18, -8)$. Hence, the initial state of the system is $R_{8,13,5}$. Also, assume that the leader is in $R_{7,13,5}$. Since

the leader is located on the path of follower towards the desired position, a collision avoidance alarm will be generated to activate the collision avoidance mechanism and to push the follower UAV to avoid collision. The collision avoidance behavior of the system is shown in Fig. 4.23. The projection of the relative distance between the follower and its desired position onto the $x - y$ plane is shown in Fig. 4.24. The follower first has moved towards the region $R_{8,14,5}$ to avoid the collision, and then, it has resumed reaching the formation to complete the mission.

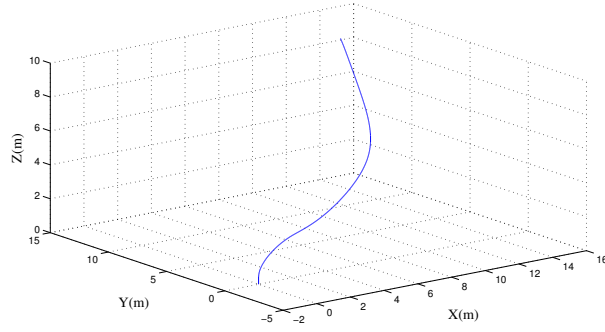


Figure 4.23: The position of the UAV for the collision avoidance mechanism.

4.6.4.2 Simulation of Keeping the Formation

To monitor reaching and keeping the formation, let the leader track a circle with the diameter of 20m and choose the partitioning parameters like the previous mentioned scenario. The follower is initially located at $(dx, dy, dz) = (-20, -20, -20)$ with respect to the leader. It is expected that after a while, the follower reaches the relative distance of $(dx, dy, dz) = (5, 5, 5)$ with respect to the leader. The behavior of the follower UAV is shown in Fig. 4.25. As it can be seen the follower has finally

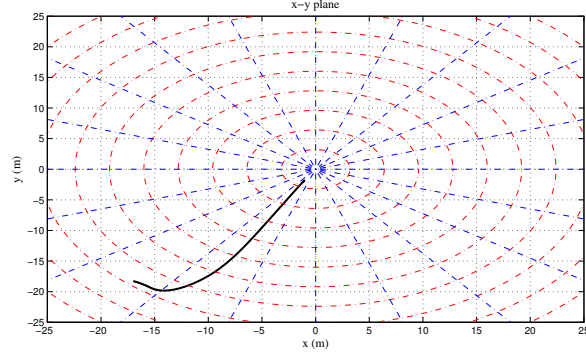


Figure 4.24: The relative distance between the follower and the desired position for the collision avoidance mechanism projected onto x-y plane.

reached the desired formation and has successfully kept it.

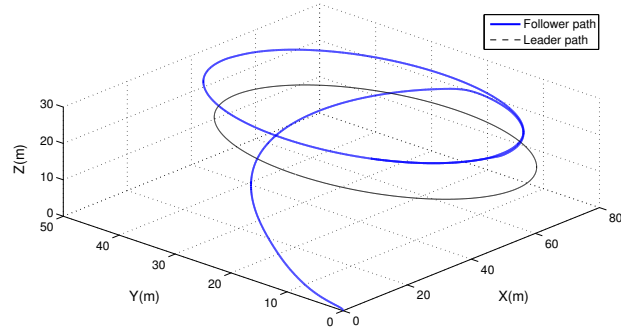


Figure 4.25: The position of the UAVs in a circle formation mission.

4.7 Conclusion

In this chapter, 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 motion space. Several multi-affine feedback controllers were designed to keep the system inside a partitioning element or derive 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, a 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. Then, the method was extended to a 3-D space using spherical abstraction of the motion space.

Next chapter will discuss some implementation issues on the proposed algorithm and will demonstrate actual flight test results for this algorithm.

Chapter 5

Implementation Issues and Flight Test

Results for the Proposed Hybrid

Formation Algorithm

5.1 Introduction

In the previous chapter, we proposed a hybrid supervisory control algorithm for UAV helicopters which integrated the discrete supervisory logic of the system and its continuous low level control and addressed all parts of a formation mission within a unified framework. The approach was based on polar abstraction of the motion space and utilizing the properties of multi-affine functions over the partitioned space. This abstraction technique, 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) [67]. Due to the proven bisimulation relation, the abstracted system can behave as the same as the original system so that the discrete supervisor, designed for the discrete finite model, can be applied to the original system. However, there are certain issues that need to be considered before applying the designed supervisor to the real flying systems.

This chapter focuses on the implementation issues of the proposed algorithm. More specifically, firstly, an interface layer is introduced to connect the discrete supervisor layer to the continuous plant. This interface layer on the one hand converts the continuous signals of the plant into some symbols understandable by the discrete supervisor, and on the other hand, translates discrete commands of the supervisor into continuous signals applicable to the continuous plant. Secondly, the time scheduling of the events being generated by the system has been investigated and has been correspondingly considered in the implementation of the supervisor. Thirdly, a control scheme is proposed to smoothly transit through the partitioning elements so that there is no jump in the generated control signal when the system transit from one region to the adjacent region. Finally, the proposed algorithm has been verified through actual flight tests.

The rest of this chapter is organized as follows. In Section 5.2, a hierarchical hybrid control structure is proposed which has the discrete supervision layer on the top, and an interface layer which connects the supervision layer to the continuous plant. Section 5.3 describes implementation issue for the algorithm. The flight test results are demonstrated in Section 5.4. The chapter is concluded in Section 5.5.

5.2 Hierarchical Control Structure for the Formation Control

In chapter 4, we designed a discrete supervisor which pushes the system trajectories to pass through the desired regions. The designed discrete supervisor cannot directly be connected to the continuous plant. Hence, it is required to construct an interface layer which can translate continuous signals of the plant to a sequence of discrete symbols understandable for the supervisor. Also, the interface layer is responsible to convert discrete commands received from the supervisor, to continuous control inputs to be given to the plant. These two jobs are respectively realized by the blocks Detector and Actuator embedded in the interface layer as it is shown in Fig. 5.1. In this control hierarchy, the supervisor layer was discussed in Section 4.4.2. Here, the elements of the interface layer are explained in the following section.

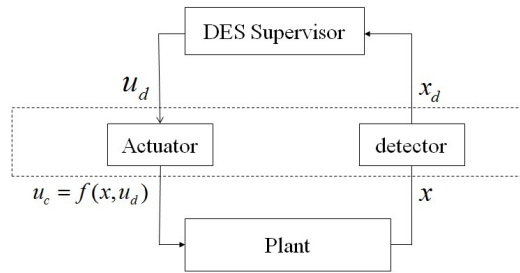


Figure 5.1: Linking the discrete supervisor to the plant via an interface layer.

5.2.1 The Interface Layer

5.2.1.1 The Detector Block

When the system's trajectory crosses the boundaries of the region, a detection event will be generated which informs the supervisor that the system has entered a new region.

More specifically, a detection event $\hat{d}([i', j'], [i, j])$ will happen at $t(\hat{d}([i', j'], [i, j]))$ when the system's trajectory $x(t)$ satisfies the following conditions:

- $\exists \tau > 0$ such that $x(t) \notin R_{i,j}$ for $t \in (t(\hat{d}([i', j'], [i, j])) - \tau, t(\hat{d}([i', j'], [i, j])))$
- $\exists \tau_d > 0$ such that $x(t) \in R_{i,j}$ for $t \in [t(\hat{d}([i', j'], [i, j])), t(\hat{d}([i', j'], [i, j])) + \tau_d)$

Also, if the leader position locates in the way of the follower towards the desired position, the event C will be generated to inform the supervisor about the risk of collision.

5.2.1.2 The Actuator Block

Having the information about the newly entered region, the supervisor can issue a discrete command to push the system trajectory to move towards the desired region. However, the discrete symbols generated by the supervisor need to be translated to a continuous form. For such a purpose, we utilize the properties of multi-affine function by which we can design continuous controllers that drive the system's trajectory to either exit from one of its edges or to stay in the current region for ever.

Solving the inequalities given in Corollaries 2 and 3, for the dynamics given in

(4.1) the following control values at the vertices of the region $R_{i,j}$ can make it an invariant region or can make one of its edges an exit edge. For the invariant controller, the control label is C_0 and the control values at the vertices are:

$$\begin{cases} u(v_0) = 1\angle(\theta_j + 0.5 \mid \theta_j - \theta_{j+1} + \frac{\pi}{2} \mid) \\ u(v_1) = 1\angle(\theta_j + \pi - 0.5 \mid \theta_j - \theta_{j+1} + \frac{\pi}{2} \mid) \\ u(v_2) = 1\angle(\theta_{j+1} - 0.5 \mid \theta_j - \theta_{j+1} + \frac{\pi}{2} \mid) \\ u(v_3) = 1\angle(\theta_{j+1} + \pi + 0.5 \mid \theta_j - \theta_{j+1} + \frac{\pi}{2} \mid) \end{cases}$$

To have the edge E_r^+ as the exit edge, the control label is C_r^+ and the control values at the vertices are:

$$\begin{cases} u(v_0) = u(v_1) = 1\angle(\theta_j + 0.5 \mid \theta_j - \theta_{j+1} + \frac{\pi}{2} \mid) \\ u(v_2) = u(v_3) = 1\angle(\theta_{j+1} - 0.5 \mid \theta_j - \theta_{j+1} + \frac{\pi}{2} \mid) \end{cases}$$

To have the edge E_r^- as the exit edge, the control label is C_r^- and the control values at the vertices are:

$$\begin{cases} u(v_0) = u(v_1) = 1\angle(\theta_j + \pi - 0.5 \mid \theta_j - \theta_{j+1} + \frac{\pi}{2} \mid) \\ u(v_2) = u(v_3) = 1\angle(\theta_{j+1} + \pi + 0.5 \mid \theta_j - \theta_{j+1} + \frac{\pi}{2} \mid) \end{cases}$$

To have the edge E_θ^+ as the exit edge, the control label is C_θ^+ and the control values at the vertices are:

$$\begin{cases} u(v_0) = 1\angle(\theta_j + 0.5 \mid \theta_j - \theta_{j+1} + \frac{\pi}{2} \mid) \\ u(v_1) = 1\angle(\theta_j + \pi - 0.5 \mid \theta_j - \theta_{j+1} + \frac{\pi}{2} \mid) \\ u(v_2) = 1\angle(\theta_{j+1} + 0.5 \mid \theta_j - \theta_{j+1} + \frac{\pi}{2} \mid) \\ u(v_3) = 1\angle(\theta_{j+1} + \pi - 0.5 \mid \theta_j - \theta_{j+1} + \frac{\pi}{2} \mid) \end{cases}$$

To have the edge E_θ^- as the exit edge, the control label is C_θ^- and the control values at the vertices are:

$$\begin{cases} u(v_0) = 1\angle(\theta_j - 0.5 \mid \theta_j - \theta_{j+1} + \frac{\pi}{2} \mid) \\ u(v_1) = 1\angle(\theta_j + \pi + 0.5 \mid \theta_j - \theta_{j+1} + \frac{\pi}{2} \mid) \\ u(v_2) = 1\angle(\theta_{j+1} - 0.5 \mid \theta_j - \theta_{j+1} + \frac{\pi}{2} \mid) \\ u(v_3) = 1\angle(\theta_{j+1} + \pi + 0.5 \mid \theta_j - \theta_{j+1} + \frac{\pi}{2} \mid) \end{cases}$$

Now, the responsibility of the actuator is to relate the discrete symbols $u_d \in \{C_0, C_r^-, C_r^+, C_\theta^+, C_\theta^-\}$ to the continuous control signal $u_c(x)$. Using the properties of multi-affine functions described in Theorem 1, the control signal can be constructed as $u_c(x) = f(x, u_d) = \sum_{m=0}^3 \lambda_m(x)u(v_m)$, where $u(v_m), m = 0, \dots, 3$, are the control values at the vertices corresponding to the control label u_d .

5.2.2 Applying the Discrete Supervisor to the Continuous Plant via the Interface Layer

In Section 4.4.2, a discrete supervisor was designed for the discrete model of the plant. Here, we will show that this discrete supervisor can be applied to the original system via the interface layer, while the same result can be achieved and the desired specification still is achievable.

Theorem 8 *With the aid of interface layer, the discrete supervisor $S = S_C || S_F$ can be applied to the original partitioned system, T_Q , so that the closed-loop system satisfies the required specification, $K_F \cap K_C$.*

Proof: As constructed in Section 4.3.4, the plant and the interface layer elements, together with the actuator and the detector, form the transition system T_Q . Theorem 4 shows that this transition system can be bisimilarly abstracted to the finite state machine T_ξ for which we designed the discrete supervisor. Due to the bisimilarity of T_Q and T_ξ , the designed supervisor for T_ξ can work for T_Q so that their closed-loop system behaviors are the same. ■

5.3 Implementation Issues

5.3.1 Time Sequencing of the Events

An important issue in the designed control structure is the time scheduling of the events. In the proposed framework, we assume that the discrete control signals, C_0 , C_r^+, C_r^- , C_θ^+ , or C_θ^- , can be applied after entering a new region, unless a collision alarm be generated which requires an immediate reaction. But, the question is that, transiting to a new region, when should exactly the new control signals be applied to the system?

Indeed, from practical reasons, the detector cannot recognize entering a region until the system trajectory crosses the region's boundary. This is why in the definition of the exit edge we have considered a time delay $\tau_d > 0$. Only after this time delay, the controller can be ensured that the system trajectory has transited to a new region and hence, a new actuation event C_0 , C_r^+, C_r^- , C_θ^+ , or C_θ^- , can be generated based on the desired behavior. This also guarantees that the resulting model is not Zeno [99],

meaning that the number of discrete transitions in a finite time is finite.

More precisely, when the last visited region is $R_{i,j}$ and the supervisor detects an event $\hat{d}([i,j], [i',j'])$, it means that the system trajectory has entered the new region $R_{i',j'}$. Then, a control command C_q^s will be generated which pushes the system trajectory to enter another region $R_{i'',j''}$. Again, when the system's trajectory crosses the boundaries of the region $R_{i'',j''}$, this will cause the event $\hat{d}([i',j'], [i'',j''])$ to appear. Hence, for the successive events $\hat{d}([i,j], [i',j'])$, C_q^s , $\hat{d}([i',j'], [i'',j''])$, we will have:

$$t(\hat{d}([i,j], [i',j'])) < t(C_q^s) < t(\hat{d}([i',j'], [i'',j''])) \quad (5.1)$$

To consider the time delay $\tau_d > 0$, the sequence of the events should respect the following condition:

$$t(C_q^s) \geq t(\hat{d}([i,j], [i',j'])) + \tau_d \quad (5.2)$$

5.3.2 Smooth Control over the Partitioned Space

When the system trajectory enters a new region, a new discrete command will be generated. This may cause the discontinuity in the generated control signal to be applied to the lower levels of the control structure.

Fig. 5.2, shows a case that the control command C_r^- has pushed the system's trajectory to transit from the region R_1 to the region R_2 . After reaching the region R_2 , the control command has changed from C_r^- to C_θ^+ . Since the generated continuous control signal is a multi-affine function, based on Corollary 1, the control value at

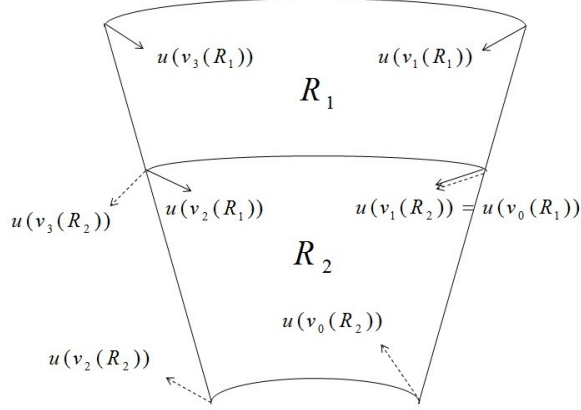


Figure 5.2: The control value at the vertices while transiting through the regions.

any point on the edges is determined by the control values at its vertices. In this example, $u(v_0(R_1)) = u(v_1(R_2))$ but $u(v_2(R_1)) \neq u(v_3(R_2))$. Since, the control values at the vertices of the common edge between R_1 and R_2 change, there is a jump on the generated continuous control signal. Next theorem shows how we can resolve this problem.

Theorem 9 *Let the command C_q^s steers the system's trajectory from the region $R_{i,j}$ to the region $R_{i',j'}$ and then, the supervisor issues the new command $C_{q'}^{s'}$. For this transition, the multi-affine controller $u(x) = \sum_{v_m \in V_c} \lambda_m \hbar(u(v_m)_{new}, u(v_m)_{old}) + \sum_{v_m \in V_n} \lambda_m(u(v_m))$ provides a continuous control signal, while preserving the bisimulation relation given in Theorem 4.*

where λ_m , $m = 0, 1, 2, 3$, are given in Theorem 1, V_n is the set of vertices whose control values do not change due to the transition, and V_c is the set of vertices whose control values change after the system's trajectory enters the region $R_{i',j'}$. For these vertices, $u(v_m)_{old}$ and $u(v_m)_{new}$ are the control values at the vertex v_m before and

after transiting to $R_{i',j'}$, respectively. The function \hbar provides a smooth rotation from

$u(v_m)_{old}$ to $u(v_m)_{new}$ and it can be presented as $\hbar(u(v_m)_{new}, u(v_m)_{old}) =$

$$\begin{cases} r_m \angle (\frac{t}{\Delta t} \theta_{m_{new}} + (1 - \frac{t}{\Delta t}) \theta_{m_{old}}) & t < \Delta t \\ r_m \angle \theta_{m_{new}} & t \geq \Delta t \end{cases}$$

where $u(v_m)_{new} = r_m \angle \theta_{m_{new}}$, $u(v_m)_{old} = r_m \angle \theta_{m_{old}}$, and Δt is the transition time.

Proof: Let $C_q^s = C_r^-$ and $C_{q'}^{s'} = C_\theta^+$. As shown in Fig. 5.2, for this sequence of control commands, after transiting from $R_{i,j}$ to $R_{i',j'}$, the control value at the vertex v_3 changes from $u(v_3)_{old}$ to $u(v_3)_{new}$, and for the other vertices v_m , $m = 0, 1, 2$, there is no jump for the value of the vector field.

From the definition of the transition rule, \hbar , since for the whole transition time, the control values at the vertices satisfy the conditions of Corollary 2, the system's trajectory cannot leave the region through the edges $E_{0,2}$, $E_{0,1}$, $E_{1,3}$. However, at the beginning of the transition mode, the control values at the vertex v_3 does not satisfy the conditions of Theorem 3, and hence, it cannot be concluded that the system's trajectory leaves the region through $E_{2,3}$. But, at some time, $u(v_3)$ will reach $u(v_3)_{new}$, and the configuration of the vector field at the vertices will eventually satisfy the conditions of Theorem 3 so that it can be guaranteed that the system's trajectory for sure leaves the region $R_{i',j'}$ through the edge $E_{2,3}$, while there is no jump at the value of the control signal due to the smooth transition of the control values at the vertices. The same reasoning can be done for the other sequences of the control commands. ■

5.4 Implementation Results

To verify the algorithm, we have conducted several flight tests. In the first scenario, to monitor reaching the formation behavior of the UAVs, the follower should reach the desired position with respect to a fixed leader. In this test the control horizon $R_m = 50m$, $n_r = 10$, and $n_\theta = 20$. The follower is initially located at a point which has a relative distance of $(dx, dy) = (-40, -5)$ with respect to the desired position as shown in Fig. 5.3. The follower state variables and control signals are shown in Fig. 5.4 and Fig. 5.5, respectively. The follower UAV position in the relative frame is shown in Fig. 5.6. As it can be seen the follower UAV has started from the region $R_{9,11}$ and finally has reached the region $R_{1,11}$ which is located in the first circle and hence, the formation has been achieved.

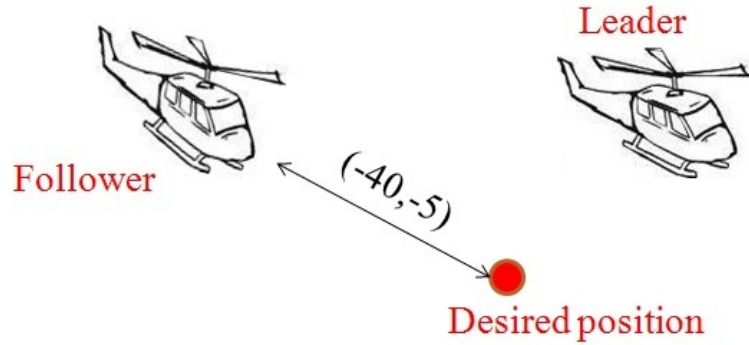


Figure 5.3: The schematic of the scenario with a real follower and a virtual fixed leader.

In the second scenario, to monitor how the follower is able to maintain the achieved formation, the leader tracks a line path, and the follower should reach and

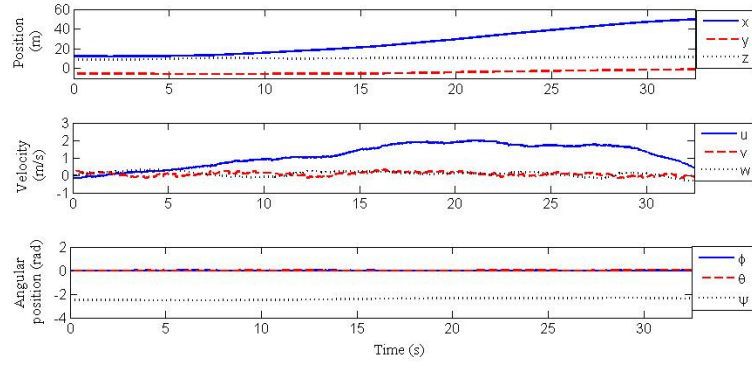


Figure 5.4: The state variables of the follower.

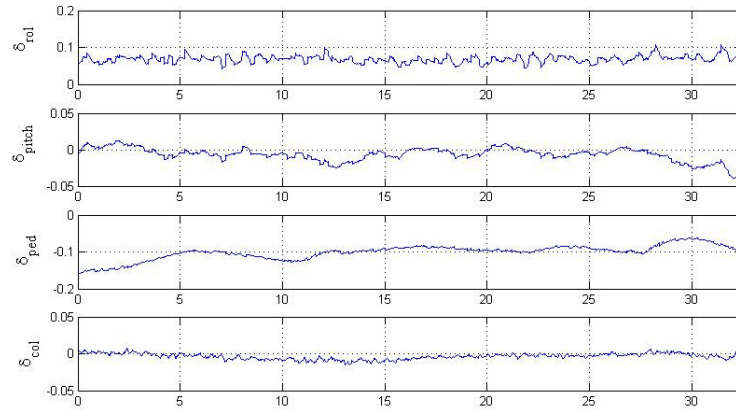


Figure 5.5: Control signals of the follower UAV.

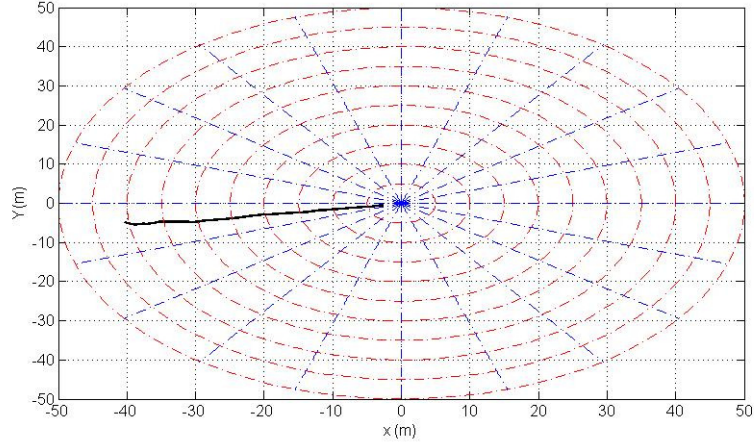


Figure 5.6: The leader position in the relative frame.

keep the formation. In this test, the control horizon R_m is 50 meter, $n_r = 10$, and $n_\theta = 20$. The follower is initially located at a point which has a relative distance of $(dx, dy) = (-17.8, 11.4)$ with respect to the desired position and the distance between the desired position and the leader is $(dx, dy) = (-5, -15)$ as shown in Fig. 5.7.

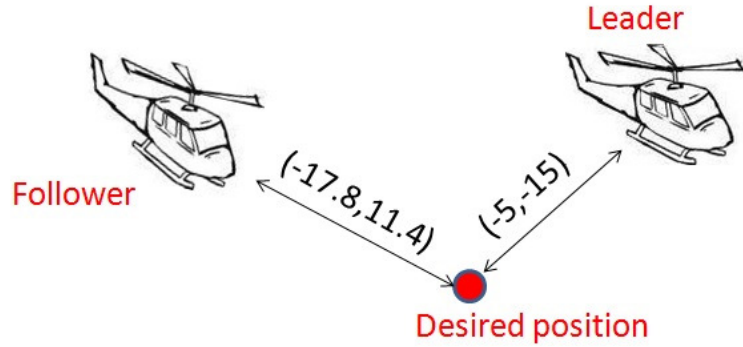


Figure 5.7: The schematic of the scenario for a leader-follower case for tracking a line.

The position of the UAVs in x-y plane is shown in Fig. 5.8. The follower state variables and control signals are shown in Fig. 5.9 and Fig. 5.10, respectively. The

state variables of the leader are shown in Fig. 5.11. The relative distance of the follower UAV from the desired position is shown in Fig. 5.12. As it can be seen the follower UAV has finally reached the first circle after 17 sec and then, it has been able to maintain the formation.

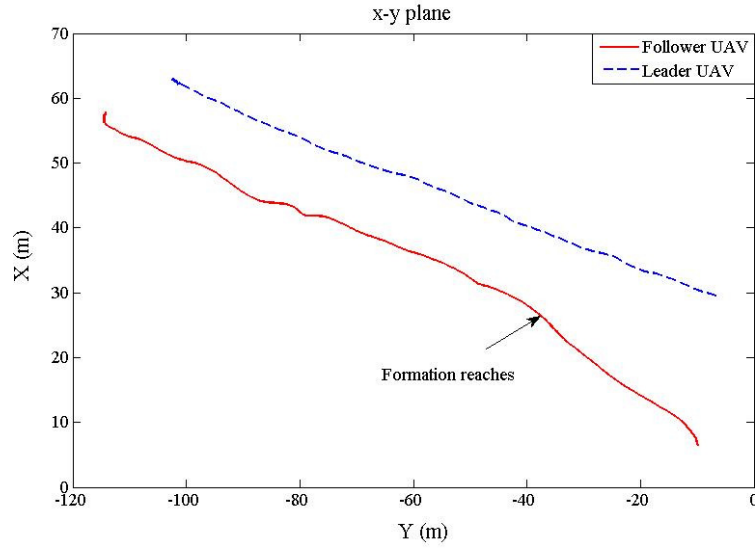


Figure 5.8: The position of the UAVs in the x-y plane.

In the third flight test, the leader path is a circle which is a more complex path. Here, the control horizon R_m is 50 meter, $n_r = 10$, and $n_\theta = 20$. The follower is initially located at a point which has a relative distance of $(dx, dy) = (-30.5, 13.2)$ with respect to the desired position and the distance between the desired position and the leader is $(dx, dy) = (-5, -15)$ as shown in Fig. 5.13. In this test the leader tracks a circle path with a diameter of 40 m. After a while, the follower reaches the formation and can keep it for the rest of the mission. The position of the UAVs in x-y plane is shown in Fig. 5.14. The follower state variables and control signals are shown in

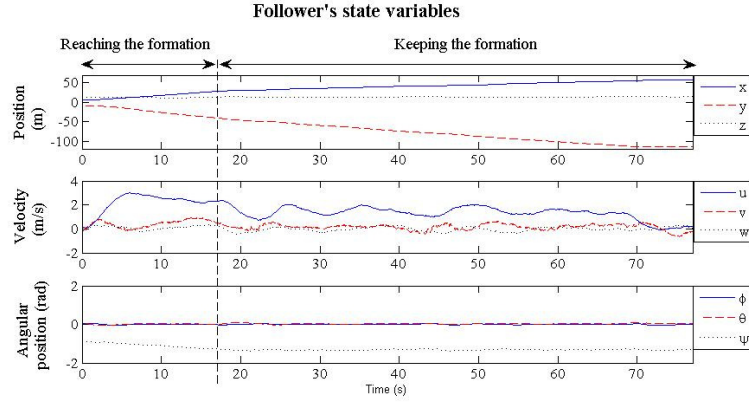


Figure 5.9: The state variables of the follower.

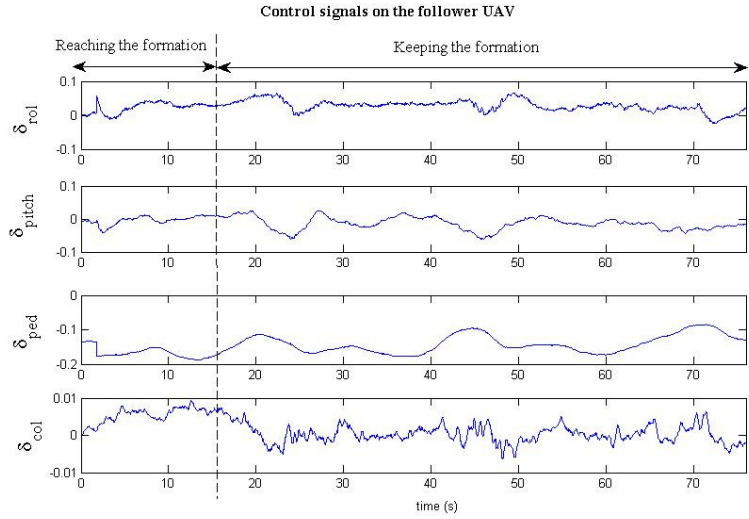


Figure 5.10: Control signals of the follower UAV.

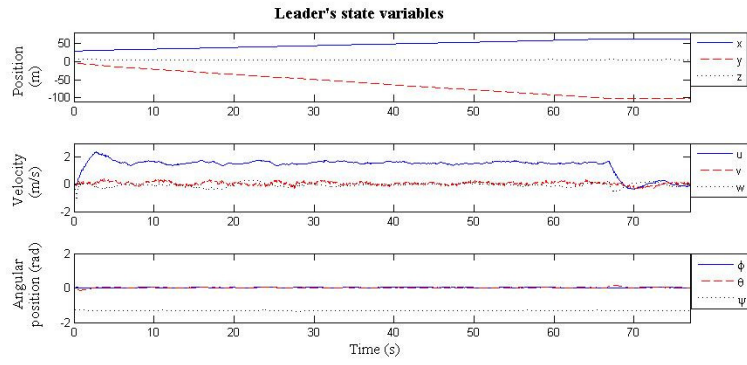


Figure 5.11: The state variable of the leader.

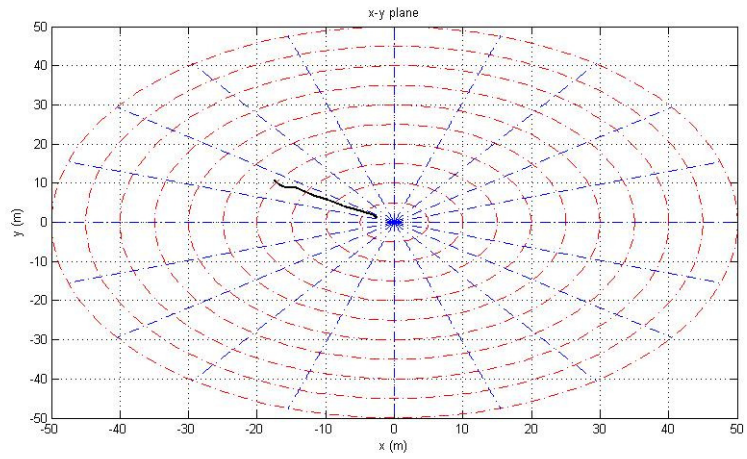


Figure 5.12: The distance of the follower from the desired position.

Fig. 5.15 and Fig. 5.16, respectively. The state variables of the leader and its control signals during the mission are shown in Fig. 5.17 and Fig. 5.18, respectively. The relative distance of follower UAV from the desired position is shown in Fig. 5.19. As it can be seen the follower UAV has finally reached the first circle and the formation has been achieved. A video for the second and third experiments is available at uav.ece.nus.edu.sg/video/2dHybridFormation.mpg.

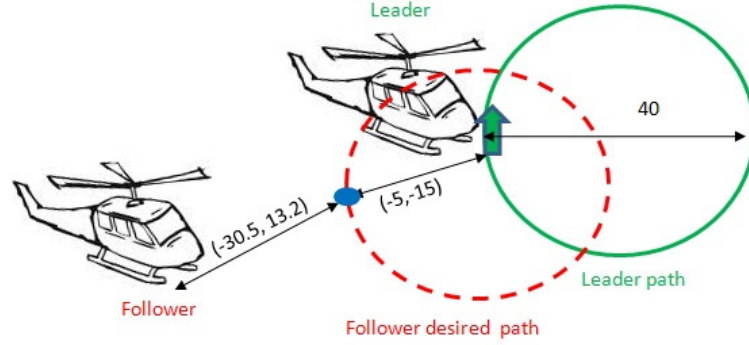


Figure 5.13: The schematic of the scenario for a leader-follower case for tracking a circle.

In the last case, this algorithm is implemented for the 3-D hybrid formation control which was explained in Section 4.6. In this experiment, the initial relative distance between the follower and the desired position is $(dx, dy, dz) = (16.1, 22.5, -14.7)$, and the distance between the desired position and the desired position is $(dx, dy, dz) = (15, 10, 10)$. The UAVs' position are shown in Fig. 5.20. The projection of the relative distance onto the x-y plane is shown in Fig. 5.21. In this experiment, after a while, the formation has been reached and it has been successfully maintained. A video of this experiment is available at <http://uav.ece.nus.edu.sg/video/hybridformation.mpg>.

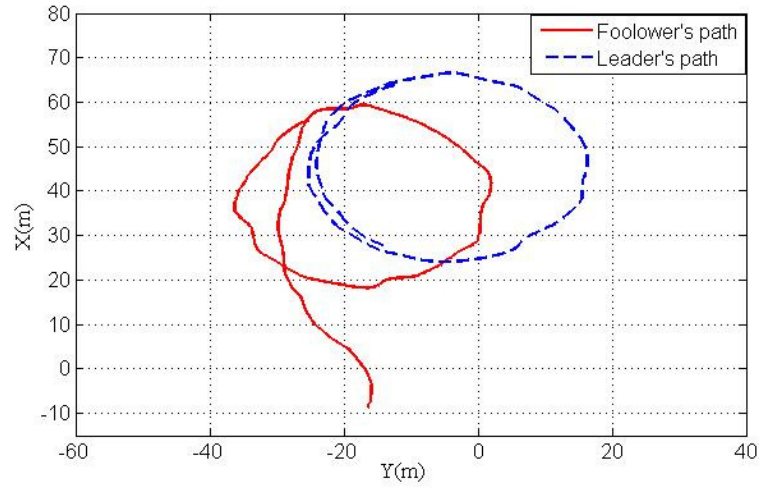


Figure 5.14: The position of the UAVs in the x-y plane.

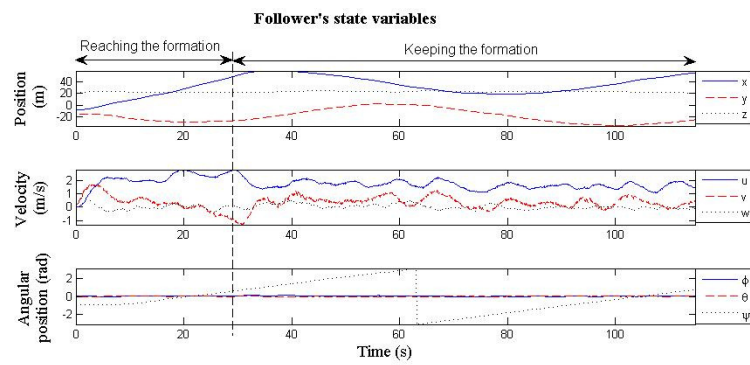


Figure 5.15: The state variables of the follower.

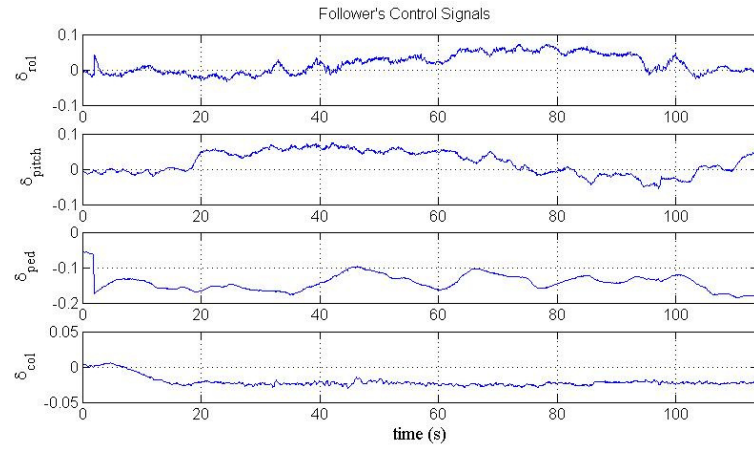


Figure 5.16: Control signals of the follower UAV.

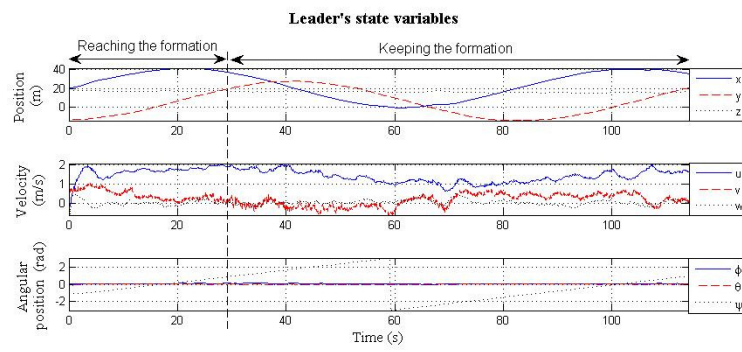


Figure 5.17: The state variables of the leader.

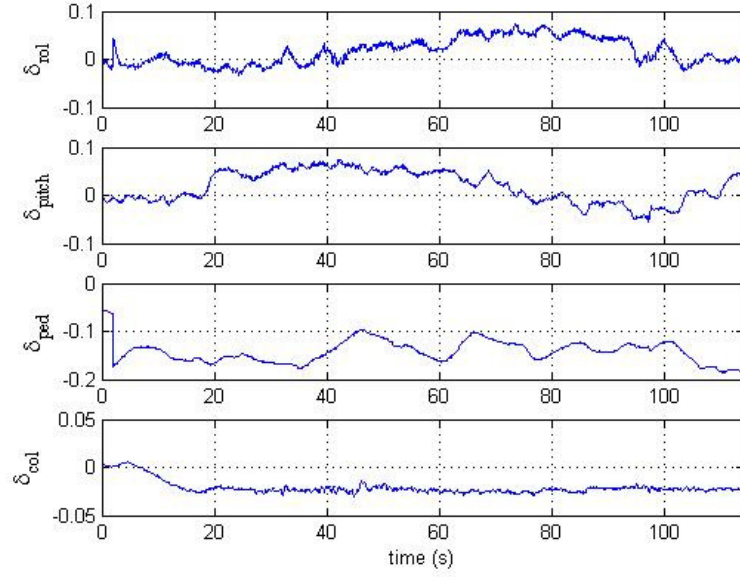


Figure 5.18: Control signals of the leader UAV.

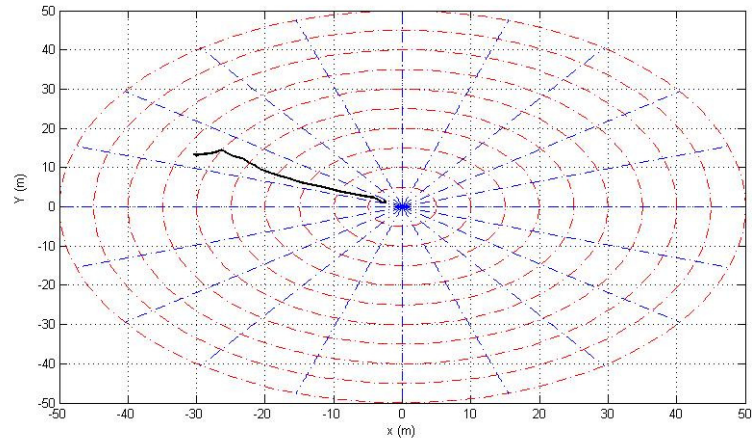


Figure 5.19: The distance of the follower from the desired position.

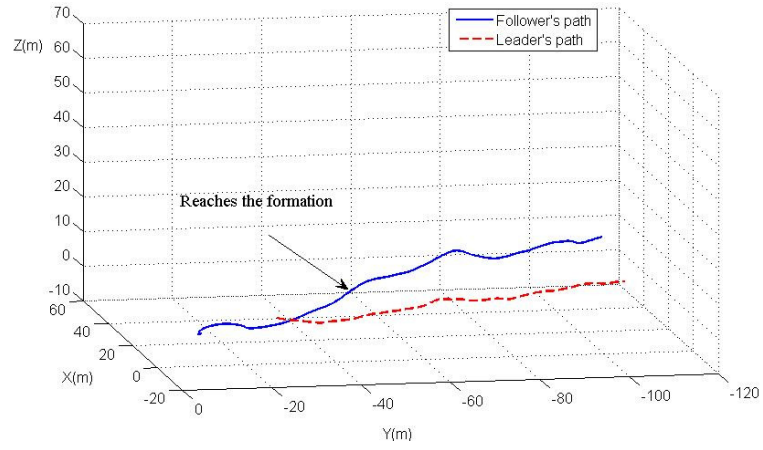


Figure 5.20: The position of the UAVs.

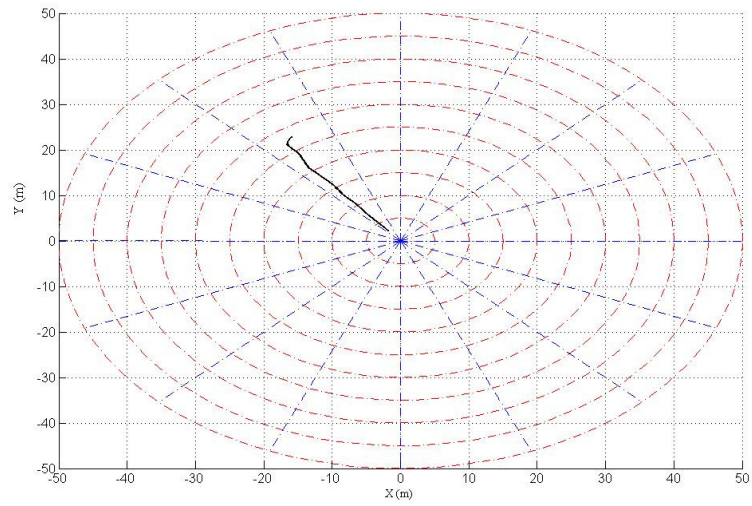


Figure 5.21: The relative distance between the UAVs projected onto x-y plane.

5.5 Conclusion

In this chapter, The implementation issues for the proposed hybrid formation algorithm were investigated. To implement the algorithm, an interface layer was introduced which connects the discrete supervisor to the regulation layer of the UAV. This interface layer is composed of two main blocks: the detection block to generate the detection events based on the plant continuous signals; and the actuator block to convert discrete commands of the supervisor to a continuous form , applicable to the plant. Also, a method was introduced to smoothly generate control signals during the transition through the partitioned regions. Several actual flight tests were conducted to verify the algorithm.

Chapter 6

Conclusions

This thesis proposed a formal method for the hybrid supervisory control mechanism for the formation of unmanned helicopters. Formation is a typical cooperative task and is composed of several subtasks for which a hybrid supervisor is required to manage the logical rules and govern the continuous low level control of the system. Therefore, the thesis has addressed two main problems: (1) developing a hybrid control structure for a single unmanned helicopter in which the low level control and the discrete decision rules are integrated. (2) developing a hybrid control for team of UAVs doing a formation mission.

Motivating by these two problems, the main contributions of the dissertation can be described as follows:

- Focusing on the low level control of the UAVs, a two-layer controller was developed for the NUS UAV helicopter. The lower level is an H_∞ controller, which is responsible for the control of the attitude of the UAV. It provided a stable and robust control performance for the UAV in the hovering mode. The higher layer aims at the position control of the UAV and consists of a P-controller combined with a nonlinear transformation. This control structure makes the UAV able to perform the path tracking mission. The control structure was im-

plemented on the NUS UAV helicopter and experimental results showed that the UAV is able to follow the generated path or do stable hovering successfully.

- To distribute the control tasks among the layers, a hierarchical hybrid control structure was developed for a UAV helicopter. This hierarchy consists of three layers: the regulation layer which is responsible for reference tracking; the coordination layer which is responsible for the path planning, and the supervision layer which is responsible for the task scheduling and decision making. Each layer was modelled by an Input/Output hybrid automaton and the discrete transitions and continuous dynamics of the system were simultaneously captured within the hybrid framework. Then, a composed hybrid operation was proposed to synchronise the layers of the control hierarchy and to obtain the whole closed-loop system. Using this control structure, a flight test was conducted to verify the proposed approach. In this experiment the UAV was successfully involved in a mission composed of several successive tasks.
- A new approach of hybrid supervisory control for the leader follower formation problem was proposed. The approach was based on the polar partitioning of the state space. 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, a collision avoidance mechanism was embedded in the control structure

in a modular way. After reaching the formation, the supervisor will keep the obtained configuration. Moreover, the velocity bounds were applied through the design procedure.

- The proposed hybrid supervisory control mechanism was extended to a 3-D space. For this purpose, a spherical partitioning method was used to abstract the system to find a finite state model of the partitioned space. The general procedure is similar to the 2-D case, however, since the model is different, the designed supervisor was accordingly changed to accomplish the formation task.
- The implementation issues for the proposed hybrid formation control method were investigated. To apply the discrete supervisor to the original continuous plant, an interface layer was developed which is composed of two main blocks: the detection block that generates the detection events based on the plant continuous signals; and the actuator block that converts discrete commands of the supervisor to a continuous form, applicable to the plant. Also, a control mechanism was proposed which generate a smooth control signal for the regulation layer while preserve the bisimulation relation between the abstract model and the original continuous system. Furthermore, the proposed formation algorithm was implemented and several actual flight tests were conducted to verify the algorithm.

This dissertation may represent a promising step towards developing and implementing a hybrid supervisory control framework for the cooperative control of UAVs. The method was developed for a leader-follower scenario for two UAV helicopters.

However, there is a potential to extend the results for more complicated tasks and take the advantages of symbolic control techniques for the abstracted model of the system. Therefore, the following directions could be considered for future studies based on the achievements of this thesis:

- The algorithm was developed for two UAV helicopters. A practically important problem is to extend the method for a multi-follower scenario. In this case, the formation specification is a global task for a team of UAVs and to develop a decentralized control structure, it is required to design local controllers to satisfy the global specification. Another important challenge is that a more advance collision avoidance mechanism is required. Here, the collision avoidance can be modelled by a safety property and using symbolic control techniques, a hybrid controller can be designed to satisfy this goal.
- Having a more advance collision avoidance mechanism, the method can be used to develop a hybrid formation reconfiguration mechanism. Indeed, an interesting feature is to have a team of UAVs that can quickly respond and reconfigure for different tasks or scenarios. For instance, to avoid an obstacle one way is to breaking the formation and after the avoidance of the obstacle, the team members can merge and reconfigure back to the original formation. This feature, also improve the team flexibility in the sense that the formation can be reconfigured based on the assigned tasks which may be changed during the mission.
- The method was developed for a mass point model of the unmanned helicopters,

whereas the approach can be extended to more complex multi-affine dynamics. Furthermore, through the design procedure, only the velocity constraints were applied while it is desired to consider other constraints such as acceleration bounds and turning rates.

Bibliography

- [1] “The unmanned systems integrated roadmap fy2011-2036,” Department of Defense of United States of America, Tech. Rep. Reference Number: 11-S-3613, 2011.
- [2] P. Doherty, P. Haslum, F. Heintz, T. Merz, P. Nyblom, T. Persson, and B. Wingman, “A distributed architecture for autonomous unmanned aerial vehicle experimentation,” in *In Proceedings of the 7th International Symposium on Distributed Autonomous Robotic Systems*, 2004, pp. 221–230.
- [3] M. J. Vachon, R. J. Ray, K. R. Walsh, and K. Ennix, “F/a-18 performance benefits measured during the autonomous formation flight project,” NASA Dryden Flight Research Center, Edwards, CA, Tech. Rep. NASA-TM-2003-210734, September 2003.
- [4] B. Yun, B. Chen, K. Lum, and T. Lee, “A leader-follower formation flight control scheme for uav helicopters,” in *Automation and Logistics, 2008. ICAL 2008. IEEE International Conference on*, Sept. 2008, pp. 39–44.
- [5] N. Metni and T. Hamel, “A uav for bridge inspection: Visual servoing control law with orientation limits,” *Automation in Construction*, vol. 17, no. 1, pp. 3–10, 2007.

- [6] A. Ahmadzadeh, G. Buchman, P. Cheng, A. Jadbabaie, J. Keller, V. Kumar, and G. Pappas, “Cooperative control of uavs for search and coverage,” *Proceedings of the AUVSI Conference on Unmanned Systems*, 2006.
- [7] P. Doherty and P. Rudol, “A uav search and rescue scenario with human body detection and geolocalization,” in *AI 2007: Advances in Artificial Intelligence*, M. Orgun and J. Thornton, Eds., vol. 4830. Springer Berlin, Heidelberg, 2007, pp. 1–13.
- [8] S. Sasa, Y. Matsuda, M. Nakadate, and K. Ishikawa, “Ongoing research on disaster monitoring uav at jaxas aviation program group,” in *SICE Annual Conference, 2008*, Aug. 2008, pp. 978–981.
- [9] M. Bryson and S. Sukkarieh, “Co-operative localisation and mapping for multiple uavs in unknown environments,” in *Aerospace Conference, 2007 IEEE*, March 2007, pp. 1–12.
- [10] A. Puri, K. Valavanis, and M. Kontitsis, “Statistical profile generation for traffic monitoring using real-time uav based video data,” in *Control and Automation, 2007. MED '07. Mediterranean Conference on*, June 2007, pp. 1–6.
- [11] J. Tian, L. Shen, and Y. Zheng, “Formulation and a moga based approach for multi-uav cooperative reconnaissance,” in *CDVE*, 2006, pp. 99–106.
- [12] X. Tian, Y. Bar-Shalom, and K. R. Pattipati, “Multi-step look-ahead policy for autonomous cooperative surveillance by uavs in hostile environments,” in *CDC*, 2008, pp. 2438–2443.
- [13] R. Fierro, P. Song, A. Das, and V. Kumar, “Cooperative control of robot for-

- mations,” in *Cooperative Control and Optimization*, ser. Applied Optimization, R. Murphey, P. M. Pardalos, P. M. Pardalos, and D. W. Hearn, Eds. Springer US.
- [14] M. Karimadini and H. Lin, “Fault-tolerant cooperative tasking for multi-agent systems,” *International Journal of Control*, vol. 84, no. 12, pp. 2092–2107, 2011.
 - [15] M. Karimadini, “Cooperative tasking for multi-agent systems,” Ph.D. dissertation, National University of Singapore, 2011.
 - [16] M. Karimadini and H. Lin, “Guaranteed global performance through local coordinations,” *Automatica*, vol. 47, no. 5, pp. 890–898, 2011.
 - [17] G. Lee and N. Y. Chong, “Decentralized formation control for small-scale robot teams with anonymity,” *Mechatronics*, vol. 19, no. 1, pp. 85–105, 2009.
 - [18] M. Zamani, “Formation and reconfiguration control for multi-robotic systems,” Master’s thesis, National University of Singapore, 2009.
 - [19] B. Anderson, B. Fidan, C. Yu, and D. Walle, “Uav formation control: Theory and application,” in *Recent Advances in Learning and Control*, ser. Lecture Notes in Control and Information Sciences, V. Blondel, S. Boyd, and H. Kimura, Eds. Springer Berlin, Heidelberg, 2008.
 - [20] D. van der Walle, B. Fidan, A. Sutton, C. Yu, and D. Anderson, “Non-hierarchical uav formation control for surveillance tasks,” in *American Control Conference*, 2008, pp. 777–782.
 - [21] R. Beard, T. McLain, D. Nelson, D. Kingston, and D. Johanson, “Decentralized

- cooperative aerial surveillance using fixed-wing miniature uavs,” *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1306–1324, 2006.
- [22] T. Sobh and B. Benhabib, “Discrete event and hybrid systems in robotics and automation: an overview,” *Robotics Automation Magazine, IEEE*, vol. 4, no. 2, pp. 16–19, 1997.
- [23] M. Dong, B. M. Chen, G. Cai, and K. Peng, “Development of a real-time on-board and ground station software system for a uav helicopter,” *AIAA J. Aerosp. Comput., Inf., Commun*, vol. 4, no. 8, pp. 933–955, 2007.
- [24] M. Fatemi, J. Millan, J. Stevenson, T. Yu, and S. O’Young, “Discrete event control of an unmanned aircraft,” in *Proc. 9th IEEE Int. Workshop on Discrete Event Systems*, 2008, pp. 352–357.
- [25] G. Le Lann, “An analysis of the ariane 5 flight 501 failure - a system engineering perspective,” in *Proceedings of the 1997 international conference on Engineering of computer-based systems*, ser. ECBS’97. IEEE Computer Society, 1997, pp. 339–346.
- [26] X. Koutsoukos, P. Antsaklis, J. Stiver, and M. Lemmon, “Supervisory control of hybrid systems,” *Proceedings of the IEEE*, vol. 88, no. 7, pp. 1026–1049, Jul. 2000.
- [27] P. Tabuada, *Verification and control of hybrid systems: a symbolic approach*. Springer-Verlag New York Inc, 2009.
- [28] P. Antsaklis, J. Stiver, and M. Lemmon, “Hybrid system modeling and autonomous control systems,” in *Hybrid Systems*, ser. Lecture Notes in Computer

- Science, R. Grossman, A. Nerode, A. Ravn, and H. Rischel, Eds. Springer Berlin, Heidelberg, 1993, vol. 736, pp. 366–392.
- [29] P. Antsaklis and A. Nerode, “Hybrid control systems: An introductory discussion to the special issue,” *IEEE Trans. Autom. Control*, vol. 43, no. 4, pp. 457–460, Apr. 1998.
- [30] I. Kotini and G. Hassapis, “A hybrid automaton model of the cement mill control,” *IEEE Trans. Control Syst. Technol.*, vol. 16, no. 4, pp. 676–690, 2008.
- [31] C. E. Seah and I. Hwang, “Stochastic linear hybrid systems: Modeling, estimation, and application in air traffic control,” *IEEE Trans. Control Syst. Technol.*, vol. 17, no. 3, pp. 563–575, May 2009.
- [32] M. Oishi, I. Mitchell, A. Bayen, and C. Tomlin, “Invariance-preserving abstractions of hybrid systems: Application to user interface design,” *IEEE Trans. Control Syst. Technol.*, vol. 16, no. 2, pp. 229–244, 2008.
- [33] D. Pepyne and C. Cassandras, “Optimal control of hybrid systems in manufacturing,” *Proc. IEEE*, vol. 88, no. 7, pp. 1108–1123, Jul. 2000.
- [34] S. Bortoff, “The university of toronto rc helicopter: a test bed for nonlinear control,” in *Control Applications, 1999. Proceedings of the 1999 IEEE International Conference on*, vol. 1, 1999, pp. 333–338.
- [35] G. Cai, B. Chen, K. Peng, M. Dong, and T. Lee, “Modeling and control system design for a uav helicopter,” in *Proc. 14th IEEE Mediterranean Conf. Control and Automation*, June 2006, pp. 1–6.

- [36] A. Karimoddini, H. Lin, B. Chen, and T. H. Lee, “Developments in hybrid modeling and control of unmanned aerial vehicles,” in *Proc. IEEE Conf. Control and Automation*, 2009, pp. 228–233.
- [37] M. Hirata, Y. Hashimoto, S. Noguchi, and S. Adachi, “A hybrid modeling method for mechanical systems,” *Mechatronics*, vol. 20, no. 1, pp. 59–66, 2010.
- [38] R. Zhu, D. Sun, and Z. Zhou, “Integrated design of trajectory planning and control for micro air vehicles,” *Mechatronics*, vol. 17, no. 4–5, pp. 245–253, 2007.
- [39] T. J. Koo, F. Hoffmann, F. H. Mann, H. Shim, B. Sinopoli, and S. Sastry, “Hybrid control of an autonomous helicopter,” in *Proc. IFAC Workshop on Motion Control*, 1998, pp. 285–290.
- [40] S. Bayraktar, G. Fainekos, and G. Pappas, “Experimental cooperative control of fixed-wing unmanned aerial vehicles,” in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 4, 2004, pp. 4292–4298.
- [41] J. Gillula, H. Huang, M. Vitus, and C. Tomlin, “Design of guaranteed safe maneuvers using reachable sets: Autonomous quadrotor aerobatics in theory and practice,” in *Proc. IEEE Conf. Robotics and Automation*, 2010, pp. 1649–1654.
- [42] R. Naldi, L. Marconi, and L. Gentili, “Robust takeoff and landing for a class of aerial robots,” in *Proc. 48th IEEE conf. Decision and Control held jointly with 28th Chinese Control Conf. (CDC/CCC)*, 2009, pp. 3436–3441.
- [43] C. W. Seibel, J.-M. Farines, and J. E. R. Cury, “Towards using hybrid automata for the mission planning of unmanned aerial vehicles,” in *Hybrid Systems V*. London, UK: Springer-Verlag, 1999, pp. 324–340.

- [44] E. Frazzoli, M. Dahleh, and E. Feron, “Robust hybrid control for autonomous vehicle motion planning,” in *Proc. 39th IEEE Conf. Decision and Control*, vol. 1, 2000, pp. 821–826.
- [45] S. Wei, M. Zefran, and R. DeCarlo, “Optimal control of robotic systems with logical constraints: Application to uav path planning,” in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, May 2008, pp. 176–181.
- [46] T. Schouwenaars, B. Mettler, E. Feron, and J. How, “Hybrid architecture for full-envelope autonomous rotorcraft guidance,” in *American Helicopter Society 59th Annual Forum, Arizona*, 2003.
- [47] M. D. Mesarović, D. Macko, and Y. Takahara, *Theory of Hierarchical, Multilevel Systems*, ser. Mathematics in Science and Engineering. Academic Press, 1970, vol. 68.
- [48] W. Findeisen, F. N. Bailey, M. Brdeys, K. Malinowski, P. Tatjewski, and J. Wozniak, *Control and coordination in hierarchical systems*, ser. Appl. Syst. Anal. Chichester: Wiley, 1980.
- [49] R. G. Simmons, T. Smith, M. B. Dias, D. Goldberg, D. Hershberger, A. Stentz, and R. Zlot, “A layered architecture for coordination of mobile robots,” in *Multi-Robot Systems: From Swarms to Intelligent Automata*, A. C. Schultz and L. E. Parker, Eds. Kluwer, 2002.
- [50] N. Lynch, R. Segala, and F. Vaandrager, “Hybrid i/o automata,” *Information and Computation*, vol. 185, no. 1, pp. 105–157, 2003.
- [51] J. How, E. King, and Y. Kuwata, “Flight demonstrations of cooperative con-

- trol for uav teams,” in *AIAA 3rd Unmanned Unlimited Technical Conference, Workshop and Exhibit*, 2004.
- [52] M. De Gennaro and A. Jadbabaie, “Formation control for a cooperative multi-agent system using decentralized navigation functions,” in *American Control Conference*, June 2006, pp. 1346–1351.
- [53] T. Paul, T. Krogstad, and J. Gravdahl, “Modelling of uav formation flight using 3d potential field,” *Simulation Modelling Practice and Theory*, vol. 16, no. 9, pp. 1453–1462, 2008.
- [54] G. Hassan, K. Yahya, and I. ul Haq, “Leader-follower approach using full-state linearization via dynamic feedback,” in *Emerging Technologies, ICET '06. International Conference on*, 2006, pp. 297–305.
- [55] I. Shames, B. Fidan, and B. D. Anderson, “Minimization of the effect of noisy measurements on localization of multi-agent autonomous formations,” *Automatica*, vol. 45, no. 4, pp. 1058–1065, 2009.
- [56] N. Linorman and H. Liu, “Formation uav flight control using virtual structure and motion synchronization,” in *American Control Conference*. IEEE, 2008, pp. 1782–1787.
- [57] M. Zamani and H. Lin, “Structural controllability of multi-agent systems,” in *American Control Conference*, june 2009, pp. 5743–5748.
- [58] J. Jansson and F. Gustafsson, “A framework and automotive application of collision avoidance decision making,” *Automatica*, vol. 44, no. 9, pp. 2347 – 2351, 2008.

- [59] R. Schlanbusch, R. Kristiansen, and P. J. Nicklasson, “Spacecraft formation reconfiguration with collision avoidance,” *Automatica*, vol. 47, no. 7, pp. 1443 – 1449, 2011.
- [60] B. Cetin, M. Bikdash, and F. Hadaegh, “Hybrid mixed-logical linear programming algorithm for collision-free optimal path planning,” *Control Theory Applications, IET*, vol. 1, no. 2, pp. 522–531, march 2007.
- [61] R. Fierro, A. Das, V. Kumar, and J. Ostrowski, “Hybrid control of formations of robots,” in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 1, 2001, pp. 157–162.
- [62] D. Cernega and R. Solea, “Hybrid control structure for multi-robot formation,” in *Artificial Neural Networks ICANN 2010*, ser. Lecture Notes in Computer Science, K. Diamantaras, W. Duch, and L. Iliadis, Eds. Springer Berlin, Heidelberg, 2010, vol. 6353, pp. 307–316.
- [63] S. Zelinski, T. Koo, and S. Sastry, “Hybrid system design for formations of autonomous vehicles,” in *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, vol. 1, 2003, pp. 1 – 6 Vol.1.
- [64] D. M. Stipanovic, G. Inalhan, R. Teo, and C. J. Tomlin, “Decentralized overlapping control of a formation of unmanned aerial vehicles,” *Automatica*, vol. 40, no. 8, pp. 1285–1296, 2004.
- [65] Q. Chen and U. Ozguner, “A hybrid system model and overlapping decomposition for vehicle flight formation control,” in *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, vol. 1, 2003, pp. 516–521.

- [66] A. Iftar and U. Ozguner, “Overlapping decompositions, expansions, contractions, and stability of hybrid systems,” *Automatic Control, IEEE Transactions on*, vol. 43, no. 8, pp. 1040–1055, Aug 1998.
- [67] P. Ramadge and W. Wonham, “The control of discrete event systems,” *Proceedings of the IEEE*, vol. 77, no. 1, pp. 81–98, Jan 1989.
- [68] R. Alur, T. Henzinger, G. Lafferriere, and G. Pappas, “Discrete abstractions of hybrid systems,” *Proceedings of the IEEE*, vol. 88, no. 7, pp. 971–984, Jul. 2000.
- [69] G. Lafferriere and S. Yovine, “A new class of decidable hybrid systems,” in *In Hybrid Systems : Computation and Control*. Springer, 1999, pp. 137–151.
- [70] G. Lafferriere, G. J. Pappas, and S. Sastry, “O-minimal hybrid systems,” *Mathematics of Control, Signals, and Systems*, vol. 13, pp. 1–21, 2000.
- [71] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya, “What’s decidable about hybrid automata?” Ithaca, NY, USA, Tech. Rep., 1995.
- [72] C. Belta and L. Habets, “Constructing decidable hybrid systems with velocity bounds,” in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 1, 2004, pp. 467–472.
- [73] —, “Controlling a class of nonlinear systems on rectangles,” *Automatic Control, IEEE Transactions on*, vol. 51, no. 11, pp. 1749–1759, 2006.
- [74] M. Broucke, “Reach control on simplices by continuous state feedback,” in *American Control Conference*. IEEE, 2009, pp. 1154–1159.
- [75] L. Habets, P. Collins, and J. van Schuppen, “Reachability and control synthe-

- sis for piecewise-affine hybrid systems on simplices,” *Automatic Control, IEEE Transactions on*, vol. 51, no. 6, pp. 938–948, 2006.
- [76] R. Enns and J. Si, “Helicopter trimming and tracking control using direct neural dynamic programming,” *Neural Networks, IEEE Transactions on*, vol. 14, no. 4, pp. 929 – 939, july 2003.
- [77] A. Isidori, L. Marconi, and A. Serrani, “Robust nonlinear motion control of a helicopter,” in *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, vol. 5, 2001, pp. 4586 –4591 vol.5.
- [78] K. Peng, M. Dong, B. M. Chen, G. Cai, K. Y. Lum, and T. H. Lee, “Design and implementation of a fully autonomous flight control system for a uav helicopter,” in *Control Conference, 2007. CCC 2007. Chinese*, June, Ed., 2007, pp. 662–667.
- [79] D. Shim, H. Kim, and S. Sastry, “Decentralized nonlinear model predictive control of multiple flying robots,” in *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, vol. 4, dec. 2003, pp. 3621 – 3626 vol.4.
- [80] Q.-G. Wang, C. Lin, Z. Ye, G. Wen, Y. He, and C. C. Hang, “A quasi-lmi approach to computing stabilizing parameter ranges of multi-loop pid controllers,” *Journal of Process Control*, vol. 17, no. 1, pp. 59 – 72, 2007.
- [81] B. L. Stevens and F. L. Lewis, *Aircraft control and simulation*. Wiley, New York, 1992.
- [82] K. Peng, G. Cai, B. M. Chen, M. Dong, and T. H. Lee, “Comprehensive modeling and control of the yaw dynamics of a uav helicopter,” in *Control Conference, CCC 2006. Chinese*, aug. 2006, pp. 2087–2092.

- [83] B. M. Chen, *Robust and H_∞ control*. Springer, New York, London, 2000.
- [84] I. Postlethwaite and A. G. Macfarlane, *A Complex Variable Approach to the Analysis of Linear Multivariable Feedback Systems*. Lecture Notes in Control Information Sciences, 1979, vol. 12, pp. 58–76.
- [85] G. Cai, B. M. Chen, T. H. Lee, and M. Dong, “Design and implementation of a hardware-in-the-loop simulation system for small-scale uav helicopters,” *Mechanics*, vol. 19, no. 7, pp. 1057 – 1066, 2009.
- [86] G. Cai, B. M. Chen, X. Dong, , and T. H. Lee, “Supplementary document: Linearized models of helion uav and the corresponding inner-loop controllers,” National University of Singapore, Tech. Rep., Tech. Rep., 2010.
- [87] J. Gadewadikar, F. L. Lewis, K. Subbarao, and B. M. Chen, “Structured h_∞ command and control loop design for unmanned helicopters,” *AIAA Journal of Guidance, Control and Dynamics*, vol. 31, no. 4, pp. 1093–1102, 2008.
- [88] J. Liu, X. Liu, T. Koo, B. Sinopoli, S. Sastry, and E. Lee, “A hierarchical hybrid system model and its simulation,” in *Proc. 38th IEEE Conf. Decision and Control*, vol. 4, 1999, pp. 3508–3513.
- [89] R. Alur and D. L. Dill, “A theory of timed automata,” *Theoretical Computer Science*, vol. 126, no. 2, pp. 183–235, 1994.
- [90] K. H. Johansson, “Introduction to hybrid systems,” Lecture notes, Department of Signals, Sensors and Systems. Royal Institute of Technology, April 2005.
- [91] N. Lynch, R. Segala, and F. Vaandrager, “Hybrid i/o automata revisited,” in

- Proceedings Fourth International Workshop on Hybrid Systems: Computation and Control (HSCC'01)*. Springer-Verlag, 2001, pp. 403–417.
- [92] S. Rashid and J. Lygeros, “Hybrid systems: modeling, analysis and control - open hybrid automata and composition,” Lecture notes, University of California at Berkley, 1999.
 - [93] T. A. Henzinger, P. Ho, and H. Wong-Toi, “Hytech: a model checker for hybrid systems,” *International Journal on Software Tools for Technology Transfer (STTT)*, vol. 1, pp. 110–122.
 - [94] R. Alur, C. Courcoubetis, T. Henzinger, and P. Ho, “Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems,” in *Hybrid Systems*, R. Grossman, A. Nerode, A. Ravn, and H. Rischel, Eds. Springer Berlin, Heidelberg, 1993, vol. 736, pp. 209–229.
 - [95] P. Antsaklis, J. Stiver, and M. Lemmon, “Hybrid system modeling and autonomous control systems,” in *Hybrid Systems*, ser. Lecture Notes in Computer Science, R. Grossman, A. Nerode, A. Ravn, and H. Rischel, Eds. Springer Berlin / Heidelberg, vol. 736, pp. 366–392.
 - [96] P. Tabuada, G. J. Pappas, and P. Lima, “Feasible formations of multi-agent systems,” *Proceedings of the American Control Conference*, pp. 56–61, 2001.
 - [97] R. Kumar and V. K. Garg, *Modeling and Control of Logical Discrete Event Systems*, ser. The Springer International Series in Engineering and Computer Science. Springer, 1995, vol. 300.
 - [98] K. Peng, G. Cai, B. M. Chen, M. Dong, K. Y. Lum, and T. H. Lee, “Design

- and implementation of an autonomous flight control law for a uav helicopter,” *Automatica*, vol. 45, no. 10, pp. 2333–2338, 2009.
- [99] K. H. Johansson, M. Egerstedt, J. Lygeros, and S. Sastry, “On the regularization of zeno hybrid automata,” *Systems and amp, Control Letters*, vol. 38, no. 3, pp. 141–150, 1999.

Chapter 7

APPENDIX

7.0.1 Proof for Theorem 4

Consider the relation $R = \{(q_Q, q_\xi) | q_Q \in X_Q, q_\xi \in X_\xi, \text{ and } q_Q \in \mathfrak{S}(q_\xi)\}$. We will show that this relation is a bisimulation relation between T_Q and T_ξ .

Let's start with the first condition of the bisimulation relation, defined in Definition 6. For any $q_Q \in X_{Q_0}$ there exists a region $R_{i,j}$ such that $q_Q \in R_{i,j}$. For this region, there exists a label, $\tilde{R}_{i,j}$ such that $R_{i,j} = \mathfrak{S}(\tilde{R}_{i,j})$ and $\tilde{R}_{i,j} \in X_{\xi_0}$. Hence, $(q_Q, \tilde{R}_{i,j}) \in R$. Conversely, it can be similarly shown that for any $q_\xi \in X_{\xi_0}$, there exists a $q_Q \in X_{Q_0}$ such that $(q_\xi, q_Q) \in R$.

For the second condition of the bisimulation relation, following from the definition of T_ξ , for any $(q_Q, q_\xi) \in R$ and $q_Q \xrightarrow{u}_Q q'_Q$, there exists a transition $q_\xi \xrightarrow{u}_\xi q'_\xi$, where $q'_Q \in \mathfrak{S}(q'_\xi)$ or equivalently $(q'_Q, q'_\xi) \in R$. For the converse case, assume that $q_\xi \xrightarrow{u}_\xi q'_\xi$. According to the definition of R , all $x \in \mathfrak{S}(q_\xi)$ are related to q_ξ . Hence, to prove the second condition of the bisimulation relation, we should investigate it for all $x \in \mathfrak{S}(q_\xi)$. Based on the control construction procedure, the labels u , q_ξ , and q'_ξ can be one of the following cases:

1. $u = C_0$ and $q_\xi = q'_\xi$. In this case, since the controller C_0 makes the region an invariant region (Theorem 2), all of the trajectories starting from any $q_Q \in$

$\mathfrak{S}(q_\xi)$ will remain inside the region $\mathfrak{S}(q_\xi)$. Therefore, for any $q_Q \in \mathfrak{S}(q_\xi)$, there exists a $q'_Q \in \mathfrak{S}(q_\xi)$ such that $q_Q \xrightarrow{u} q'_Q$ and $q'_Q \in \mathfrak{S}(q'_\xi)$.

2. $u \in C_q^s$, $q_\xi \in \{\tilde{R}_{i,j} | 1 \leq i \leq n_r - 1, 1 \leq j \leq n_\theta - 1\}$, and $q'_\xi \in \{\tilde{d}([i,j], [i',j']) | 1 \leq i, i' \leq n_r - 1, 1 \leq j, j' \leq n_\theta - 1\}$. In this case, based on Theorem 3 and Lemma 2, starting from any $q_Q \in \mathfrak{S}(q_\xi)$, the controller C_q^s drives the system trajectory towards the detection element $\mathfrak{S}(q'_\xi)$. Therefore, for any $q_Q \in \mathfrak{S}(q_\xi)$, there exists a $q'_Q \in \mathfrak{S}(q'_\xi)$ such that $q_Q \xrightarrow{u} q'_Q$ and $q'_Q \in \mathfrak{S}(q'_\xi)$.
3. $u \in U_c = \{\hat{d}([i,j], [i',j']) | 1 \leq i, i' \leq n_r - 1, 1 \leq j, j' \leq n_\theta - 1\}$ and $q'_\xi \in \{\tilde{R}_{i',j'} | 1 \leq i' \leq n_r - 1, 1 \leq j' \leq n_\theta - 1\}$, and $q_\xi \in \{\tilde{d}([i,j], [i',j']) | 1 \leq i, i' \leq n_r - 1, 1 \leq j, j' \leq n_\theta - 1\}$. In this case, based on Lemma 2, for any $q_Q \in \mathfrak{S}(q_\xi)$ there exists a controller $v \in C_q^s$ that has led the trajectory of the system from the region $R_{i,j}$ to the point q_Q on the detection element $d([i,j], [i',j'])$. Since $R_{i',j'}$ is the unique adjacent region of the element $R_{i,j}$, common in the detection element $d([i,j], [i',j'])$, based on the definition of the controller for the exit edge and Theorem 3, the controller v leads the trajectory of the system to a point inside the region $R_{i',j'}$ so that the detection event $u = \hat{d}([i,j], [i',j'])$ is generated. Therefore, for any $q_Q \in \mathfrak{S}(q_\xi)$, there exists a $q'_Q \in \mathfrak{S}(q'_\xi)$ such that $q_Q \xrightarrow{u} q'_Q$ and $q'_Q \in \mathfrak{S}(q'_\xi)$.
4. $u \in U_e$ is the external event. In this case, the state of the system does not change, meaning that $q_Q = q'_Q$ and $q_\xi \in q'_\xi$. Therefore, trivially for any $q_Q \in \mathfrak{S}(q_\xi)$ and $q_\xi \xrightarrow{u} q'_\xi$, we have $q_Q \xrightarrow{u} q'_Q$, where $q'_Q \in \mathfrak{S}(q'_\xi)$.

In all of the above mentioned cases, the second condition of the bisimulation relation for the converse case holds true. Hence, T_ξ and T_Q are bisimilar. ■

List of Publications

- book chapter:

1. A. Karimoddini, G. Cai, B.M. Chen, H. Lin, T. H. Lee, “Hierarchical Control Design of a UAV Helicopter,” in *Advances in Flight Control Systems*, INTECH, Vienna, Austria, 2011.

- Journal papers:

1. A. Karimoddini, H. Lin, B. M. Chen, and T. H. Lee, “Hybrid three-dimensional formation control for unmanned helicopters,” *Automatica*, Vol 49, No. 2, 2013, pages 424-433.
2. A. Karimoddini, H. Lin, B. M. Chen, and T. H. Lee, “Hybrid Formation Control of the Unmanned Aerial Vehicles, ” *Mechatronics*, Vol. 21, No. 5, 2011, page 886–898.
3. A. Karimoddini, H. Lin, X. Dong, G. Cai, L. Feng, B. M. Chen, and T. H. Lee, “Hierarchical Hybrid Modelling and Control of the Unmanned Aerial Vehicles,” submitted for publication, 2012.
4. A. Karimoddini, H. Lin, B. M. Chen, and T. H. Lee, “A Bumpless Hybrid Supervisory Control for the Formation of Unmanned Aerial Vehicles,” Submitted for publication.

- Conference papers:

1. A. Karimoddini, H. Lin, B. M. Chen, and T. H. Lee, “A Smooth Symbolic Control Mechanism Over a Partitioned space,” To appear in Proc. of the 2013 American Control Conference, USA, 2013.
2. A. Karimoddini, X. Dong, G. Cai, L. Feng, H. Lin, B. M. Chen, T. H. Lee, “A Composed Hybrid Structure for the Autonomous Flight Control of Unmanned Helicopters,” 18th IFAC World Congress, Italy, 2011.
3. A. Karimoddini, G. Cai, B. M. Chen, H. Lin, and T. H. Lee, “Multi-layer Flight Control Synthesis and Analysis of a Small-scale UAV Helicopter,” 4th IEEE International Conference on Robotics, Automation and Mechatronics, Singapore, 2010.
4. A. Karimoddini, H. Lin, B. M. Chen, and T. H. Lee, “Developments in Hybrid Modeling and Control of Unmanned Aerial Vehicles,” 7th IEEE International Conference on Control and Automation, New Zealand, 2009.