Reactive navigation of a fleet of drones in interaction

Osamah Saif

Par **Osamah SAIF**

Reactive navigation of a fleet of drones in interaction

Thèse présentée
pour l'obtention du grade
de Docteur de l'UTC

Soutenue le 23 mars 2016
Spécialité : Technologie de l'Information et des Systèmes

D2269

Reactive Navigation of a Fleet of Drones in Interaction

Osamah SAIF

Thèse soutenue le 23 mars 2016 devant le jury composé de :

Rapporteurs:

Mohamed BOUTAYEB
Professeur des universités
Université de Lorraine

Nicolas MARCHAND
Directeur de Recherche CNRS
GIPSA-lab, Université de Grenoble

Examinateurs:

Philippe BONNIFAIT
Professeur des universités
Université de Technologie de Compiègne

Dominique LUZEAUX
Ingénieur général de l'armement HDR
DIRISI, Ministère de la Défence

Olivier SIMONIN
Professeur des universités
INSA Lyon - CITI-Inria Lab

Paolo ROBUFFO GIORDANO
Chargé de recherche CNRS
Irisa - Inria

Directrice de Thèse:

Isabelle FANTONI
Directrice de Recherche CNRS
Heudiasyc, Université de Technologie de Compiègne

Université de Technologie de Compiègne

Laboratoire Heudiasyc UMR CNRS 7253

23 - 03 - 2016

Contents

List of Figures

Publications

1. Saif, O., Fantoni, I., and Zavala-Rio, A. (2015). Real-time flocking of multiple-quadrotor system of systems. In *IEEE System of Systems Engineering (SOSE), San Antonio, TX, USA*.

2. Saif, O., Fantoni, I., and Zavala-Rio, A. (2014). Flocking of multiple unmanned aerial vehicles by lqr control. In *IEEE International Conference on Unmanned Aircraft Systems (ICUAS), Orlando, USA, pages 222-228*.

3. Saif, O. and Fantoni, I. (2013). Commande lqr d'une flotte de multiples véhicules aériens. In *Journées Doctorales / Journés Nationales MACS (JD / JNMACS), Strasbourg, France*.

4. Saif, O., Hou, Z., Fantoni, I., and Sanahuja, G. (2013). Simulateur de flotte de drones. In Journées Démonstrateurs en Automatique, Angers, France.

Résumé

De nos jours, les applications utilisant des quadrirotors autonomes sont en plein essor. La surveillance et la sécurité de sites industriels ou sensibles, de zones géographiques pour l ?agriculture par exemple sont quelques-unes des applications les plus célèbres des véhicules aériens sans pilote (UAV). Actuellement, certains chercheurs et scientifiques se concentrent sur le déploiement multi-drones pour l'inspection et la surveillance de vastes zones. L'objectif de cette thèse est de concevoir des algorithmes afin de réaliser une commande de vol en formation distribuée/décentralisée de multi-UAVs en temps réel dans une perspective de systèmes de systèmes.

Tout d'abord, nous avons passé en revue certains travaux récents de la littérature sur la commande de vol en formation et la commande de quadrirotors. Nous avons présenté une brève introduction sur les systèmes de systèmes, leur définition et leurs caractéristiques. Ensuite, nous avons introduit la commande de vol en formation avec ses structures les plus utilisées dans la littérature. Nous avons alors présenté quelques travaux existants traitant du *flocking* (comportement de regroupement en flotte), les méthodes de modélisation les plus utilisés pour les quadrirotors et quelques approches de commande les plus utilisées pour stabiliser des quadrirotors.

Deuxièmement, nous avons utilisé la structure de la commande comportementale pour réaliser un vol en formation de plusieurs UAVs. Nous avons conçu un comportement pour réaliser le vol en formation de multi-UAVs sans fragmentation. Le comportement proposé traite le problème *flocking* dans une perspective globale, c'est-à-dire, nous avons inclus une tendance dans chaque drone pour former une formation.

Les défis des Systèmes de systèmes nous a motivés à chercher des algorithmes de *flocking* et de consensus introduits dans la littérature qui peuvent être utiles pour répondre à ces défis. Cela nous a amenés à proposer quatre lois de commande en visant à être compatibles avec le modèle non linéaire des quadrirotors et pouvant être expérimentés sur des plates-formes réelles. Les lois de commande ont été exécutées à bord de chaque quadrirotor dans la formation et chaque quadrirotor interagit avec ses voisins pour assurer un vol en formation sans collision.

Enfin, nous avons validé nos lois de commande par des simulations et des expériences en temps réel. Pour la simulation, nous avons utilisé un simulateur de multi

quadrirotors développé au laboratoire Heudiasyc. Pour les expériences, nous avons mis en œuvre nos lois de contrôle sur des quadrirotors ArDrone2 évolués dans un environnement intérieur équipé d'un système de capture de mouvement (Optitrack).

Mots clés : *Commande de vol en formation, Flocking et algorithmes de consensus, Commande LQR, Commande comportementale, Véhicules aériens sans pilote (UAV).*

Abstract

Nowadays, applications of autonomous quadrotors are increasing rapidly. Surveillance and security of industrial sites, geographical zones for agriculture for example are some popular applications of Unmanned Aerial Vehicles (UAVs). Nowadays, researchers and scientists focus on the deployment of multi-UAVs for the inspection and the surveillance of large areas. The objective of this thesis is to design algorithms and techniques to perform a real-time distributed/decentralized multi-UAVs flight formation control, from a system of systems perspective.

Firstly, we reviewed recent works of the literature about flight formation control and the control of quadrotors. We presented a brief introduction about systems of systems, their definition and characteristics. Then, we introduced the flight formation control with its most used structures in the literature, some existing works dealing with flocking. Finally, we presented the most used modeling methodologies for quadrotors and some control approaches that are used to stabilize quadrotors.

Secondly, we used the behavioral-based control structure to achieve a multiple-UAV flocking. We conceived a behavior intending to address the control design towards a successful achievement of the flocking task without fragmentation. The proposed behavior treats the flocking problem from a global perspective, that is, we included a tendency of separated UAVs to form a flock.

System of systems challenges motivated us to look for flocking and consensus algorithms introduced in the literature that could be helpful to answer to these challenges. This led us to propose four flocking control laws aiming at being compatible with the nonlinear model of quadrotors and at being implemented on experimental platforms. The control laws were run aboard each quadrotor in the flock. By running the control law, each quadrotor interacts with its neighbors to ensure a collision-free flocking.

Finally, we validated our proposed control laws by simulations and real-time experiments. For the simulation, we used a PC-based simulator of flock of multiple quadrotors which was developed at Heudiasyc laboratory. For experiments, we implemented our control laws on ArDrone2 quadrotors evolved in an indoor environment equipped with an Optitrack motion capture system.

Key Words: *Flight formation control, Flocking and consensus algorithms, LQR*

control, Behavioral-based control, Unmanned Aerial Vehicles (UAV).

Introduction

Recently, relatively cheap Unmanned Aerial Vehicles (UAV), equipped by sensing and actuation capabilities, are emerging rapidly. Moreover, with the miniaturization revolution, these devices are able to perform local decision-making as well as short-range communications. These novelties motivate scientists and researches to raise different questions about the techniques that could be introduced to coordinate and control these devices. One of the biggest challenges in this new field of research is to define local interaction rules, between these devices, that lead to a global desired behavior, in other word, "*Think globally, act locally*". A global desired behavior could be any objective or mission achieved by one device, for instance, surveillance, inspection, transportation, etc. However, this objective, if it is done by one device, could be expensive in terms of time, reliability and computational power. Therefore, the question could be reformulated to be "How should the interaction rules and the control algorithms be designed to efficiently achieve the global desired objective?"

"System of Systems Engineering" (SoSE) is one of the recent research fields that seeks finding answers of the previous question. In fact, SoSE seeks to propose methodologies to optimize the interactions of stand-alone systems in order to satisfy a global objective. Several challenges are addressed and raised in this field, such as, operational and managerial independence of systems, emergent behaviors, heterogeneity, scalability, etc.

Before technological systems scientists, the study of collective behaviors of animals that lead to a global objective, was inaugurated by biologists. They studied flocks of birds, swarms of insects, herds of quadruped and schools of fish. They tried to understand the secret of free-collision, harmonic and collective motion of individuals. Moreover, they searched to discover motivations that lead animals to aggregate in groups like flocks of birds. Detailed discussions and studies could be found in [Partridge, 1982] [Couzin, 2009].

In robotics and control theory, scientists and researches were interested in imitating the collective behavior of animals using autonomous terrestrial robots, UAVs and submarine robots. From swarm robotics to flight formation, techniques and structures are proposed, such as, leader-followers, virtual structures and behavioral-based control. The most appropriate control structure, that can answer

SoSE challenges, is the behavioral-based control, since it is biologically inspired and defines local rules for robots to achieve a global objective.

However, since *more questions leads to more knowledge* and *vice versa*, we try in this work, based on works in the literature, to seek initial answers to the following questions:

- How to control and navigate a fleet of UAVs taking into account model nonlinearities and collision avoidance?
- What are the effects of model uncertainties and exogenous perturbations on the System of Systems?
- How can individuals achieve local rules in a precise and robust way?
- While a System of Systems is supposed to be scalable, is it fault tolerant?

To answer all or some of these questions, we design algorithms and techniques to perform a real-time distributed multi-UAVs system of systems that can navigate in a collective motion. These algorithms are applied on a simulator of fleet of UAVs developed at Heudiasyc laboratory. Moreover, based on the results obtained in the simulator, we improve our algorithms and apply them on an experimental platform of multiple quadrotors and a motion capture system "Optitrack". We believe that, at the end of the simulation and experimental work, we can find some answers of the raised questions.

This thesis was carried out in the framework of the Labex MS2T, which was funded by the French Government, through the program "Investments for the future" managed by the National Agency for Research (Reference ANR-11-IDEX-0004-02). The device Labex (Laboratories of Excellence) aims to provide the best national laboratories with international visibility and significant resources to enable them to compete with their foreign counterparts, to attract researchers and academics of international renown and build a political integrated research, training and development of high level. The Labex MS2T (Control of Technological Systems-of-Systems) is a multi-disciplinary scientific problem that targets a scope of application that may be very large. Backed by the experience of the three partner laboratories, Labex MS2T will make significant progress on three significant social issues : Transport and Mobility, Security, Health (ICT and Health Engineering).[1]

The work in this thesis is organized as the following:

In the first chapter, we review different recent works of the literature about System of Systems, flight formation control and the control of quadrotors. Firstly,

[1] https://www.hds.utc.fr/labex-ms2t-484

we present a brief introduction about systems of systems, their definition and characteristics. Then, we introduce the flight formation control with its different structures that are found in the literature. After that, we introduce the most used modeling methodologies for quadrotors. We present some control approaches that are used to stabilize quadrotors. Finally, we present some existing works dealing with flocking.

In the second chapter, we use the behavioral-based control structure to achieve a multiple-UAV flocking. We conceive a behavior intending to address the control design towards a successful achievement of the flocking task without fragmentation. The proposed behavior treats the flocking problem from a global perspective, that is, we include a tendency of separated UAVs to form a flock.

In the third chapter, we are interested in performing real-time flocking of multiple UAVs in the context of system of systems. We propose control methods that are based on the flocking and consensus algorithm introduced in the literature. We propose four improved control laws aiming at being compatible with the nonlinear model of quadrotors and experimental works. The control laws are run aboard each quadrotor in the flock. By running the control law, each quadrotor interacts with its neighbors to ensure a collision-free flocking.

In the last chapter, we show our simulation and experimental results of our proposed control laws. For the simulation, we use a PC-base simulator of flock of multiple quadrotors which is developed at Heudiasyc laboratory. For experiments, we show the results of the implementation of our control laws on a platform of ArDrone2 quadrotors evolved in an indoor environment of Optitrack motion capture system.

Finally, a general conclusion redraws the essential points of this thesis, presents the lessons learned , the difficulties faced during our work, as well as research prospects.

State of the art

Contents

1.1 Introduction

In this chapter, we review different recent works of the literature about System of Systems, flight formation control and the control of quadrotors. Firstly, we present a brief introduction about systems of systems, their definition and characteristics. Then, we introduce the flight formation control with its different structures that are found in the literature. After that, we present some existing works dealing with flocking. Finally, since our interest is to design an autonomous SoS of multiple UAVs, we introduce the most used modeling methodologies for quadrotors as well as some control approaches that are used to stabilize quadrotors.

1.2 System of Systems

A system is a set of interacting or interdependent component parts forming a complex/intricate whole [Wikipedia,]. When independent systems are networked together to achieve a common goal, we talk then about "System of Systems" (SoS). In the literature, there are several definitions of SoS depending on the context and the field of subsystems, for instance, enterprise, information, transportation, etc.

A global definition of a SoS is given in [Jamshidi, 2008]: *systems of systems are large-scale integrated systems which are heterogeneous and independently operable on their own, but are networked together for a common goal.*

Several characteristics of networked systems render them a SoS, for instance, operational and managerial independence, geographical distribution, etc. Operational independence means that the subsystems are individually functional, while managerial independence means that the subsystems are not managed by a central system [Jamshidi, 2008]. Combining the two previous characteristics leads to a self-organizing SoS.

Air vehicles, airport operations, mobile robotics, military defense systems, swarm robotics, etc., are some examples of System of Systems. All of these systems and their characteristics are discussed in details in [Jamshidi, 2008]. In this work, our interest is to design an autonomous SoS of multiple UAVs.

1.3 Flight Formation Control

Autonomous formation control is a field that addresses the control of multiple robots in order to realize geometrical patterns. Inspired from biology, these geometrical patterns could be regular, such as the V-shape seen in migratory birds, lattice or irregular as seen in a flock of birds. The *Flight Formation Control* field uses and develops techniques of autonomous formation control and applies them on aircraft and multiple aerial robots [Guerrero and Lozano, 2012]. We try in the following subsections to classify different structures and architectures used in autonomous formation control.

1.3.1 Formation control structures

In the existing literature, we can distinguish three formation control structures: *Leader-follower*, *Virtual structure* and *Behavioral-based*. In the following, we describe these structures with their main advantages and drawbacks.

1.3.1.1 Leader-follower structure

In the *leader-follower* structure, individuals in the formation follow one agent (or airplane) which is designated as a leader. A formation flight mission trajectory is loaded in the leader and the followers track their leader. Moreover, different strategies have been implemented in this structure, for instance *leader mode* or *front mode* [Giulietti et al., 2000]. In the *leader mode* strategy, all the followers follow directly the leader. However, in the *front mode* strategy, each agent in the

formation follows its preceding, or neighboring agent and so on until reaching the leader, which drive the whole formation [Hou and Fantoni, 2015].

This structure is simple and widely implemented in multi-agent formation [Guerrero and Lozano, 2012], [Chiaramonti et al., 2006]. Moreover, stability analysis, of systems using this structure, is relatively simple. Experimental work was conducted in [Vasarhelyi et al., 2014] using this control structure. However, this control structure reveals some drawbacks. One of them is that the entire formation depends on one agent, so if there is a problem with the leader, the whole formation will be affected. Moreover, error propagation and non self-organization formation are some disadvantages of this structure.

Solutions of some of these drawbacks are found in [Shi et al., 2006] and [Hou and Fantoni, 2015]. In [Shi et al., 2006], the leader in the formation is replaced by a virtual one. In fact, the virtual leader is a reference trajectory sent to all the agent in the formation. This solution resolves the reliability problem when depending on a physical leader as well as the problem of error propagation. In [Hou and Fantoni, 2015], a multiple leader solution is proposed to solve the problem of one leader dependency. However, the self-organization problem is still persistent in this structure.

1.3.1.2 Virtual structure

In the *virtual structure*, each agent in the formation has its own trajectory to follow. The overall trajectories form the desired formation. Trajectories are calculated in a central computer and sent to agents in the formation. Generally, no interactions between agents are considered. Examples of experimental works of such control strategy can be found in [Kushleyev et al., 2013] and [Schollig et al., 2010].

This structure exhibits some drawbacks. In fact, since there is no interaction between agents in the formation, collisions could not be avoided in the presence of perturbations. Moreover, the centralized control of this structure increases the computational and communication cost.

1.3.1.3 Behavioral-based structure

In the *behavioral-based* structure, each agent follows some rules to achieve the formation. The objective of the formation control is, therefore, broken down into small rules (or behaviors). In fact, this structure is inspired from the collective motion of animals. Among the first technical work on this structure is the distributed behavioral model introduced by *Reynolds* [Reynolds, 1987]. Although Reynolds was specialized in computer graphics, his work inspired researchers in control theory and

robotics in the way of applying Reynolds' rules in a theoretical and experimental framework. Reynolds inspired his rules from biologists' studies of collective motion of animals. He considered that each individual in a formation should follow these rules in order to perform a behavioral-based structure. These rules are: 1) Collision Avoidance; 2) Velocity Matching, and 3) formation Centering.

In the *collision avoidance* rule, each agent in the formation should ensure a predefined security distance with its neighboring agents. This could be done by an embedded controller on each agent. This controller is responsible to generate a repulsion force if the distance with the neighboring agent is less than the security distance.

The *velocity matching* rule is sometimes called *velocity consensus* or *velocity alignment*. In this rule, each agent attempts to match its velocity with nearby agents. This could be ensured by a controller that regulates the relative velocities to zero with respect to the neighboring agents.

In the *formation centering* rule or also called *formation cohesion*, each agent tries to stay close to nearby agents. This is achieved by a controller that generates an attraction force toward the neighboring agents. Moreover, each agent should converge to a global objective of the formation. This objective could be a rendezvous point or a reference trajectory known by all the agents in the formation.

This structure is found in the literature, for example in the work of [Olfati-Saber, 2006], [Tanner et al., 2007] and [Antonelli et al., 2010]. In [Olfati-Saber, 2006], two flocking algorithms are introduced with Lyapunov stability analysis. The author proved analytically that these algorithms embed the rules of Reynolds. In [Tanner et al., 2007], a nonsmooth analysis is used to prove the stability of a group of agents applying the rules of Reynolds. [Antonelli et al., 2010] introduce a Null-Space-based behavioral control laws to follow the rules of Reynolds.

The main advantages of the behavioral-based structure are the self-organization aspect of agents in the formation, scalability and their distributed control. A behavioral-based structure is easily self organized since each agent has to follow a set of rules and knows the objective trajectory or destination. The scalability feature is seen when increasing or decreasing the number of agents in the formation. Since each agent interacts only with neighbors, its computational capabilities are not affected. The interaction with neighbors shows also the distributed control aspect.

One of the main disadvantages of this structure is that we cannot ensure a fixed geometrical pattern of the formation. Patterns such as V-shape, rectangles could not be guaranteed. Only lattice patterns with fixed inter-agent distances could be formed. Moreover, stability analysis, of systems using this structure, is relatively difficult.

1.3.2 Formation control architectures

In the formation control, we can distinguish three types of architectures: centralized, decentralized and distributed. In the following, we try to summarize these three architectures and show their main advantages and drawbacks.

1.3.2.1 Centralized control architecture

In this architecture, agents in the formation are controlled by one centralized controller. This controller senses the states of all the agents, calculates the appropriate control inputs to each of them, and send the controlling signals to agents. Figure 1.1 illustrates the centralized control architecture. The location of this controller could be in a ground base station or aboard an agent in the formation, for instance, the leader agent.

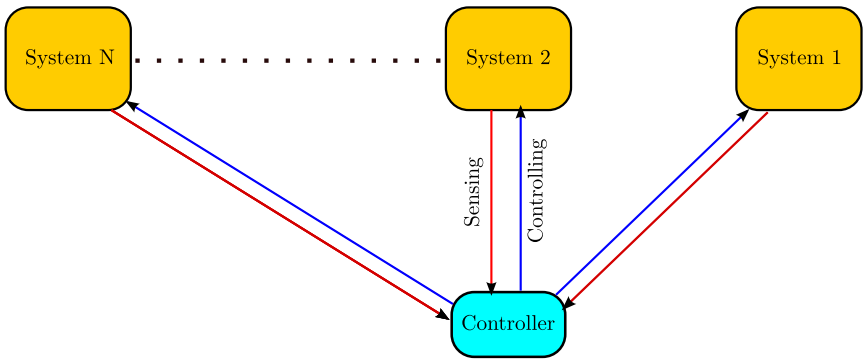


Figure 1.1 – Centralized control architecture of multiple systems

In the literature, we can find several works such as [Bellingham et al., 2002] and [Richards and How, 2002]. This kind of architecture is clearly identified in the aforementioned virtual structure. Moreover, a centralized control architecture could be implemented in the *leader-follower* structure, especially in the *leader mode*. [Kushleyev et al., 2013] and [Schollig et al., 2010] show experimental works that used the centralized control architecture.

The main asset of this control architecture is the global knowledge of the state of all agents in the formation. This global knowledge is beneficial in designing a controller to achieve an optimal path planning and self organization. However, this control architecture needs high computational and communication power. Moreover, the scalability problem could be easily identified on this architecture, since the computational cost depends on the number of agents in the formation. In addition, the centralized control renders this architecture unreliable, *i.e.* if there is a problem in the central controller, the whole formation will be affected.

1.3.2.2 Distributed control architecture

The distributed control architecture is the most used and successful architecture for autonomous flight formation control. In this architecture, each agent in the formation has its own aboard controller that uses its own states and informations from neighbors. The informations from neighboring agents could be their positions, velocities, ranges from the agent, etc. The informations from neighbors could be gathered through sensing, using on board sensors such as cameras or LIDAR, or through communication. Figure 1.2 shows the distributed control architecture of multiple systems.

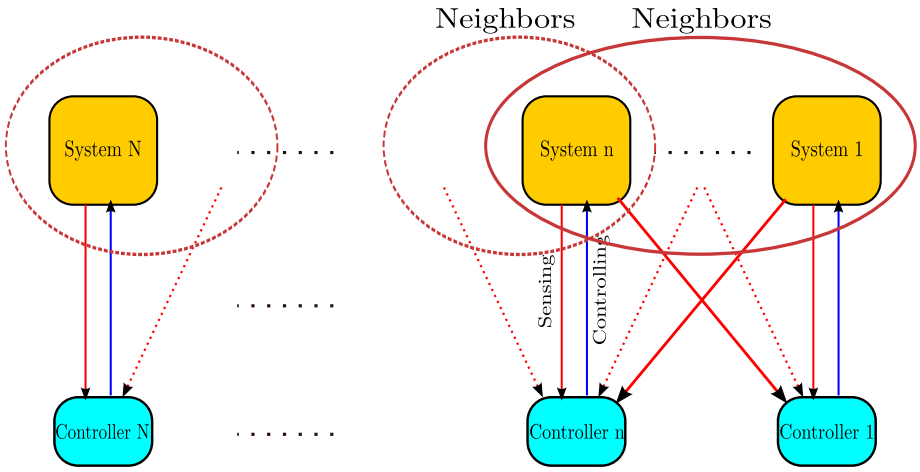


Figure 1.2 – Distributed control architecture of multiple systems

In the literature, we can find several works that used the distributed control architecture in the autonomous formation control. In the works of [Olfati-Saber, 2006] and [Tanner et al., 2007], distributed controllers were designed to control multiple-dynamical agents. In [Hou and Fantoni, 2015], a distributed control architecture is used to ensure a formation of a *leader-follower* structure of multiple UAVs. Several works in autonomous and flight formation control, use the name "decentralized control" to indicate distributed control architecture, for instance, [Zhang, 2006], [Roberson and Stilwell, 2006] and [Vasarhelyi et al., 2014].

The distributed control architecture has several advantages. Since the interaction between agents in the formation is limited only to neighbors, this architecture is the most scalable one. The reliability is also one of the main advantages of this architecture. However, the optimality of achieving an optimal path planning and self organization of the formation are deteriorated. This drawback is caused by the fact that the controller on each agent only uses informations from neighbors.

1.3.2.3 Decentralized control architecture

This control architecture is mainly used in interconnected subsystems [Bakule, 2008]. The interconnections between subsystems are principally mechanical, for instance, springs. Each system will have its own controller. The controller only measures and controls the states of the system where it is implemented [Jovanovic, 2004]. In other word, the only knowledge a controller in a subsystem has is the state of the subsystem itself. Figure 1.3 illustrates the decentralized control architecture.

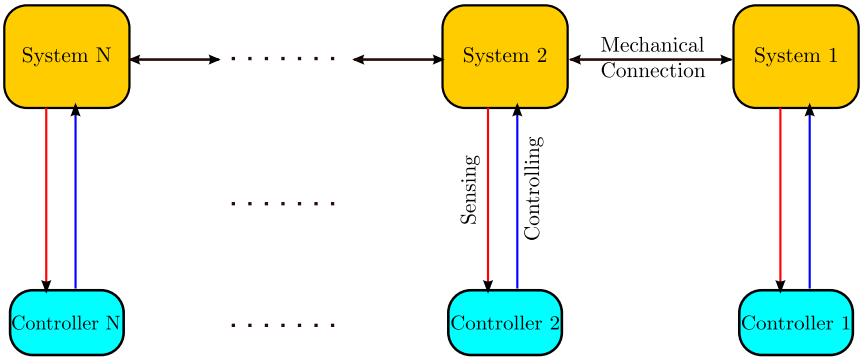


Figure 1.3 – Decentralized control architecture of multiple systems

Since the controller in the decentralized control architecture has no knowledge about the states of the other subsystems, this architecture could not be applied for autonomous flight formation control. However, the wide use of name "*Decentralized control*" in some formation control literature, for instance the works of [Zhang, 2006], [Roberson and Stilwell, 2006] and [Vasarhelyi et al., 2014], could be conflicted with the *distributed architecture* as mentioned before. In fact, the authors claim the development of decentralized controllers, while in their articles they mentioned that the controller on each agent uses informations from neighbors.

This conflict could be explained by the fact that multiple-robot systems are studied by researchers from different specialties, for example, information technology, sensor network, robotics and control theory, etc. In this work, we choose the "distributed architecture" as a name of our control strategies.

1.4 Flocking in literature

In biology, flocking is the collective, harmonic and collision-free motion of birds toward a common objective. In control theory and autonomous formation control, it could be defined as the convergence of the pose and velocity of multiple agents to a common objective. The objective could be a rendezvous point or a trajectory. In the steady state, the agents achieve a velocity consensus, and their positions form

a cohesive lattice around the objective, [Dimarogonas and Kyriakopoulos, 2006], [Olfati-Saber, 2006].

Flocking could be in fact characterized as a *behavioral-based* control structure. In addition to achieving the rules of Reynolds, authors in the literature include graph theory tools to model the multiple-agent systems. Moreover, stability analysis is also included in the design of algorithms and control laws that ensure the flocking. In the literature we can find several works that deal with the flocking problem, for instance, [Tanner et al., 2005], [Olfati-Saber, 2006], [Dimarogonas and Kyriakopoulos, 2006], [Tanner et al., 2007] and [Valbuena Reyes and Tanner, 2015].

In this thesis, we try to develop and apply flocking algorithms on multiple quadrotors in real-time experiments. In the following, we summarize some preliminaries of graph theory that is used to model and analyze flocking of multiple agents.

Preliminaries in graph theory

Graph theory is used to describe the topology of a multi-quadrotor system. A multi-quadrotor system is represented by an undirected graph $G = (\mathcal{V}, E)$, where $\mathcal{V}$ is a set of nodes $\mathcal{V} = \{1, 2, ..., M\}$, and E is a set of edges $E \subseteq \{(i, j) : i, j \in \mathcal{V}, i \neq j\}$. Every node represents a quadrotor and edges depict the sensing between quadrotors.

An adjacency matrix $\mathcal{A}$ is an $M \times M$ matrix with elements $a_{ij} = 1$ if $(i, j) \in E$ and $a_{i,j} = 0$ otherwise. Connected quadrotors with a quadrotor i in the graph can be modeled by a set $\gamma_i = \{j \in \mathcal{V} : a_{i,j} \neq 0\}$. We assume that every quadrotor has an omni-directional detection capability, *i.e.* it can detect in all directions. This capability of each quadrotor means that there is a mutual detection between connected quadrotors. The adjacency matrix is then symmetric $\mathcal{A}^T = \mathcal{A}$. Therefore, our graph is undirected. For more information about graph theory, the reader can refer to [Diestel, 2005].

Figure 1.4 shows a multi-quadrotor system of four quadrotors represented as an undirected graph.

Before working on the dynamics of quadrotors, we need to represent our multi-quadrotor system in the Euclidean space. Therefore, to every node i in the graph, a position vector $q_i \in \mathbb{R}^f$ is associated, where f is the dimension of the space (example: $f = 2, 3$). The configuration of all nodes of the graph is defined by the vector $q = col(q_1, ..., q_n) \in \mathbb{R}^{fM}$.

A set of spacial neighbors of a quadrotor i is defined by:

$$N_i = \{j \in \mathcal{V} : \|q_j - q_i\| < c\} \tag{1.1}$$

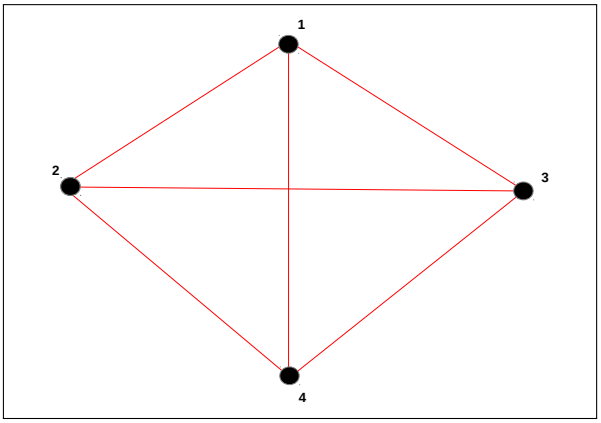


Figure 1.4 – Four quadrotors in an undirected graph

where $\|.\|$ is the Euclidean norm, and c is the interaction range. A position-induced graph $G(q) = (\mathcal{V}, E(q))$ is called a proximity net and is defined by $\mathcal{V}$ and the set of edges $E(q) = \{(i,j) \in \mathcal{V} \times \mathcal{V} : \|q_j - q_i\| < c, j \neq i\}$.

The desired conformation of multiple quadrotors in a flock could be written as follows:

$$\|q_j - q_i\| = d \quad \forall j \in N_i(q) \tag{1.2}$$

where d is the desired inter-distance. A proximity net that ensures the objective in (1.2) is defined as an "α-*Lattice*". However, implicit inaccuracies give rise to an α-*Lattice* with some edge-length uncertainty. This type of proximity net is called a "*quasi-*α*-Lattice*" [Olfati-Saber, 2006], and it is described by the following inequality:

$$-\delta < \|q_j - q_i\| - d < \delta \quad \forall (i,j) \in E(q) \tag{1.3}$$

where δ is the edge-length uncertainty.

1.5 Quadrotor Modeling and Control

Quadrotors are aerial robots that have been extensively used in the last decades. Together, in research and industrial fields, quadrotors seem to be a promising platform. They are type of Unmanned Aerial Vehicles (UAV) that have useful properties comparing with fixed-wing airplanes and helicopters.

A quadrotor is a flying robot with four rotors, controlled by a CPU (Control Processing Unit), which uses sensors, such as Inertial Measurement Unit and Ultra Sound Range Finder to measure its altitude and orientations. GPS (Global Positioning System), motion capture systems (ex: Vicon or Optitrack) or vision sensors, are used to precisely estimate poses and velocities. The CPU then uses these information to control the poses and the velocities through the actuation of the four rotors.

The simple structure and the ease of use of quadrotors render them an ideal platform for inspection and surveillance missions. They are recently used in surveillance of manifestations and sport events [ScienceEtVie, 2014]. Moreover, their ability of Vertical TakeOff and Landing (VTOL) makes them efficient in the inspection of monuments, buildings, large-scale infrastructures, etc [ScienceEtVie, 2014].

1.5.1 Quadrotor modeling

A quadrotor is modeled as a rigid body that evolves in 3D space. Several studies dealt with the modeling of quadrotors as in [Lozano, 2010], [Mahony et al., 2012], [Fresk and Nikolakopoulos, 2013] and [Bouabdallah and Siegwart, 2007]. A quadrotor is composed of four rotors and motors located at corners of a square. We begin by defining the frames of reference. Let $I = (I_x, I_y, I_z)$ be the global inertial frame, and let $B = (B_x, B_y, B_z)$ be the body-fixed frame, see Figure 1.5.

The nonlinear model of a quadrotor is given as follows:

$$\dot{\xi} = \nu \tag{1.4a}$$

$$m\ddot{\xi} = G + \mathbf{R}U \tag{1.4b}$$

$$\dot{\eta} = W\Omega \tag{1.4c}$$

$$J\dot{\Omega} = -\Omega \times J\Omega + \tau \tag{1.4d}$$

where $\xi = [x, y, z]^T$ is the position of the center of mass of the quadrotor in the I frame, $\nu = [\nu_x, \nu_y, \nu_z]$ is the vector of linear velocities in the I frame, $\eta = [\phi, \theta, \psi]^T$ is the vector of Euler angles named as *Roll*, *Pitch* and *Yaw* respectively. $\Omega = [w_{B_x}, w_{B_y}, w_{B_z}]^T$ is the vector of the angular velocities in the B frame, m is the

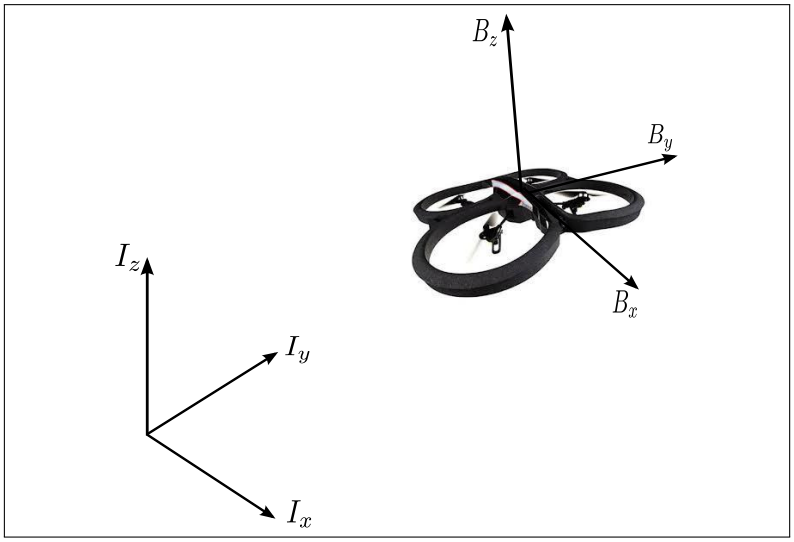


Figure 1.5 – Representation of a quadrotor with the global inertial and the body-fixed frames

mass of the quadrotor, $G = [0, 0, -g]^T$ with g is the gravitational acceleration, $U = [0, 0, F]^T$ is the thrust vector and $\tau = [\tau_\phi, \tau_\theta, \tau_\psi]^T$ is the torque vector.

J is the moment of inertia matrix given as follows:

$$J = \begin{pmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{pmatrix} \tag{1.5}$$

The J matrix is diagonal because we suppose that the quadrotor structure is symmetric. This symmetry implies that $J_x = J_y$. W is a transformation matrix between the angular velocities and the derivatives of Euler angles, and is given by the following expression:

$$W = \begin{pmatrix} 1 & s_\phi t_\theta & c_\phi t_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & \frac{s_\phi}{c_\theta} & \frac{c_\phi}{c_\theta} \end{pmatrix} \tag{1.6}$$

where c, s, and t refer to cos, sin, and tan functions respectively.

$\mathbf{R}$ is the rotation matrix from the B frame to the I frame, which is given as follows:

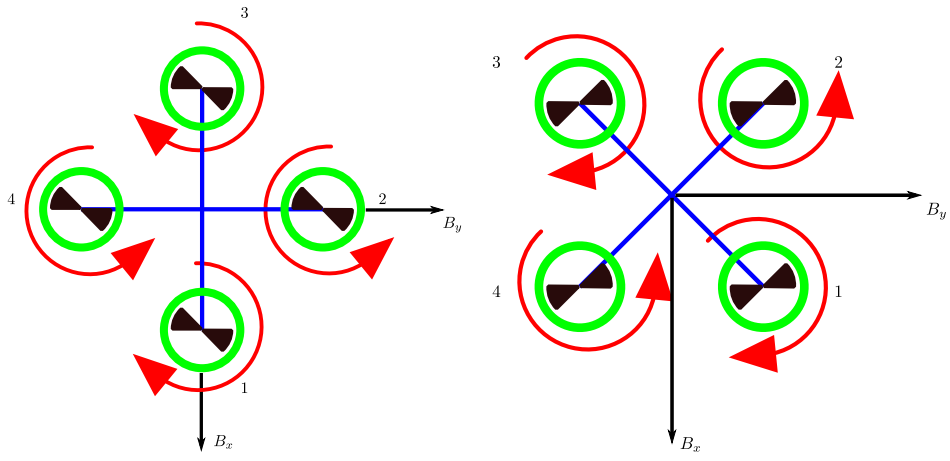
$$\mathbf{R} = \begin{pmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{pmatrix} \tag{1.7}$$

The detailed nonlinear model of a quadrotor is then given as follows:

$$\dot{x} = \nu_x \tag{1.8a}$$
$$\dot{y} = \nu_y \tag{1.8b}$$
$$\dot{z} = \nu_z \tag{1.8c}$$
$$\dot{v}_x = (1/m)(c_\phi s_\theta c_\psi + s_\phi s_\psi)F \tag{1.8d}$$
$$\dot{v}_y = (1/m)(c_\phi s_\theta s_\psi - s_\phi c_\psi)F \tag{1.8e}$$
$$\dot{v}_z = -g + c_\phi c_\theta F \tag{1.8f}$$
$$\dot{\phi} = w_{B_x} + s_\phi t_\theta w_{B_y} + c_\phi t_\theta w_{B_z} \tag{1.8g}$$
$$\dot{\theta} = c_\phi w_{B_y} - s_\phi w_{B_z} \tag{1.8h}$$
$$\dot{\psi} = \frac{s_\phi}{c_\theta} w_{B_y} + \frac{c_\phi}{c_\theta} w_{B_z} \tag{1.8i}$$
$$\dot{w}_{B_x} = (\frac{J_y - J_z}{J_x}) w_{B_y} w_{B_z} + (\frac{1}{J_x})\tau_\phi \tag{1.8j}$$
$$\dot{w}_{B_y} = (\frac{J_z - J_x}{J_y}) w_{B_x} w_{B_z} + (\frac{1}{J_y})\tau_\theta \tag{1.8k}$$
$$\dot{w}_{B_z} = (\frac{1}{J_z})\tau_\psi \tag{1.8l}$$

1.5.1.1 Quadrotor configurations

A quadrotor could be constructed in two configurations. The first one is the *Plus-configuration* and the second is the *X-configuration*. In the *Plus-configuration*, see figure 1.6*a*, the arms of the quadrotor are aligned with the B_x and the B_y axis of the body-fixed frame. However, in the *X-configuration*, see figure 1.6*b*, the quadrotor has the *Plus-configuration* rotated by 45° about the B_z axis.

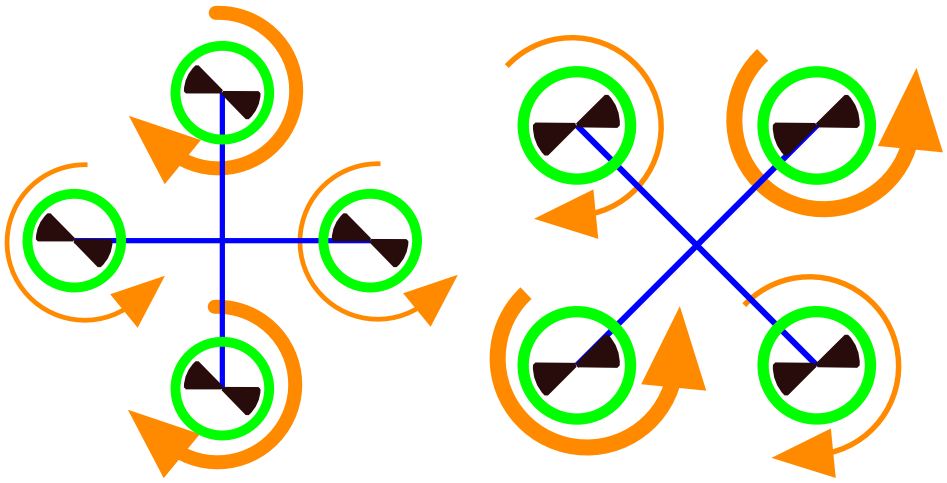


(a) Plus configuration (b) X configuration

Figure 1.6 – *Plus* and *X configurations* of quadrotor in hovering mode. The rotors have the same angular velocity. B_x axis is the frontal direction

In both configurations, two rotors rotate clockwise and the others rotates counter clockwise to compensate yaw torque. Moreover, the front of the quadrotor is on the B_x axis.

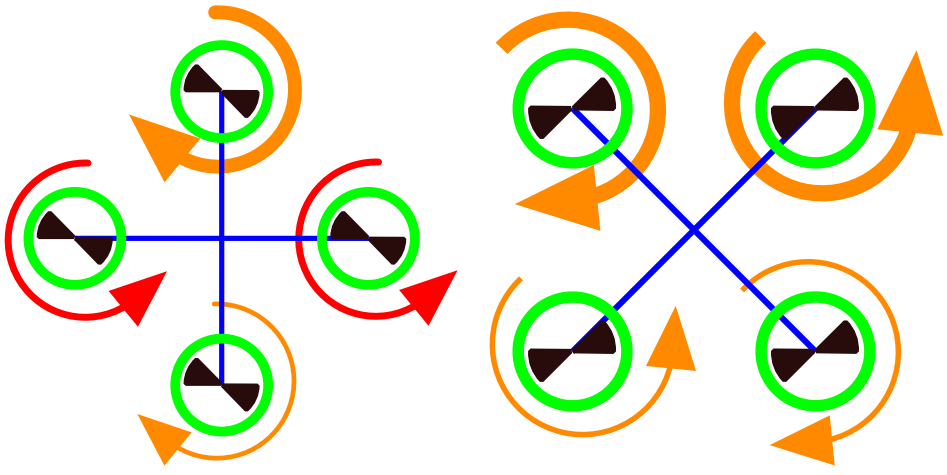
To control the altitude z, also known as hovering mode, the four rotors are actuated by the same angular velocities, as illustrated in figure 1.6. Figure 1.7 shows the control of the yaw angle in both configurations. We decrease the angular velocities of the rotors that turn in the same direction and we increase the others.



(a) Turning counter clockwise (b) Turning clockwise

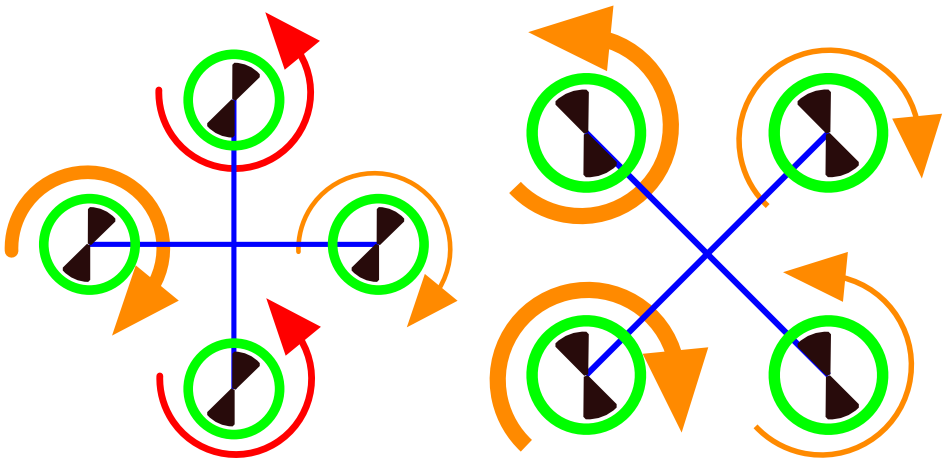
Figure 1.7 – Controlling the yaw angle in the *Plus* and *X configurations*. The width of the arrows is proportional to the rotors angular velocity.

To tilt the pitch angle, we increase the angular velocity of the rear rotor (or rotors in the *X-configuration*) and we decrease the one of the front rotor, see figure 1.8. The same principle is applied to tilt the roll angle, see figure 1.9.



(a) (b)

Figure 1.8 – Controlling the pitch angle in the *Plus* and *X configurations*



(a) (b)

Figure 1.9 – Controlling the roll angle in the *Plus* and *X configurations*

We control the x and y translations through the control of the pitch and yaw angles. A fixed tilt in the pitch angle will lead the quadrotor to move in the x direction. The same principle is used to control the y displacement through roll tilting.

1.5.1.2 Torques and thrust

According to the type of configuration, we can define a relation between the angular velocities of rotors in one hand and the torques and the thrust in the other hand. We recall the relation between the angular velocity ω_i of one rotor and the generated vertical force or thrust F_i on the rotor plan:

$$F_i = K_t \omega_i^2 \tag{1.9}$$

where K_t is a constant that contains aerodynamic parameters of the blade. Moreover, a drag moment is induced on each rotor due to aerodynamic forces. The direction of the drag moment is opposite to the rotation of the rotor. This moment is given by the following formula:

$$\tau_i = K_g \omega_i^2 \tag{1.10}$$

The aerodynamic parameters relating the angular velocities and the drag moment are included in the constant K_g. Figure 1.10 illustrates the different forces and drag moments on a quadrotor.

The overall thrust generated by the four rotors is given as follows:

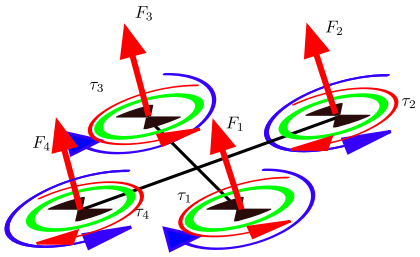


Figure 1.10 – Forces and drag moments on a quadrotor

$$F = \sum_{i=1}^{4} F_i = K_t \sum_{i=1}^{4} \omega_i^2 \tag{1.11}$$

The subscript i indicates the rotor number as shown in figure 1.6. The relation (1.11) is valid for the X and *Plus configurations*.

In the *Plus configuration*, the relation between control torques, forces and drag moments could be given as follows:

$$\tau = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} l(F_2 - F_4) \\ l(F_3 - F_1) \\ (\tau_1 + \tau_3) - (\tau_2 + \tau_4) \end{bmatrix} \tag{1.12}$$

In the X *configuration*, however, the relations in (1.12) will be modified as follows:

$$\tau = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} l\frac{\sqrt{2}}{2}((F_1 + F_2) - (F_3 + F_4)) \\ l\frac{\sqrt{2}}{2}((F_2 + F_3) - (F_1 + F_4)) \\ (\tau_1 + \tau_3) - (\tau_2 + \tau_4) \end{bmatrix} \tag{1.13}$$

Equations (1.11), (1.12) and (1.13) could be written in a matrix form as follows:

- *Plus configuration*

$$\begin{bmatrix} F \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} K_t & K_t & K_t & K_t \\ 0 & lK_t & 0 & -lK_t \\ -lK_t & 0 & lK_t & 0 \\ K_g & -K_g & K_g & -K_g \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{1.14}$$

- *X configuration*

$$\begin{bmatrix} F \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} K_t & K_t & K_t & K_t \\ \frac{\sqrt{2}}{2}lK_t & \frac{\sqrt{2}}{2}lK_t & -\frac{\sqrt{2}}{2}lK_t & -\frac{\sqrt{2}}{2}lK_t \\ -\frac{\sqrt{2}}{2}lK_t & \frac{\sqrt{2}}{2}lK_t & \frac{\sqrt{2}}{2}lK_t & -\frac{\sqrt{2}}{2}lK_t \\ K_g & -K_g & K_g & -K_g \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{1.15}$$

1.5.2 Quadrotor control

Quadrotors received a great attention from researchers in control theory and robotics all over the world. As a cheap and easy assembled platform, a quadrotor represents an ideal platform to test, validate and improve different control laws and algorithms. Hence, it is difficult to find and limit an exhaustive list of control techniques and algorithms used for quadrotors.

In this part, we will introduce the General nested control loop of quadrotor system. Then we will classify the existing works on controlling quadrotors.

1.5.2.1 General nested control loop of quadrotor

A quadrotor is an under-actuated system, *i.e.*, it possesses less control actuators than controlled degrees of freedom. In a quadrotor, we have four rotors (four actuators) and six degrees of freedom (three translations x, y, z and three rotations roll ϕ, pitch θ and yaw ψ). To control such system, we need three nested-control-closed loops, see figure (1.11).

The first is the outer loop. It is used to control the translational states x, y, z, $\dot{x}$, $\dot{y}$, $\dot{z}$, where their dynamics are represented by equations (1.4a) and (1.4b). The inputs of this control loop is a desired translational trajectory r_d and its time derivatives. The output of this loop is the thrust F, and the desired roll (ϕ_d) and pitch (θ_d) angles.

The second loop is responsible of the control of the rotational states ϕ, θ, ψ, $\dot{\phi}$, $\dot{\theta}$, $\dot{\psi}$. The dynamics of these states are modeled in equations (1.4c) and (1.4d). The inputs of this control is the desired (ϕ_d) and (θ_d) angles as well as the desired yaw angle ψ_d and its time derivative. The outputs of this control are the roll, pitch and yaw torques τ_ϕ, τ_θ, τ_ψ.

We can use any of the control techniques mentioned in the subsection (1.5.2.2) to control the translational and the rotational dynamics. The outputs F, τ_ϕ, τ_θ, τ_ψ will be used to generate the desired rotors angular velocities, ω_{1d}, ω_{2d}, ω_{3d}, ω_{4d}, by using matrix equations (1.14) or (1.15) depending on the quadrotor configuration. For example, in the *X-configuration* we get the squares of the desired angular velocities as follows:

$$\begin{bmatrix} \omega_{1d}^2 \\ \omega_{2d}^2 \\ \omega_{3d}^2 \\ \omega_{4d}^2 \end{bmatrix} = \begin{bmatrix} K_t & K_t & K_t & K_t \\ \frac{\sqrt{2}}{2}lK_t & \frac{\sqrt{2}}{2}lK_t & -\frac{\sqrt{2}}{2}lK_t & -\frac{\sqrt{2}}{2}lK_t \\ -\frac{\sqrt{2}}{2}lK_t & \frac{\sqrt{2}}{2}lK_t & \frac{\sqrt{2}}{2}lK_t & -\frac{\sqrt{2}}{2}lK_t \\ K_g & -K_g & K_g & -K_g \end{bmatrix}^{-1} \begin{bmatrix} F \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} \tag{1.16}$$

In the internal loop, we control the rotors, which are generally blades mounted on DC brushless motors. The idea is to have an angular velocity tracking controller on each rotor, often called drivers. This controller ensures the tracking of the desired angular velocities in (1.16).

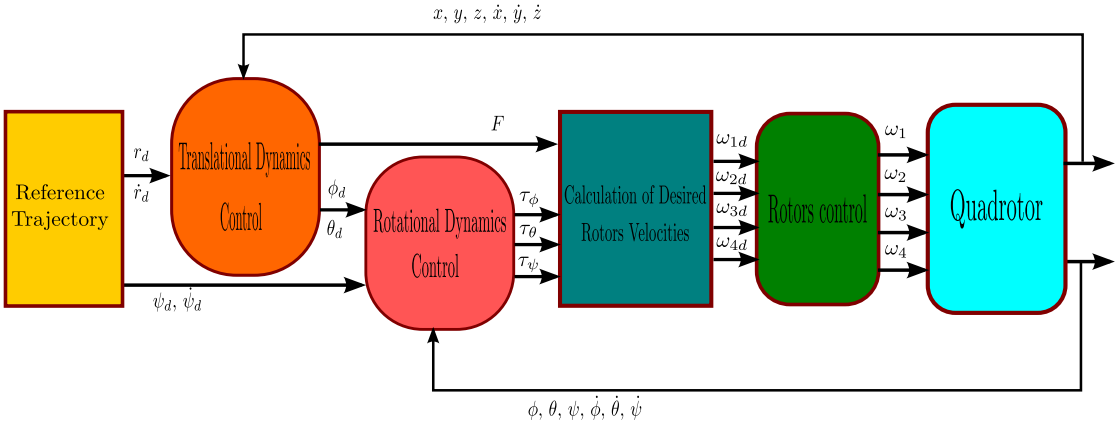


Figure 1.11 – Global structure of quadrotor control

1.5.2.2 Classification of quadrotor existing control laws

In this section, a classification of control laws applied on quadrotor systems is introduced. It is important to note that in the last decade, the literature of quadrotor control has been exploded of articles. In fact, since quadrotors are low-cost experimentation, they have become the most appropriate platform in navigation and control strategies. Therefore, it is difficult to give an exhaustive list of existing controlling methods in the literature. However, we try, in the following, to give a variety of the most well-known works on the control of quadrotors systems, to the best of our knowledge.

We can classify the control laws of quadrotors in two major types, *linear* and *nonlinear*. For linear control, the model in (1.4) is firstly linearized. Then a linear control technique is used, such as, PID, Linear Quadratic, or H_∞, etc. For nonlinear control, techniques such as, backstepping, sliding mode, nested saturation, or Lyapunov redesign are used. These techniques could be associated, or not, with a feedback or feedforward linearization [Formentin and Lovera, 2011]. Theoretical principles of these techniques could be found in [Khalil, 2002] and [Hespanha, 2009].

In linear control techniques, we can find works of [Bouabdallah et al., 2004b] and [Bouabdallah et al., 2004a] where PID and LQ controllers are proposed, as well as linear control based on Lyapunov functions. The authors validated their results by simulations and experimental works on a quadrotor platform. Moreover, PD control is used in [Erginer and Altug, 2007]. They used simulations to validate their

proposed control law. In [Bouffard et al., 2012], a learning-based model predictive control is proposed to control a quadrotor in real-time.

In nonlinear techniques, [Castillo et al., 2004] and [Kendoul et al., 2007] applied nested saturation controllers to stabilize the attitude of a quadrotor and they validated their work on a real-time platform. Moreover, the nested saturation control with a quaternion-based feedback was used in [Guerrero-Castellanos et al., 2011] to stabilize the attitude of a real quadrotor. Backstepping and sliding-mode techniques are used in [Bouabdallah and Siegwart, 2005] and [Madani and Benallegue, 2006] with simulation and experimental validations. In addition, [Tayebi and McGilvray, 2006] proposed and tested experimentally a control scheme for attitude stabilization of quadrotor based on quaternions. In [Lee et al., 2010], a geometric tracking technique is proposed to control a quadrotor on the special Euclidean group SE(3). The work in [Lee et al., 2010] has been improved in [Lee, 2013] with a more robust and adaptive controller and experimental tests on a real quadrotor.

Recently, a bio-inspired biomimetic-based output feedback control was introduced in [Guerrero Castellanos et al., 2015], in order to stabilize the attitude of a quadrotor. This control was validated by theoretical and experimental works. An event-triggered nonlinear control to stabilize the attitude of quadrotor was introduced in [Guerrero Castellanos et al., 2014]. It was proved experimentally that this control law reduce by 80% the communications of the embedded system without deteriorating the performance of the whole system. A feedback linearization approach to design a quadrotor controller to perform a trajectory following was introduced in [Ghandour et al., 2014]. The proposed approach is used to tolerate a rotor failure, so the quadrotor enter a stable spin around its vertical axis.

1.6 Conclusion

In this chapter, we tried to summarize a state of the art on System of Systems, on flight formation control strategies and on quadrotors modeling and control.

In the beginning, we gave a brief introduction on the concept of System of Systems. Then, formation control structures were also introduced in this chapter. We mainly focused on the most used control structures in flight formation control. Moreover, we summarized advantages and drawbacks of these control structures. In addition, formation control architectures were recalled. We tried to clarify ambiguities of some of them and to show their main advantages and disadvantages.

After that, We summarized the definition of the flocking behavior in the literature. Preliminaries on graph theory tools used in flocking algorithms in the literature were introduced. These preliminaries will be used in the following chapters

to introduce our control algorithms of multiple quadrotor system of systems.

Finally, We introduced the dynamical model of quadrotor that will be used in the following chapters of this thesis. In the control of quadrotors, we tried to mention the existing control laws in the literature. Moreover, we introduced the general nested control loops that is usually used in quadrotors.

Flocking by trajectory generation

Contents

2.1 Introduction

In this chapter, we use the behavior-based control structure to achieve a multiple-UAV flocking. We conceive a behavior intending to address the control design towards a successful achievement of the flocking task without fragmentation. The proposed behavior treats the flocking problem from a global perspective, that is, we include a tendency of separated UAVs to form a flock.

We propose a new control strategy to achieve the rules of Reynolds and the new proposed behavior. This strategy is based on the LQR control. Moreover, this strategy allows us to design an LQR controller, for each UAV, which is independent of the number of UAVs within a flock. In fact, we consider the flocking as a problem of trajectory (or reference) generation, rather than an issue of control design. The desired trajectory for each UAV is generated by using the measured states of the UAVs in its field of view. Furthermore, we expand our control strategy to perform a navigation behavior.

In this chapter, we suppose that UAVs have simple linear dynamics (double integrator), in a similar manner as the work in [Olfati-Saber, 2006]. Nonlinear models will be treated in an upcoming chapter.

2.2 Simplified dynamics of multiple UAVs

In this chapter, the proposed dynamical model of UAV is a double integrator. Furthermore, we assume that the body of a UAV is a particle in the Euclidean framework. Equation (2.1) is a state-space representation model of a UAV i in a flock.

$$\begin{cases} \dot{\mathbf{x}}_i &= A_i\,\mathbf{x}_i + B_i\,\mathbf{u}_i \\ \mathbf{y}_i &= C_i\,\mathbf{x}_i \end{cases} \tag{2.1}$$

where $\mathbf{x}_i = [q_i \;\; \dot{q}_i]^T$, $\mathbf{x}_i \in \mathbb{R}^l$ is the state vector with $q_i \in \mathbb{R}^f$ and f is the dimension of the space (example: $f = 2, 3$), $\mathbf{u}_i \in \mathbb{R}^f$ is the control input and $\mathbf{y}_i \in \mathbb{R}^l$ is the system output, with $(l = 2f)$. In this paper, we consider $\mathbf{y}_i = \mathbf{x}_i$. The state-space matrices are:

$$A_i = \begin{pmatrix} \mathbf{0}_f & \mathbf{I}_f \\ \mathbf{0}_f & \mathbf{0}_f \end{pmatrix} \quad B_i = \begin{pmatrix} \mathbf{0}_f \\ \mathbf{I}_f \end{pmatrix}$$

$$C_i = \mathbf{I}_l$$

where, $\mathbf{I}_f$ and $\mathbf{I}_l$ are $f \times f$ and $l \times l$ identity matrices and $\mathbf{0}_f$ is an $f \times f$ zero matrix. This system is, therefore, controllable and observable.

Now, we can formulate the state-space model of multiple UAVs. Let us consider, as in [Olfati-Saber, 2006], $q = col\,(q_1, ..., q_M) \in \mathbb{R}^{f \times M}$ the position configuration of all nodes in the multiple UAVs graph, where M is the number of UAVs. Moreover, all the states, the outputs, and the control inputs of the multiple UAVs system are written as $\mathbf{x} = col\,(\mathbf{x}_1, ..., \mathbf{x}_M) \in \mathbb{R}^{l \times M}$, $\mathbf{y} = col\,(\mathbf{y}_1, ..., \mathbf{y}_M) \in \mathbb{R}^{l \times M}$ and $\mathbf{u} = col\,(\mathbf{u}_1, ..., \mathbf{u}_M) \in \mathbb{R}^{f \times M}$ respectively. Thus, we can write the multiple UAVs state-space system as follows:

$$\begin{cases} \dot{\mathbf{x}} &= \mathbf{A}\,\mathbf{x} + \mathbf{B}\,\mathbf{u} \\ \mathbf{y} &= \mathbf{C}\,\mathbf{x} \end{cases} \tag{2.2}$$

where, $\mathbf{A} = \mathbf{I}_M \otimes A_i$, $\mathbf{B} = \mathbf{I}_M \otimes B_i$, and $\mathbf{C} = \mathbf{I}_M \otimes C_i$, with $\otimes$ being the Kronecker product.

We define $\mathbf{x}_{\eta_i}^j = [q_j \;\; \dot{q}_j]^T$, $\in \mathbb{R}^l$, with $j \in \gamma_i$, as the state of a neighboring UAV j measured by a UAV i. Then, the measured states of all the neighbors could be written as the vector $\mathbf{x}_{\eta_i} = col\left(\mathbf{x}_{\eta_i}^1, ..., \mathbf{x}_{\eta_i}^q\right) \in \mathbb{R}^{l \times N}$ where $N = |\eta_i|$ is the number of

neighbors.

2.3 Linear quadratic control

A linear quadratic control is a class of modern control schemes known as *optimal control methods*. It aims to find a linear control law that stabilizes a linear plant in the best possible way compared to systems of particular types [Anderson and Moore, 1990]. One of the main advantages of linear quadratic control, compared with some classical schemes such as pole placement, is that it gives a systematic method to calculate the state feedback control gain [Ogata, 2010]. The linear quadratic control could be employed on several control problems, such as, time-variant, time-invariant, infinite and finite-time regulation and trajectory following. In this work, we consider the simple infinite time-invariant regulation and trajectory following problem. Although this control problem is considered as *suboptimal*, [Anderson and Moore, 1990] and [Lewis and Syrmos, 2012], we still can benefit from the systematic methodology of linear quadratic control for designing the control law. More details about the other problems could be found in [Anderson and Moore, 1990].

2.3.1 Regulator problem

In the infinite-time-invariant regulator problem, the LQR (Linear Quadratic Regulator) objective for a dynamical system such as (2.1) is to find the control input $\mathbf{u}_i(t)$ that drives the nonzero initial state vector $\mathbf{x}_i(0)$ to zero, while minimizing the following quadratic cost function:

$$J_{LQR} = \int_0^\infty \mathbf{x}_i^T(t)\ Q\ \mathbf{x}_i(t) + \mathbf{u}_i^T(t)\ R\ \mathbf{u}_i(t)\ \mathrm{d}t \tag{2.3}$$

where $\mathbf{x}_i$ is the controlled state vector, Q and R are symmetric positive-definite matrices of dimensions $l \times l$ and $f \times f$ respectively. The quadratic cost function, also named *quadratic performance index*, is used as a measure of the performance specifications of the control input bounds. In fact, the two matrices Q and R are weighting matrices used to design the control law in order to respect the performance specifications of the controlled system.

In one hand, the matrix R is used to constrain the control inputs. If the elements of this matrix are small, we get a large control input and vice versa. In the other hand, the matrix Q is used to constrain the output signal. If the elements of Q are small, the output signal response will be fast, and vice versa. It is up to the

user to choose the values of the matrices Q and R depending on the performance specifications [Hespanha, 2009].

In this chapter, we suppose that the state vector $\mathbf{x}_i$ is available or completely measurable, that is, $\mathbf{y}_i = \mathbf{x}_i$. If it is not the case, a state estimator could be used to retrieve the state vector from the available output vector.

The control input $\mathbf{u}_i$, which minimizes J_{LQR} and which stabilizes the controlled output to zero, is then given by:

$$\mathbf{u}_i = -K\mathbf{x}_i \tag{2.4}$$

where K is the LQR gain matrix:

$$K = R^{-1}B_i^T P \tag{2.5}$$

with P is a symmetric positive-definite matrix and the solution of the following Algebraic Riccati Equation [Hespanha, 2009].

$$A_i^T P + PA_i + Q - PB_i R^{-1} B_i^T P = 0 \tag{2.6}$$

2.3.2 Trajectory following problem

In the trajectory following problem, the output of the system $\mathbf{y}_i$ needs to track or to follow a desired trajectory $r(t)$ in some optimal sense. Moreover, the trajectory should be bounded and feasible , *i.e.*, the trajectory must be an equilibrium solution of the closed-loop system, [Anderson and Moore, 1990] [Murray, 2010]. In this case, the cost function is written as follows:

$$J_{LQR} = \int_0^\infty (C_i\mathbf{x}_i - r)^T \; Q \; (C_i\mathbf{x}_i - r) + \mathbf{u}_i^T(t) \; R \; \mathbf{u}_i(t) \; \mathrm{d}t \tag{2.7}$$

The control input is then written as the sum on two terms: state feedback control and a feedforward control as the following:

$$\mathbf{u}_i = -K\mathbf{x}_i + R^{-1}B_i^T\mathbf{u}_d \tag{2.8}$$

with K is given as in equation (2.5) and u_d is the output of the following system:

$$\dot{\mathbf{u}}_d = -(A_i - B_iK)^T\mathbf{u}_d - C_i^T Q\, r(t) \tag{2.9}$$

The equation (2.9) could be approximated by considering $\dot{\mathbf{u}}_d = 0$. Therefore, the expression of $\mathbf{u}_d$ will be written as follows:

$$\mathbf{u}_d = -[(A_i - B_i K)^T]^{-1} C_i^T Q\, r(t) \tag{2.10}$$

Therefore, the tracking control law in (2.8) will be written as follows:

$$\mathbf{u}_i = -K\mathbf{x}_i + K_r\ r(t) \tag{2.11}$$

with $K_r = -R^{-1}B_i^T[(A_i - B_i K)^T]^{-1}C_i^T Q$.

In this case, the trajectory $r(t)$ should be slowly varying [Anderson and Moore, 1990]. The development of the previous equations and more details about the trajectory following problem in a linear quadratic optimal control could be found in [Anderson and Moore, 1990] and [Lewis and Syrmos, 2012].

2.4 Aggregation behavior

Several studies on flocking and schooling behavior discuss a *behavioral tendency* in the individuals [Partridge, 1982] [Couzin, 2009]. In [Partridge, 1982], a crucial question was: *What makes fish gather in a school?* The answer to this question reveals a tendency in the individuals to join and to stay within a school for different reasons, like hunting or protection.

In the technological side of dealing with flocking or schooling phenomena, scientists focus on a local problem. They try to answer the question: *How do individuals avoid collision and align themselves?* They supposed that individuals are already in a flock. A global problem is to consider, in addition to the local problem, the trend to form a flock from scattered individuals or small flocks.

The consideration of the flocking local problem is clear in the rules of Reynolds [Reynolds, 1987]. The rules of Reynolds focus on collision avoidance, velocity matching, and flock centering, which are local flocking problems. In [Olfati-Saber, 2006], the algorithm 1 of *Olfati-Saber* embodied the rules of Reynolds. This algorithm failed to perform the flocking, as emphasized by the author, since it leads to a fragmented flock. Moreover, in the second algorithm of *Olfati-Saber*, a rendezvous point was defined to attract the individuals through a navigational feedback in order to form a flock.

From a robotics and control theory point of view, flock formation should be autonomous. Therefore, no rendezvous point should be predefined to form a flock. However, a rendezvous point could be defined if we want the flock achieving a navigational task toward a destination.

In fact, defining an attracting rendezvous point and restricting the neighboring region limit the computational cost due to the increase of connections between

agents. This limitation of neighboring region leads to a fragmentation phenomenon, which appeared in the simulations results of *Olfati-Saber* without a rendezvous point [Olfati-Saber, 2006], see Figure (2.1). The same considerations are also found in [Antonelli et al., 2010].

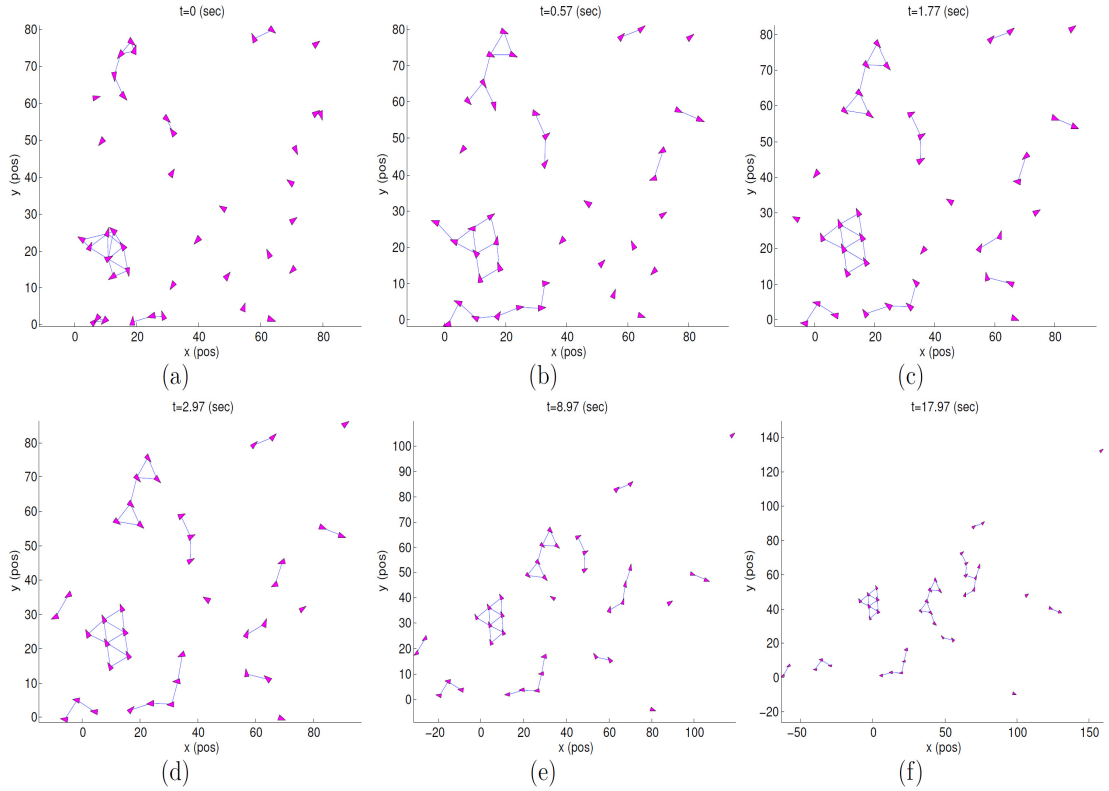


Figure 2.1 – Fragmentation problem in flocking algorithm 1 of [Olfati-Saber, 2006]

The limitation of neighboring regions is useful to fix distances between agents and to reduce computational cost on each agent since the number of neighboring agents will be limited. However, naturally, the distances between flock or school individuals are not fixed [Partridge, 1982]. In this work, we prefer to follow the natural phenomenon of flocking and consider variable security distances between agents.

In this work, we introduce a new behavior, which views the flocking problem from a global perspective. We suppose that individuals in a flock are not yet grouped. This behavior is the *Aggregation behavior*.

Definition 2.1. *(Aggregation behavior) It is the trend of individuals or sub-flocks to join each other, in their range of sight, in order to form a flock.*

To model such behavior in a flock of UAVs, we propose a new set for each UAV. This set includes all the individuals in a UAV range of sight. It is defined as $\Gamma_i = \{\nu_j \in V : \|q_j - q_i\| < b\}$, where b is the UAV range of sight and $b > c$ (c is the

neighboring region).

Unlike the works of *Olfati-Saber* in [Olfati-Saber, 2006] and *G. Antonelli et al* in [Antonelli et al., 2010], there is no need for a rendezvous point to gather UAVs in a flock. Moreover, we realize that the fragmentation problem, issued in the two aforementioned works, is overcome by implementing this behavior in the individuals of the flock.

2.5 Flocking by trajectory generation

The problem of multiple agents (or vehicles) is that the computational cost and the control design complexity increase with the raise of the number of agents and the connectivity between them [Olfati-Saber and Murray, 2002]. In this section, we introduce a new strategy of multiple UAVs control that shows the independence (or slight-dependence) of control design on the number of UAVs and on the connectivity. Moreover, we assume that each UAV can measure the states of all the agents in its range of sight.

The most crucial part of our work is the trajectory generation. By trajectory generation, we mean that on each UAV, we generate an on-line trajectory to be followed. This trajectory is computed (or generated) aboard each UAV by using the measured states of neighboring agents. We emphasize here that the idea of trajectory generation is different from the well-known trajectory planning concept in robotics. In fact, in this chapter, we claim that the flocking is a trajectory generation problem rather than a control design issue. The desired trajectory for each UAV is generated by using the measured states of the UAVs in its field of view.

Drawing inspiration from *Reynolds* [Reynolds, 1987], one of the objectives of multi-UAVs flocking control is to ensure a collision-free region between the flock individuals. The control objective could be written as follows:

$$\|q_j(t) - q_i(t)\| \to d\,, \quad t \to \infty\,, \quad \forall j \in \eta_i, \forall i \in \mathcal{V} \tag{2.12}$$

This objective could be written in a vectorial form as:

$$q_j(t) - q_i(t) \to d\,\mathbf{n}_{ij}(t)\,, \quad t \to \infty, \quad \forall j \in \eta_i, \forall i \in \mathcal{V} \tag{2.13}$$

where d is a security distance or the radius of the collision-free region, with $d < c$. $\mathbf{n}_{ij}$ is the unit vector indicating the direction from UAV i to UAV j. This vector

could be represented by two different ways, as in following equations.

$$\mathbf{n}_{ij} = \frac{q_j - q_i}{\|q_j - q_i\|} \tag{2.14}$$

$$\mathbf{n}_{ij} = [\cos\theta_{ij} \;\; \sin\theta_{ij}]^T \tag{2.15}$$

where θ_{ij} is the orientation angle. The representation in equation (2.15) is given in figure 2.2, with $f = 2$.

Moreover, another objective of multi-UAVs flocking is the velocity matching or consensus of all the UAVs in the flock. This objective could be represented mathematically as follows:

$$\dot{q}_j(t) - \dot{q}_i(t) \to 0\,, \quad t \to \infty, \quad \forall j \in \eta_i, \forall i \in \mathcal{V} \tag{2.16}$$

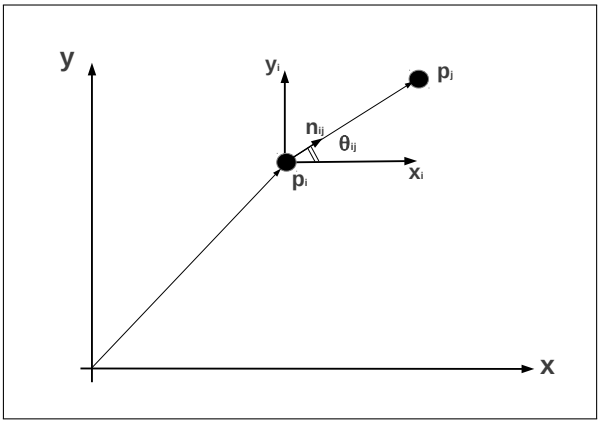


Figure 2.2 – Unit vector representation.

Let us consider the simple case of two UAVs i and j that detect each other and aggregate to form a flock. We deal with this problem as a trajectory tracking problem. The error system in UAV i could be, therefore, written by using equations (2.13) and (2.16), as the following:

$$\mathbf{e}_{ij} = \begin{pmatrix} q_i - q_j + d\,\mathbf{n}_{ij} \\ \dot{q}_i - \dot{q}_j \end{pmatrix} = \mathbf{x}_i - \mathbf{r}_{ij} \tag{2.17}$$

where $\mathbf{r}_{ij}$ is considered as the state of the desired trajectory. In other words, it is the desired relative state of UAV i. In fact, $\mathbf{r}_{ij}$ is the state of UAV j, shifted by a security distance d. $\mathbf{r}_{ij}$ is written as follows:

$$\mathbf{r}_{ij} = \mathbf{x}_j - \mathbf{x}_{dij} \tag{2.18}$$

where $\mathbf{x}_{dij}$ is a shifting state defined as:

$$\mathbf{x}_{dij} = \begin{pmatrix} d\,\mathbf{n}_{ij} \\ \mathbf{0} \end{pmatrix} \in \mathbb{R}^l \tag{2.19}$$

According to the linear quadratic control discussed in previous section, and using equation (2.11), the control law, in the UAV i, that ensures the convergence of UAVs and the control constraint in (2.13) is written as follows:

$$\mathbf{u}_{ij} = -K\mathbf{x}_i + K_r\,\mathbf{r}_{ij} \tag{2.20}$$

We calculate the control law of UAV j by the same way.

In the problem of multiple UAVs, we define the vector of shifting states of a UAV i as $\mathbf{x}_{di} = col(\mathbf{x}_{di1}, ..., \mathbf{x}_{diN}) \in \mathbb{R}^{l\times N}\ \forall j \in \eta_i$. Moreover, we can obviously see that each UAV in a flock has more than one neighbor. Therefore, the dimensions of the vector of the neighboring states $\mathbf{x}_{\eta_i}$ and the state $\mathbf{x}_i$ of a UAV i are not compatible. This problem could be solved by mapping $\mathbf{x}_{\eta_i}$ to be compatible with the dimensions of $\mathbf{x}_i$.

Let $\xi : \mathbb{R}^{l\times N} \rightarrow \mathbb{R}^l$ be a differentiable function that is used to generate the trajectory of a UAV i in the flock in order to achieve a desired behavior. Then $\mathbf{r}_i$ could be written as:

$$\mathbf{r}_i = \xi(\mathbf{x}_{\eta_i}, \mathbf{x}_{di}) \tag{2.21}$$

The function in (2.21) means that the flocking problem is focused on the reference generation rather than the control design. Moreover, it is not necessary to design multiple controllers for each UAV. This method decreases the computational cost.

2.5.1 Average strategy

Different ways could be proposed to generate a trajectory from the measured states of the neighboring agents. In the following, we propose a strategy for flocking of multiple UAVs based on averaging. We define the average function of a vector $z = col(z_1, ..., z_n)$ as follows:

$$Ave_n(z) = \frac{1}{n}\sum_{j=1}^{n} z_j \tag{2.22}$$

Now, we are ready to present our first control strategy in the following proposition.

Proposition 2.1. *(Average control strategy) Consider q UAVs that tend to form a flock. The following control law of UAV $i \in \mathcal{V}$*

$$\mathbf{u}_i = -K\mathbf{x}_i + K_r\, \xi_{avg}(\mathbf{x}_{\eta_i}, \mathbf{x}_{di}) \tag{2.23}$$

ensures the aggregation behavior. The function $\xi_{avg}(\mathbf{x}_{\eta_i}, \mathbf{x}_{di})$ is written as follows:

$$\xi_{avg}(\mathbf{x}_{\eta_i}, \mathbf{x}_{di}) = Ave_N(\mathbf{x}_{\eta_i} - \mathbf{x}_{di}) \tag{2.24}$$

Proof. The LQR control law in (2.20) stabilizes the UAV i and ensures the aggregation of two UAVs. For N UAVs $\in \eta_i$, we need q control laws. Then, we have:

$$\begin{array}{ccc} \mathbf{u}_{i1} & = & -K\mathbf{x}_i + K_r\, \mathbf{r}_{i1} \\ \vdots & \vdots & \vdots \\ \mathbf{u}_{iN} & = & -K\mathbf{x}_i + K_r\, \mathbf{r}_{iN} \end{array} \tag{2.25}$$

A solution to decrease the number of controllers is to calculate the average of the N control laws and apply the output of the average to the system. We obtain:

$$\mathbf{u}_i = \frac{1}{N}\sum_{j=1}^{N} \mathbf{u}_{ij} \tag{2.26}$$

Using (2.25) in (2.26) we get:

$$\mathbf{u}_i = -K\mathbf{x}_i + \frac{K_r}{N}\sum_{j=1}^{N} \mathbf{r}_{ij} \tag{2.27}$$

Replacing $\mathbf{r}_{ij}$ by its expression in (2.18), then we get the resultant control law given by equation (2.23).

□

Remark 2.1. *The design of the control law in (2.23) is independent of the number of UAVs. Moreover, the linear nature of the LQR control is one of the main characteristics that allowed us to develop this strategy.*

Remark 2.2. *The security distance d is user-defined and it is dependent on the number of UAVs. Gaps between flock individuals will converge to distances around d. The security distance should be set higher when the number of UAVs increases. The distance d should be determined carefully in order to avoid collision, since the state of each UAV converges to an average vector.*

Conjecture 1. *Consider q UAVs that tend to form a flock. The control law in (2.23) asymptotically stabilizes the flock formation.*

This conjecture is validated by simulation and experimental work, see chapter 4. However, we will seek an analytical proof of this conjecture in our future works.

2.5.2 Sum strategy

Another way of trajectory generation from the measured states of neighboring agents is proposed in the following proposition. We define the sum function of a vector $z = col(z_1, ..., z_n)$ as follows:

$$Sum_n(z) = \sum_{j=1}^{n} z_j \tag{2.28}$$

Proposition 2.2. *(Sum control strategy) Consider N agents that tend to form a flock. The following control law of an agent $i \in V$*

$$\mathbf{u}_i = -N\,K\,\mathbf{x}_i + K_r\,\xi_{sum}(\mathbf{x}_{\eta_i}, \mathbf{x}_{di}) \tag{2.29}$$

ensures the aggregation behavior. The function $\xi_{sum}(\mathbf{x}_{\eta_i}, \mathbf{x}_{di})$ is written as the following:

$$\xi_{sum}(\mathbf{x}_{\eta_i}, \mathbf{x}_{di}) = Sum_N(\mathbf{x}_{\eta_i} - \mathbf{x}_{di}) \tag{2.30}$$

Proof. Similar to the proof of proposition 2.1, another solution is proposed by calculating the sum of the N control laws and applying the output of the sum to the system, we obtain:

$$\mathbf{u_i} = \sum_{j=1}^{N} \mathbf{u_{ij}} \tag{2.31}$$

Therefore, the resultant control law is given by equation (2.29).

□

Remark 2.3. *Remarks 2.1 and 2.2 are still valid in the sum strategy. The difference is that one part of the control law is weighted by N. N has to be known and updated from measurements.*

Conjecture 2. *Consider q UAVs that tend to form a flock. The control law in (2.29) asymptotically stabilizes the flock formation.*

This conjecture is validated by simulation and experimental work, see chapter 4. However, we will seek an analytical proof of this conjecture in our future works.

2.5.3 Navigational behavior

In this section, we introduce a navigational behavior that allows a flock of multiple UAVs to navigate from initial positions toward a destination. The destination is predefined by the user and stocked in the memory of UAVs. The navigation of a multi-UAVs flock is in a free space, *i.e.* there are no obstacles. In this section, no obstacles are faced in the path of the navigation.

The control strategy used to perform the navigational task is similar to the average strategy. It consists in defining a navigational vector that contains the position and the velocity of the destination. It could be written as $\mathbf{x}_{nav} = [q_{nav}\ \dot{q}_{nav}]^T \in \mathbb{R}^l$. The destination could be also a trajectory to be followed by the whole fleet of UAVs.

Now, we define the following function to achieve the aggregation and the navigation behaviors:

$$\xi_{avg}(\mathbf{x}_{\eta_i}, \mathbf{x}_{di}, \mathbf{x}_{nav}) = Ave_{N+1}(col(\mathbf{x}_{\eta_i} - \mathbf{x}_{di}, \mathbf{x}_{nav})) \tag{2.32}$$

Therefore, the control law that allows the multi-UAVs system to make the aggregation and the navigation behaviors is given as the following:

$$\mathbf{u}_i = -K\mathbf{x}_i + K_r\xi_{avg}(\mathbf{x}_{\eta_i}, \mathbf{x}_{di}, \mathbf{x}_{nav}) \tag{2.33}$$

A physical interpretation of this strategy could be explained by the fact that we consider the navigational vector as a state of a virtual agent that is detected by all the agents in the flock.

We can perform the navigational behavior in the sum strategy by adding a dedicated navigational feedback. In this case, we will have two control objectives, the first one is the performance of aggregation behavior and the second one is the navigation toward a predefined rendezvous point or trajectory defined by the user and known by all the UAVs. These could be resumed in the following control law:

$$\mathbf{u}_i = -N\,K\,\mathbf{x}_i + \,K_r\,\xi_{sum}(\mathbf{x}_{\eta_i}, \mathbf{x}_{di}) + \,\mathbf{u_{nav}}_i \tag{2.34}$$

where

$$\mathbf{u_{nav}}_i = -\,K_n\,(\mathbf{x}_i - \mathbf{x}_{nav}) \tag{2.35}$$

with K_n is a constant gain matrix.

2.6 Modeling and flocking of Multiple quadrotors

In chapter 1, we introduced the modeling and the control of a quadrotor as a special type of UAVs. We distinguish two dynamics: rotational and translational dynamics. This type of dynamics makes the control of quadrotors in the context of flocking of a special particularity. Recently, researchers in robotics performed several experimental works on multiple-quadrotor control as in [Kushleyev et al., 2013], [Schollig et al., 2010] and [Franchi et al., 2012]. The works in [Kushleyev et al., 2013] and [Schollig et al., 2010] consider multiple-quadrotor control as a problem of trajectory planning. Each quadrotor follows a collision-free trajectory, which is computed in a central base station. In the presence of disturbances, it is not clear that a collision-free formation could be ensured. Moreover, we believe that multiple-quadrotor control could be more challenging if it is implemented aboard quadrotors and if it considers interactions between them. On the other hand, the work in [Franchi et al., 2012] did not deal with the nonlinear model of quadrotors. Instead, it uses a high-level control that deals with an ideal double-integrator model, and the smooth output of this model is used to drive stabilized quadrotors. In the sequel of our work, we try to deal directly with the nonlinear model of quadrotors, and then we apply the proposed control laws aboard the quadrotors to stabilize and navigate the flock.

In this work, we introduce a new architecture to control multiple quadrotors in the flocking context. We separate the control problem of multiple UAVs in two parts. The first part is the control of internal dynamics of each UAV. We mean by internal dynamics, the altitude z and the rotational dynamics of each UAV. This part will not be involved in the algorithm of flocking. The second part is the control of the x, y translation and flocking dynamics of multiple UAVs.

In fact, we specify a fixed desired altitude z and heading ψ. Moreover, outputs of $x-y$ translation and flocking controllers are feedforwarded to the inputs of controllers of roll and pitch $\phi - \theta$ angles. Figure 2.3 shows the overall control architecture.

2.6.1 Control of quadrotor internal dynamics

Before starting the description of the control strategy of a UAV in a flocking perspective, we begin by linearizing the nonlinear model (1.4) about the origin $(\xi = \mathbf{0}, \dot{\xi} = \mathbf{0}, \eta = \mathbf{0}, \dot{\eta} = \mathbf{0}, F = 0, \tau = 0)$.

We remind here that for a nonlinear system of the form:

$$\dot{x} = f(x, u)$$

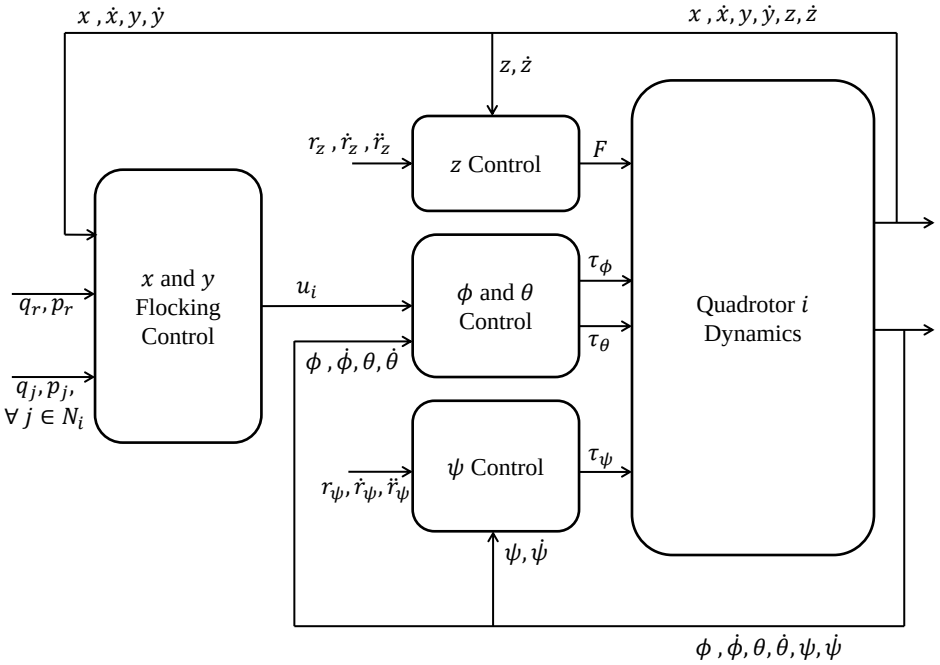


Figure 2.3 – Control architecture of quadrotor in a flocking perspective. Internal dynamics are controlled separately from the $x - y$ translation and flocking dynamics

with $x = [x_1, ..., x_n]^T$ is the state of the system, $u = [u_1, ..., u_m]$ is its control input, and $f(.) = [f_1(.), ..., f_n(.)]^T$, the linearization of this system about its origin ($x = 0$, $u = 0$) is given by the linear system:

$$\dot{x} = Ax + Bu$$

where

$$A = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(x,u)\Big|_{x=0,u=0} & \dots & \frac{\partial f_1}{\partial x_n}(x,u)\Big|_{x=0,u=0} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1}(x,u)\Big|_{x=0,u=0} & \dots & \frac{\partial f_n}{\partial x_n}(x,u)\Big|_{x=0,u=0} \end{pmatrix}$$

and

$$B = \begin{pmatrix} \frac{\partial f_1}{\partial u_1}(x,u)\Big|_{x=0,u=0} & \dots & \frac{\partial f_1}{\partial u_m}(x,u)\Big|_{x=0,u=0} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial u_1}(x,u)\Big|_{x=0,u=0} & \dots & \frac{\partial f_n}{\partial u_n}(x,u)\Big|_{x=0,u=0} \end{pmatrix}$$

We recall here the nonlinear model of quadrotor as in (1.8) which is given as

follows:

$$\dot{x} = \nu_x \tag{2.36a}$$
$$\dot{y} = \nu_y \tag{2.36b}$$
$$\dot{z} = \nu_z \tag{2.36c}$$
$$\dot{v}_x = (1/m)(c_\phi s_\theta c_\psi + s_\phi s_\psi)F \tag{2.36d}$$
$$\dot{v}_y = (1/m)(c_\phi s_\theta s_\psi - s_\phi c_\psi)F \tag{2.36e}$$
$$\dot{v}_z = -g + c_\phi c_\theta F \tag{2.36f}$$
$$\dot{\phi} = w_{B_x} + s_\phi t_\theta w_{B_y} + c_\phi t_\theta w_{B_z} \tag{2.36g}$$
$$\dot{\theta} = c_\phi w_{B_y} - s_\phi w_{B_z} \tag{2.36h}$$
$$\dot{\psi} = \frac{s_\phi}{c_\theta} w_{B_y} + \frac{c_\phi}{c_\theta} w_{B_z} \tag{2.36i}$$
$$\dot{w}_{B_x} = (\frac{J_y - J_z}{J_x}) w_{B_y} w_{B_z} + (\frac{1}{J_x})\tau_\phi \tag{2.36j}$$
$$\dot{w}_{B_y} = (\frac{J_z - J_x}{J_y}) w_{B_x} w_{B_z} + (\frac{1}{J_y})\tau_\theta \tag{2.36k}$$
$$\dot{w}_{B_z} = (\frac{1}{J_z})\tau_\psi \tag{2.36l}$$

The result of linearization about the origin is given as follows:

$$\dot{x} = \nu_x \tag{2.37a}$$
$$\dot{y} = \nu_y \tag{2.37b}$$
$$\dot{z} = \nu_z \tag{2.37c}$$
$$\dot{v}_x = (1/m)\theta \tag{2.37d}$$
$$\dot{v}_y = -(1/m)\phi \tag{2.37e}$$
$$\dot{v}_z = F \tag{2.37f}$$

$$\dot{\phi} = w_{B_x} \tag{2.38a}$$
$$\dot{\theta} = w_{B_y} \tag{2.38b}$$
$$\dot{\psi} = w_{B_z} \tag{2.38c}$$
$$\dot{w}_{B_x} = (1/J_x)\tau_\phi \tag{2.38d}$$
$$\dot{w}_{B_y} = (1/J_y)\tau_\theta \tag{2.38e}$$
$$\dot{w}_{B_z} = (1/J_z)\tau_\psi \tag{2.38f}$$

Equations (2.37a) through (2.37f) represent the translational dynamics of the UAV and equations (2.38a)-(2.38f) represent the rotational dynamics.

The control of the UAV will be as follows. Firstly, we use PID controllers to control the z dynamics (equations (2.37c) and (2.37f)) and the ψ dynamics (equations (2.38c) and (2.38f)). The control inputs of these two subsystems are given by the general expression of a PID controller in a tracking perspective, as follows:

$$F = \ddot{r}_z + k_{pz}(r_z - z) + k_{dz}(\dot{r}_z - \dot{z}) + k_{iz}\int (r_z - z)\,\mathrm{d}t \tag{2.39}$$

$$\tau_\psi = \ddot{r}_\psi + k_{p\psi}(r_\psi - \psi) + k_{d\psi}(\dot{r}_\psi - \dot{\psi}) + k_{i\psi}\int (r_\psi - \psi)\,\mathrm{d}t \tag{2.40}$$

where r_z and r_ψ are the desired altitude and heading of the UAV, and the constants $k_{(.)} > 0$ are controlling gains to be defined by the user.

Secondly, the remaining equations represent the $x-y$ translational dynamics and the $\phi-\theta$ rotational dynamics. To control these dynamics we use an approach, which is introduced in chapter 1, similar to the backstepping technique. This approach is widely used in the control of quadrotors [Bouabdallah and Siegwart, 2007]. First, we consider ϕ and θ in equations (2.37d) and (2.37e) as virtual control inputs of the translational dynamics. Then, we design a controller for the rotational dynamics (equations (2.38a)-(2.38f)), which has a double-integrator form. For this purpose, we use the nested saturation approach [Teel, 1992], [Johnson and Kannan, 2003], [Sanahuja, 2010] and [Kendoul et al., 2007]. Hence, the control inputs that stabilize the $\phi - \theta$ rotational dynamics are given as follows:

$$\tau_\phi = -Sat_{\phi_1}\left(k_{\phi_1}\dot{\phi} + Sat_{\phi_2}\left(k_{\phi_2}\dot{\phi} + k_{\phi_1}k_{\phi_2}\left(\phi - r_\phi\right)\right)\right) \tag{2.41}$$

$$\tau_\theta = -Sat_{\theta_1}\left(k_{\theta_1}\dot{\theta} + Sat_{\theta_2}\left(k_{\theta_2}\dot{\theta} + k_{\theta_1}k_{\theta_2}\left(\theta - r_\theta\right)\right)\right) \tag{2.42}$$

where $Sat_\alpha(x) = sign(x)min(|x|, \alpha)$ is a saturation function, with α_i being a real positive constant. The constants $k_{(.)}$ are tuning gains. r_ϕ and r_θ are desired references, which are the outputs of the $x - y$ position controllers.

2.6.2 Flocking of multiple quadrotors by trajectory generation

In this part, we apply the average and the sum strategies, introduced in the previous section, to achieve a flocking of multiple quadrotors. As aforementioned, in this work, we apply the flocking control on the $x-y$ translational dynamics of quadrotors. Using the translational and flocking dynamics in (2.37a, 2.37b, 2.37d, 2.37e), we can write their state-space linear system, as in (2.1), as follows:

$$\left\{ \begin{array}{rcl} \dot{\mathbf{x}}_i & = & A_i\,\mathbf{x}_i + B_i\,\mathbf{u}_i \end{array} \right. \tag{2.43}$$

where $\mathbf{x}_i = [x,\, y,\, \nu_x,\, \nu_y]^T$ and $u = [\theta,\, \phi]^T$. The matrices A_i and B_i are given as follows:

$$A_i = \begin{pmatrix} \mathbf{0}_2 & \mathbf{I}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 \end{pmatrix} \quad B_i = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ (1/m) & 0 \\ 0 & -(1/m) \end{pmatrix}$$

with $\mathbf{0}_2$ is a 2×2 zeros matrix, and $\mathbf{I}_2$ is a 2×2 identity matrix.

Therefore, all that remains to be done now is to apply our proposed control strategies, (2.23), (2.29) or (2.33), in the flocking or in the navigation perspective of multiple quadrotors. We emphasize here that we should choose Q and R matrices appropriately. Then, the references r_θ and r_ϕ of the $\theta - \phi$ control, in equations (2.42) and (2.41), will be replaced by the output of the flocking strategy, that is, $[r_\theta, r_\phi]^T = \mathbf{u}_i$.

2.7 Simulation results

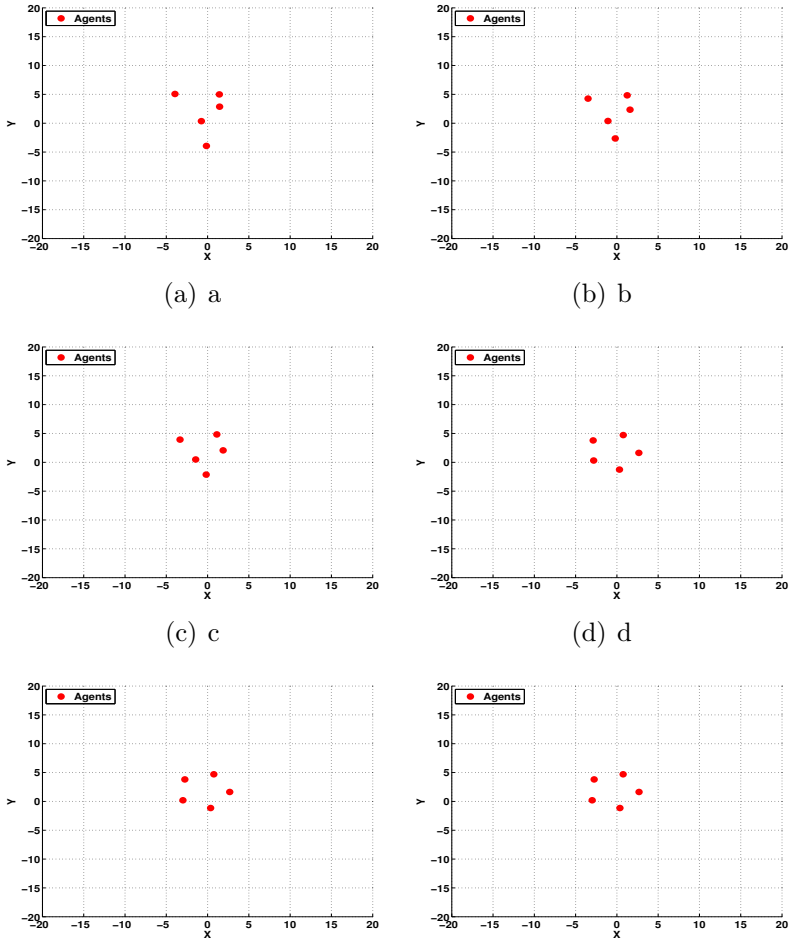
In this section, we illustrate simulations of the multi-UAVs system. We simulate both behaviors, presented in the previous section, by using MATLAB. The simulation is done in a two-dimensional space. The initial positions of UAVs are generated randomly by using the normal distribution function. Moreover, the initial velocities of UAVs are set at zeros. The step size of the simulation is $0.01s$. The weighting matrices are: $Q = \mathbf{I}_4$ and $R = 10^{-3} \times \mathbf{I}_2$. The matrix K is calculated by using the *lqr* MATLAB function [Hespanha, 2009]. Furthermore, UAVs are represented as red circles, where the real position of a UAV is the center of the circle. For clarity purposes, the size of UAVs is chosen greater than simulation scales, which could yield to ambiguous separation distances.

2.7.1 Aggregation behavior

In this part, we simulate the aggregation of 5 UAVs. The variance and the mean of the initial positions are set at 21 and 0, respectively. The range of the two-dimensional simulation plan is the x y position square $[-20, 20]$. The flocking parameters are defined as the following: $d = 5$, $c = 30$. Therefore, we suppose that the range of sight of each UAV covers the entire simulation plan. Moreover, we suppose that each UAV can measure the states of all the other UAVs in its range of sight.

Figure 2.4 shows six snapshots of running simulation over time. This simulation is performed by using the *average control strategy* (2.23). The UAVs start scattered in the simulation plan. By analyzing the simulation, each UAV performs a repulsion

action when there are some UAVs in its collision-free region. This action ensures a security distance between UAVs. Moreover, each UAV activates the aggregation behavior when a UAV or more are in its range of sight. Therefore, UAVs start to converge to each other to form a flock. The convergence is performed without collision. After a sufficient time, a flock of UAVs is formed and stabilized. In this simulation, we do not define a rendezvous point, and there is no fragmentation in the flock. Moreover, we can see clearly security distances between individuals. These security distances ensure collision-free flock.



(a) a (b) b

(c) c (d) d

(e) e (f) f

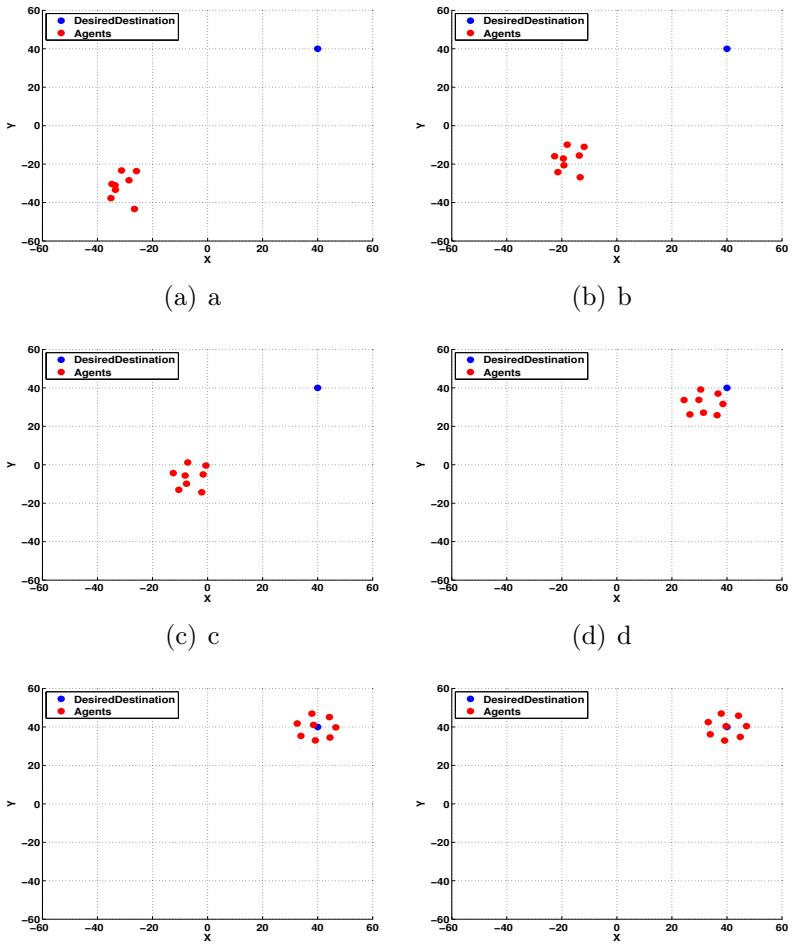
Figure 2.4 – multi-UAVs aggregation snapshots (Average strategy). No fragmentation in the flock, even if there is no rendezvous point.

2.7.2 Navigation

In this part, we illustrate the navigation of 8 UAVs in a free space. The variance of initial positions is 21 and the mean is -30. The simulation plan is expanded to be the x y position square $[-60, 60]$. The flocking parameters are defined as the

following: $d = 12$, $c = 30$. The desired destination is the position $(40, 40)$, so the navigation vector is $[40\ 40\ 0\ 0]^T$.

Fig. 2.5 shows six snapshots of the navigation behavior realized by using the control law in (2.33), which is applied in each UAV in the flock. The flock performs the aggregation behavior while navigating toward the desired destination. Safety distances are kept throughout the navigation path. Finally, the flock is uniformly formed and stabilized at the destination position.



(a) a (b) b

(c) c (d) d

(e) e (f) f

Figure 2.5 – multi-UAVs navigation snapshots. The algorithm ensures the aggregation and the navigation.

2.8 Conclusion

In this chapter, we addressed the control problem of multiple Unmanned Aerial Vehicles (UAVs) flocking by using a behavior-based strategy. We conceived a behavior intending to address the control design towards a successful achievement

of the flocking task without fragmentation. Moreover, through its implementation in UAVs, no rendezvous point was needed to perform flocking. We designed an LQR control law, which is independent of the number of UAVs in the flock. Our proposed strategy dealt with the flocking problem from a trajectory generation perspective. Simulation results showed the efficiency of our strategy in the aggregation and in the navigation of multi-UAVs flocking in a free space.

Flocking by consensus algorithms

Contents

3.1 Introduction

Our interest in this chapter is to perform a real-time flocking of multiple UAVs in the context of system of systems. We propose control methods that are based on the flocking and consensus algorithm introduced by *Olfati-Saber* in [Olfati-Saber, 2006]. The flocking control law in [Olfati-Saber, 2006] was mainly proposed for double-integrator linear system. In this chapter, we propose four improved versions of this control law aiming at being compatible with the nonlinear model of quadrotors and experimental works. The control laws are run aboard each quadrotor in the flock. By running the control law, each quadrotor interacts with its neighbors to ensure a collision-free flocking.

3.2 Preliminaries

The design of the $x - y$ controllers will be similar to the flocking algorithm in [Olfati-Saber, 2006]. In addition, we use our architecture proposed in chapter 2 (cf. 2.6). In equations (2.37a), (2.37b), (2.37d), (2.37e), we take $q_i = [x \ y]^T$, $p_i = [\nu_x \ \nu_y]^T$, and $u_i = [\frac{1}{m}\theta \ \frac{-1}{m}\phi]^T$. Therefore, the flocking dynamics could be written as follows:

$$\begin{aligned} \dot{q}_i &= p_i \\ \dot{p}_i &= u_i \end{aligned} \tag{3.1}$$

In this section, we discuss the control of such double-integrator translation system from a flocking perspective. We begin by introducing the basic principles of controlling a multi-agent system. The controlling algorithm is introduced by Olfati-Saber in [Olfati-Saber, 2006].

In section (1.4), we have introduced the preliminaries of modeling multi-agent flocking using graph theory. A symmetric adjacency matrix was introduced to represent an undirected graph of agents. This matrix is used in graph theory to describe topological connections between agents, that is, it does not depend on the positions of agents. Hence, we need to define another type of matrix to depict the position-based connections between agents in the flock. To introduce this type of matrix, two functions need to be defined: "σ-norm" and a bump function.

A "σ-norm" is a map $\mathbb{R}^n \to \mathbb{R}^+$ of a vector $z \in \mathbb{R}^n$, defined as:

$$\|z\|_\sigma = \frac{1}{\varepsilon}\left[\sqrt{1+\varepsilon\|z\|^2} - 1\right] \tag{3.2}$$

where $\varepsilon > 0$ and $\mathbb{R}^+$ is the set of non-negative real numbers. The gradient of σ-norm is defined by:

$$\sigma_\varepsilon(z) \triangleq \nabla_z \|z\|_\sigma = \frac{z}{\sqrt{1+\varepsilon\|z\|^2}} = \frac{z}{1+\varepsilon\|z\|_\sigma} \tag{3.3}$$

In fact, σ-norm is not a norm but its importance is that it is differentiable everywhere, unlike the Euclidean norm that is not differentiable at $z = 0$.

The bump function $\rho_h : \mathbb{R}^+ \to [0,1]$ with $h \in (0,1)$ is defined as:

$$\rho_h(z) = \begin{cases} 1 & \text{if } z \in [0,h) \\ \frac{1}{2}\left[1+\cos\left(\pi\frac{z-h}{1-h}\right)\right] & \text{if } z \in [h,1] \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

$\rho_h(z)$ is a smooth and scalar function that varies between 0 and 1. Using this bump function in corporation with the σ-norm we can define the spacial adjacency matrix $A(q) = [a_{ij}(q)]$ where $a_{ij}(q)$ are its elements given as follows:

$$a_{ij}(q) = \begin{cases} 0 & \text{if } i = j \\ \rho_h(\|q_j - q_i\|_\sigma / \|c\|_\sigma) & \text{if } j \neq i \end{cases} \tag{3.5}$$

A smooth collective potential function is used to design the flocking algorithm of multiple quadrotors. This function is given as follows:

$$V(q) = \frac{1}{2} \sum_{i} \sum_{j \neq i} \Psi_\alpha(\|q_j - q_i\|_\sigma) \tag{3.6}$$

where

$$\Psi_\alpha(z) = \int_{d_\alpha}^{z} \Phi_\alpha(s) ds \tag{3.7}$$

Φ_α is defined by:

$$\begin{aligned} \Phi_\alpha(z) &= \rho_h(z/c_\alpha)\Phi(z - d_\alpha) \\ \Phi(z) &= \tfrac{1}{2}[(a+b)\sigma_1(z+e) + (a-b)] \end{aligned} \tag{3.8}$$

with $\sigma_1(z) = z/\sqrt{1+z^2}$. The function $\Phi(z)$ is uneven and sigmoidal, with $0 < a \leq b$ and $e = |a-b|/\sqrt{4ab}$ that ensures $\Phi(0) = 0$.

It follows from the above formulas that $\Psi_\alpha(z)$ is a smooth pairwise repulsive/attractive potential function. It has a minimum at $z = d_\alpha = \|d\|_\sigma$, and it has a finite cut-off at $c_\alpha = \|c\|_\sigma$. The finite cut-off feature of this function is a fundamental source of scalability of the flocking algorithm [Olfati-Saber, 2006]. Moreover, every local minimum of $V(q)$ is an α-lattice.

An important matrix in graph theory and flocking control is the graph Laplacian. In our graph $G = (\mathcal{V}, E)$ with M UAVs, the graph Laplacian $M \times M$ matrix is defined as follows:

$$L = D(\mathcal{A}(q)) - \mathcal{A}(q) \tag{3.9}$$

with $\mathcal{A}(q)$ being the spacial adjacency matrix and $D(\mathcal{A}(q))$ being a diagonal matrix called, degree matrix of the graph G, and $\sum_{j=1}^{M} a_{ij}(q)$ being its diagonal elements. This matrix is positive semidefinite. The eigenvalues of the Laplacian matrix could be organized as :

$$\lambda_1 < \lambda_2 < ... < \lambda_M$$

The Laplacian matrix has always the eigenvector of $\mathbf{1_M} = (1, ..., 1)$ which is associated with the eigenvalue $\lambda_1 = 0$.

The Laplacian matrix satisfies the following properties [Godsil and Royle, 2001] and [Olfati-Saber, 2006]:

- sum-of-Square (SOS) property

$$S^T L S = \frac{1}{2} \sum_{(i,j) \in E} a_{ij}(S_j - S_i)^2 \ \ , S \in \mathbb{R}^M \tag{3.10}$$

- In a connected graph we have:

$$\lambda_2(L) = min\frac{S^T\, L\, S}{\|S\|^2} > 0, \;\; S \perp \mathbf{1_M} \tag{3.11}$$

An *f-dimensional graph Laplacian* is defined as follows:

$$\hat{L} = L \otimes I_f \tag{3.12}$$

where $\otimes$ being the Kronecker product and I_f being an $f \times f$ identity matrix. The property in (3.10) could be then written for $\hat{L}$ as the following:

$$\mathbf{S}^T\, \hat{L}\, \mathbf{S} = \frac{1}{2} \sum_{(i,j)\in E} a_{ij} \|\mathbf{S}_j - \mathbf{S}_i\|^2 \;\; , \mathbf{S} \in \mathbb{R}^{fM} \tag{3.13}$$

where $\mathbf{S} = col(S_1, S_2, ..., S_M)$ and $S_i \in \mathbb{R}^f$.

The control law introduced in [Olfati-Saber, 2006], which is applied on each agent with linear dynamics, such as in (3.1), and that ensures an α-*lattice* flock and navigation, is given as follows:

$$\begin{aligned} u_i &= \sum_{j\in N_i} [\Phi_\alpha(\|q_j - q_i\|_\sigma)\mathbf{n}_{ij} + a_{ij}(q)(p_j - p_i)] \\ &+ f_i^\gamma(q_i, p_i, q_r, p_r) \end{aligned} \tag{3.14}$$

where $f_i^\gamma(q_i, p_i, q_r, p_r) = -c_1(q_i - q_r) - c_2(p_i - p_r)$, $c_1, c_2 > 0$, and $\mathbf{n}_{ij} = \sigma_\varepsilon(q_j - q_i)$ as in equation (3.3).

This control law is composed of three terms. The first is the gradient-based term, which ensures the regulation of the relative interdistance vector between agents. The second is the velocity consensus term, which is analog to a derivative controller in a conventional PD control law. The last term $f_i^\gamma(.)$ is the navigational or the translational feedback control, with q_r and p_r being the desired position and velocity to be tracked. Moreover, the first and the second terms ensure the aggregation of every agent with its neighbors and the conservation of a collision-free flocking. The navigational feedback leads the whole flock to track a predefined objective trajectory or destination point. The objective trajectory is known by every agent in the flock, which ensures a fragmentation-free flocking.

3.3 Flocking control with tuning gains

We have applied the controller (3.14) in simulation (chapter 4), and we have observed that this control law could not be applied directly on nonlinear systems, such as

quadrotors. In fact, the simulation results of this control law on the nonlinear dynamics of quadrotors show oscillating movement of distances between quadrotors. This could be explained by the fact that the control law in (3.14) was applied on double-integrator linear models without considering uncertainties. We believe that, the elegant control law in (3.14) could be applied on nonlinear systems if we add some tuning gains to its gradient-based and velocity consensus terms. The additional gains will compensate for uncertainties of the nonlinear model. The modified control law is given in the following equation:

$$\begin{aligned} u_i = \sum_{j \in N_i} \Big[& K_p \Phi_\alpha(\|q_j - q_i\|_\sigma)\mathbf{n}_{ij} + K_p^{'} a_{ij}(q)(q_j - q_i) \\ & + K_d a_{ij}(q)(p_j - p_i) \Big] + f_i^\gamma(q_i, p_i, q_r, p_r) + \dot{p}_r \end{aligned} \tag{3.15}$$

where $K_p, K_p^{'}, K_d > 0$ are constant scalar tuning gains and their values depend on the quadrotor device. K_p and K_d are user defined. They give a relative freedom to the user to apply the control law on different quadrotor devices. $\dot{p}_r$ is the acceleration of the objective trajectory. We have added it in our control law since it was omitted in the control law of [Olfati-Saber, 2006].

To prove the efficiency of our control law, we introduce a perturbation term, $\delta_i \equiv \delta_i(q_i, p_i)$, to the agent model as follows:

$$\begin{aligned} \dot{q}_i &= p_i \\ \dot{p}_i &= u_i + \delta_i \end{aligned} \tag{3.16}$$

For the system in (3.16), we introduce the following lemma and assumption.

Assumption 3.1. *The perturbation term δ_i is an unknown function, but it has a known upper bound such that, $\|\delta_i\| \leq \kappa$ and $\kappa > 0$.*

This assumption is realistic, since in practical and real-time problems we cannot estimate perturbation signals. Instead, we prefer to define an upper bound of perturbations that we hope the system can tolerate.

Lemma 3.1. *[Olfati-Saber, 2006](Spatial-Order): Every local minima of $V(q)$ is an α-lattice and vice versa.*

Before analyzing the stability of the multiple-UAV system that applies our control law in (3.15), we need to introduce a moving frame [Olfati-Saber, 2006], [Li et al., 2011]. As in [Li et al., 2011], the moving frame has its origin in (q_r), that is, the position of the desired trajectory or the destination point being followed by the agents of the flock. Then, the velocity of the origin of the moving frame will be (p_r). The importance of the moving frame comes from the fact that it

will be easy to prove that the solutions of the multiple-UAV system are bounded [Olfati-Saber, 2006].

Positions and velocities of UAVs in the moving frame are represented as follows:

$$q_i^r = q_i - q_r; \quad p_i^r = p_i - p_r$$

It easy to see that the relative positions and velocities are the same in both the inertial and the moving frames. That is:

$$\begin{aligned} q_j^r - q_i^r &= q_j - q_i \\ p_j^r - p_i^r &= p_j - p_i \end{aligned}$$

Hence, $V(q) = V(q^r)$, and the agent model in (3.16) will be written:

$$\begin{aligned} \dot{q}_i^r &= p_i^r \\ \dot{p}_i^r &= u_i^r + \delta_i^r \end{aligned} \tag{3.17}$$

with $u_i^r = u_i(q_i^r, p_i^r) - \dot{p}_r$.

The configuration of all UAVs in the new reference frame is $q^r = col(q_1^r, ..., q_M^r) \in \mathbb{R}^{fM}$. Therefore we have, $p^r = col(p_1^r, ..., p_M^r) \in \mathbb{R}^{fM}$, $u^r = col(u_1^r, ..., u_M^r) \in \mathbb{R}^{fM}$, and $\delta^r = col(\delta_1^r, ..., \delta_M^r) \in \mathbb{R}^{fM}$. By applying the control law in (3.15) we get the following collective dynamics of multiple UAVs :

$$\begin{aligned} \dot{q}^r &= p^r \\ \dot{p}^r &= -K_p \nabla_{q_i^r} V(q^r) - K_d \hat{L}\, p^r - c_1 q^r - c_2 p^r + \delta^r \end{aligned} \tag{3.18}$$

Moreover, we remind here the theorem 3.1 of boundedness and ultimate boundedness in [Khalil, 2002] for the system:

$$\dot{x} = f(t, x) \tag{3.19}$$

where $f : [0, \infty) \times D \to R^n$ is piecewise continuous in t and locally Lipschitz in x on $[0, \infty) \times D$, and $D \subset R^n$ is a domain that contains the origin. It is important to emphasize here that the notations in the following theorem have no relation with the notations used in this thesis.

Theorem 3.1. *(boundedness and ultimate boundedness)[Khalil, 2002]: Let $D \subset R^n$ be a domain that contains the origin and $V : [0, \infty) \times D \to R$ be a continuously differentiable function such that*

$$\alpha_1(\|x\|) \leq V(t, x) \leq \alpha_2(\|x\|)$$

$$\frac{\partial V}{\partial t} + \frac{\partial V}{\partial x} \leq -W_3(x),\ \forall \|x\| \geq \mu > 0$$

$\forall t \geq 0$ and $\forall x \in D$, where α_1 and α_2 are class $\mathcal{K}$ functions and $W_3(x)$ is a continuous positive definite function. Take $r > 0$ such that $B_r \subset D$ and suppose that

$$\mu < \alpha_2^{-1}(\alpha_1(r))$$

Then, there exists a class $\mathcal{KL}$ function β and for every initial state $x(t_0)$, satisfying $\|x\| \leq \alpha_2^{-1}(\alpha_1(r))$, there is $T \geq 0$ (dependent on $x(t_0)$ and μ) such that the solution of (3.19) satisfies

$$\|x\| \leq \beta(\|x(t_0)\|, t - t_0), \quad \forall t_0 \leq t \leq t_0 + T \tag{3.20}$$

$$\|x\| \leq \alpha_1^{-1}(\alpha_2(\mu)), \quad \forall t \geq t_0 + T \tag{3.21}$$

Moreover, if $D = R^n$ and α_1 belongs to class $\mathcal{K}_\infty$, then (3.20) and (3.21) hold for any initial state $x(t_0)$, with no restriction on how large μ is.

When the system (3.19) is not explicitly dependent on t, such as our multiple-UAV system, we consider only the boundedness of the solution and not the ultimate boundedness, and we drop the time dependence in the theorem.

Now we are ready to present our proposition.

Proposition 3.1. *Consider the multiple-UAV dynamical system in (3.18) that applies the distributed control law in (3.15). Suppose that the perturbation satisfies assumption (3.1), then the solution of the multiple-UAV dynamical system (3.18) is bounded.*

Proof. In the beginning, we consider the collective system in (3.18) without perturbations. Taking inspiration from the backstepping technique [Khalil, 2002], we distinguish two subsystems in cascade in (3.18). In the first subsystem:

$$\dot{q}^r = p^r \tag{3.22}$$

we consider $p^r = U_1$ as a virtual input. Taking the following Lyapuov candidate function:

$$V_1 = \frac{1}{2}\|q^r\|^2 \tag{3.23}$$

then with $U_1 = -K_1 q^r$, the derivative of the Lyapunov function is:

$$\dot{V}_1 = -K_1\|q^r\|^2 \leq 0 \tag{3.24}$$

then the solutions of the subsystem (3.22) are asymptotically stable.

Now, considering the second subsystem, without taking in consideration the expression of the control law u^r and the perturbations:

$$\dot{p^r} = u^r \tag{3.25}$$

Taking the change of variables

$$\mathscr{Z} = p^r - U_1 \tag{3.26}$$

and applying it to the two subsystems in (3.22) and (3.25), then we get the following system:

$$\begin{aligned} \dot{q^r} &= -K_1 q^r + \mathscr{Z} \\ \dot{\mathscr{Z}} &= u^r - K_1^2 q^r + K_1 \mathscr{Z} \end{aligned} \tag{3.27}$$

Now, let $H(q^r, \mathscr{Z})$ be a Lyapunov function defined as follows:

$$H(q^r, \mathscr{Z}) = \frac{1}{2}\|q^r\|^2 + \frac{1}{2}\|\mathscr{Z}\|^2 + K_p V(q^r) \tag{3.28}$$

The time derivative of H gives:

$$\begin{aligned} \dot{H} = \; & -K_1\|q^r\|^2 + q^{rT}\mathscr{Z} + \mathscr{Z}^T u^r \\ & -K_1^2 \mathscr{Z}^T q^r + K_1\|\mathscr{Z}\|^2 \\ & -K_p K_1 \nabla_{q^r} V(q^r) q^r + K_p \nabla_{q^r} V(q^r) \mathscr{Z} \end{aligned} \tag{3.29}$$

choosing u^r such that:

$$\begin{aligned} u^r = \; & -K_p \nabla_{q^r} V(q^r) - K_d \hat{L}\, \mathscr{Z} \\ & -(1 - K_1^2) q^r - (K_1 + K_2)\mathscr{Z} \end{aligned} \tag{3.30}$$

we get :

$$\dot{H} = -K_1\|q^r\|^2 - K_2\|\mathscr{Z}\|^2 - K_p K_1 \nabla_{q^r} V(q^r) q^r - K_d\, \mathscr{Z}^T \hat{L}\, \mathscr{Z} \tag{3.31}$$

Now, in the presence of perturbations, that is, $u^r \equiv u^r + \delta^r$, we get:

$$\begin{aligned} \dot{H} = \; & -K_1\|q^r\|^2 - K_2\|\mathscr{Z}\|^2 - K_p K_1 \nabla_{q^r} V(q^r) q^r \\ & -K_d\, \mathscr{Z}^T \hat{L}\, \mathscr{Z} + \mathscr{Z}^T \delta^r \end{aligned} \tag{3.32}$$

since $\hat{L}$ is positive semi-definite, then

$$\begin{aligned} \dot{H} \leq \; & -K_1\|q^r\|^2 - K_2\|\mathscr{Z}\|^2 \\ & +K_p K_1 \|\nabla_{q^r} V(q^r)\| \|q^r\| + \|\mathscr{Z}\|\kappa \end{aligned} \tag{3.33}$$

with κ is the upper bound of the perturbation vector δ. Then, for $0 < b < 1$ and $0 < h < 1$ we can write:

$$\begin{aligned}\dot{H} \leq \; & -(1-b)K_1\|q^r\|^2 - bK_1\|q^r\|^2 \\ & -(1-h)K_2\|\mathscr{Z}\|^2 - hK_2\|\mathscr{Z}\|^2 \\ & +K_pK_1\|\nabla_{q^r}V(q^r)\|\|q^r\| + \|\mathscr{Z}\|\kappa\end{aligned} \tag{3.34}$$

Therefore,

$$\begin{aligned}\dot{H} \leq \; & -(1-b)\beta\|q^r\|^2 - (1-h)K_2\|\mathscr{Z}\|^2 \\ & \forall\|q^r\| \geq \frac{K_p\|\nabla_{q^r}V(q^r)\|}{b},\ \forall\|\mathscr{Z}\| \geq \frac{\kappa}{hK_2}\end{aligned} \tag{3.35}$$

Using the expression of $\mathscr{Z}$ in (3.26) and replacing it in the control law in (3.30) we get the control law used in the collective dynamics (3.18) with:

$$\begin{aligned}K_p^{'} &= K_dK_1 \\ c_1 &= K_1K_2 + 1 \\ c_2 &= K_1 + K_2\end{aligned} \tag{3.36}$$

Moreover, since H is a decreasing function (as confirmed by 3.35), then $H \leq H_0 = H(q^r(0), \mathscr{Z}(0))$, with $(q^r(0), \mathscr{Z}(0))$ are finite initial conditions that respect the conditions in (3.35). Therefore, we can find

$$H(q^r, \mathscr{Z}) \leq c_4\|\Lambda\|^2 \tag{3.37}$$

with $\Lambda = [q^{rT}\ \mathscr{Z}^T]^T$, and c_4 is a positive constant and $c_4\|\Lambda\|^2$ is a class $\mathcal{K}$ function verifying that $c_4\|\Lambda\|^2 \geq H_0$.

In addition, from (3.28), we have:

$$H(q^r, \mathscr{Z}) \geq \frac{1}{2}\|q^r\|^2 + \frac{1}{2}\|\mathscr{Z}\|^2$$

so we can find:

$$c_3\|\Lambda\|^2 \leq H(q^r, \mathscr{Z}) \tag{3.38}$$

with $c_3\|\Lambda\|^2$ is a class $\mathcal{K}$ function and c_3 is a positive constant. From (3.37) and (3.38) we have:

$$c_3\|\Lambda\|^2 \leq H(q^r, \mathscr{Z}) \leq c_4\|\Lambda\|^2 \tag{3.39}$$

Therefore, using the inequalities (3.39), (3.35) and the theorem (3.1) of boundedness and ultimate boundedness, we infer that the solution of the multiple-UAV dynamical system (3.18) is bounded. □

We emphasize here that our proposed control law, in the presence of perturba-

tions, ensures a bounded solution of the multiple-UAV system. In other word, the solution of multiple-UAV system will converge to a "*quasi-α-Lattice*" (cf. equation (1.3)).

3.4 Flocking with distributed integral control

In the previous section, we proposed a modified version of Olfati-Saber flocking control law. This control law was tested in a real-time experimental setup and showed good results, as it will be shown in chapter 4. However, real-time applications always suffer of perturbations that could not be eliminated easily, and need more effective control laws. Perturbations are generally caused by the wind flown from the rotors of quadrotors and by unmodeled dynamics. In real-time experiments, we noticed steady-state errors in the distances between quadrotors. Therefore, in this section, we present an alternative version of (3.15), intended to eliminate the steady-state errors.

In control theory, one of the ways used to eliminate steady-state errors is to add an integral action to the control law [Khalil, 2002]. For ordinary systems, we start by augmenting the system equations with a state that represents the integral of regulation error. However, for our system of systems, we deal with the regulation of the relative interdistance vectors with several neighboring UAVs, which renders the introduction of integral actions challenging.

To overcome this problem, we define the distributed regulation error of a single UAV as follows:

$$e_i = \sum_{j \in N_i} \Phi_\alpha(\|q_j - q_i\|_\sigma)\ \mathbf{n}_{ij} \tag{3.40}$$

Then, we augment the single UAV flocking dynamics in (3.1) by the following equation:

$$\dot{\vartheta} = e_i \tag{3.41}$$

Therefore, the overall augmented system will be written as follows:

$$\begin{aligned} \dot{\vartheta} &= e_i \\ \dot{q}_i &= p_i \\ \dot{p}_i &= u_i \end{aligned} \tag{3.42}$$

Therefore, our control law will be written as follows:

$$
\begin{aligned}
u_i = \sum_{j \in N_i} \Big[& K_p \Phi_\alpha(\|q_j - q_i\|_\sigma)\mathbf{n}_{ij} + K_p^{'} a_{ij}(q)(q_j - q_i) \\
& + K_d a_{ij}(q)(p_j - p_i)\Big] + K_i \int e_i \, \mathrm{d}t \\
& + f_i^\gamma(q_i, p_i, q_r, p_r) + \dot{p}_r
\end{aligned}
\tag{3.43}
$$

where K_i is the integral-action gain, which is tuned by the user. This alternative control law shows good results in real-time experiments, which complements *Olfati-Saber* approach. In fact, the steady-state errors are efficiently reduced, and then the solutions of the multiple-UAV system converges to an "α-*Lattice*".

3.5 Robust Flocking

In this section, we consider the nonlinear model of quadrotor with perturbations in the translational dynamics. Our control method in this section is decomposed in two parts. The first part consists of introducing a control law that linearizes the translational dynamics of quadrotors and decouples their control inputs. In the second part, we propose a flocking control law on the $x - y$ dynamics that is robust to perturbations. This control law is based on the Lyapunov redesign method [Khalil, 2002].

3.5.1 Linearizing and Decoupling control

The quadrotor nonlinear model, with perturbations on the translational dynamics, is given as follows:

$$\dot{\xi} = \nu \tag{3.44a}$$

$$m\ddot{\xi} = G + \mathbf{R}(U + \Delta) \tag{3.44b}$$

$$\dot{\eta} = W\Omega \tag{3.44c}$$

$$J\dot{\Omega} = -\Omega \times J\Omega + \tau \tag{3.44d}$$

$\Delta \in \mathbb{R}^3$ is a perturbation vector which could represent model uncertainties or exogenous perturbations. In this work we assume that the perturbation is bounded with $\|\Delta\| < \mu$.

The quadrotor nominal translational dynamics, that is, without perturbations is given as follows:

$$\begin{aligned}
\dot{x} &= \nu_x \\
\dot{y} &= \nu_y \\
\dot{z} &= \nu_z \\
\ddot{x} &= (c_\phi s_\theta c_\psi + s_\phi s_\psi)(F/m) \\
\ddot{y} &= (c_\phi s_\theta s_\psi - s_\phi c_\psi)(F/m) \\
\ddot{z} &= -g + c_\phi c_\theta (F/m)
\end{aligned} \tag{3.45}$$

To simplify the control of this part, we consider $\phi^\circ \equiv \phi$ and $\theta^\circ \equiv \theta$ as intermediate virtual control inputs, which is generally the case in quadrotor control. In addition, ψ is considered as a parameter of the system. To simplify the calculation in the beginning, we consider $\psi = 0$. Then, we apply a transformation of inputs as follows:

$$v_1 = (c_{\phi^\circ} s_{\theta^\circ})(F/m) \tag{3.46a}$$
$$v_2 = (-s_{\phi^\circ})(F/m) \tag{3.46b}$$
$$v_3 = -g + c_{\phi^\circ} c_{\theta^\circ} (F/m) \tag{3.46c}$$

After some computations we can find:

$$F = m\sqrt{v_1^2 + v_2^2 + (v_3^2 + g)^2} \tag{3.47a}$$
$$\phi^\circ = \arcsin\left(\frac{-v_2}{\sqrt{v_1^2 + v_2^2 + (v_3 + g)^2}}\right) \tag{3.47b}$$
$$\theta^\circ = \arctan\left(\frac{v_1}{v_3 + g}\right) \tag{3.47c}$$

Now, if we consider $\psi \neq 0$, the last three equations in (3.45) could be written as:

$$\begin{aligned}
\ddot{x} &= v_1 c_\psi - v_2 s_\psi \\
\ddot{y} &= v_1 s_\psi + v_2 c_\psi \\
\ddot{z} &= v_3
\end{aligned} \tag{3.48}$$

By applying another transformation of the new inputs, we choose:

$$v_1^* = v_1 c_\psi - v_2 s_\psi \tag{3.49a}$$
$$v_2^* = v_1 s_\psi + v_2 c_\psi \tag{3.49b}$$
$$v_3^* = v_3 \tag{3.49c}$$

and we can find:

$$v_1 = v_1^* c_\psi + v_2^* s_\psi \tag{3.50a}$$
$$v_2 = -v_1^* s_\psi + v_2^* c_\psi \tag{3.50b}$$
$$v_3 = v_3^* \tag{3.50c}$$

Finally, the translational system is completely decoupled and we can write it as follows:

$$\dot{x} = \nu_x \tag{3.51a}$$
$$\dot{y} = \nu_y \tag{3.51b}$$
$$\dot{z} = \nu_z \tag{3.51c}$$
$$\ddot{x} = v_1^* \tag{3.51d}$$
$$\ddot{y} = v_2^* \tag{3.51e}$$
$$\ddot{z} = v_3^* \tag{3.51f}$$

Figure 3.1 shows the architecture of quadrotor control with decoupled translational inputs. In this figure, we have $F = f_F(v_1, v_2, v_3)$ as in equation (3.47a), $f(v_1, v_2, v_3)$ represents the values of ϕ° and θ° in equations (3.47b) and (3.47c) respectively, while the function $f_v(v_1^*, v_2^*, v_3^*)$ is given by the equations in (3.50).

Concerning the quadrotor rotational dynamics, according to [Kendoul et al., 2007] and [Sanahuja, 2010], we can use the feedback linearization technique to decouple the control inputs. The rotational dynamics could then be written as follows:

$$\ddot{\phi} = \tilde{\tau}_\phi \tag{3.52a}$$
$$\ddot{\theta} = \tilde{\tau}_\theta \tag{3.52b}$$
$$\ddot{\psi} = \tilde{\tau}_\psi \tag{3.52c}$$

3.5.2 Robust control law

To control the UAVs in the flocking perspective, we use our architecture introduced in section (2.6). The internal dynamics, that is, the altitude z and the rotational dynamics, will be controlled as in equations (2.39)- (2.42) with some modifications on the ϕ and θ controller as in [Zavala-Río et al., 2003] and [López-Araujo et al., 2010]. Therefore, the control laws of internal dynamics are given as follows:

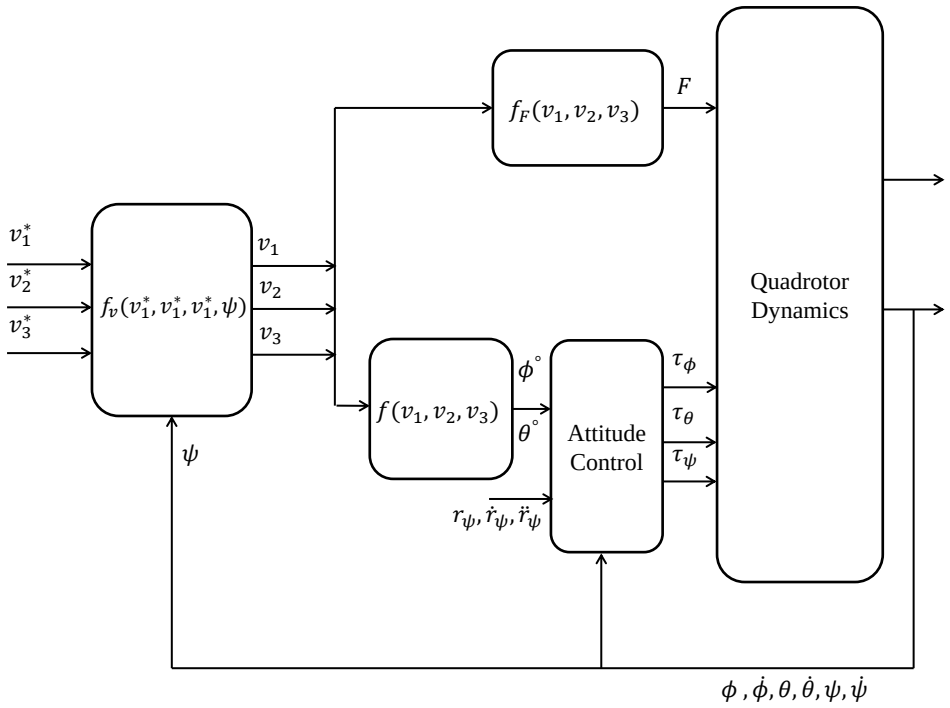


Figure 3.1 – Block diagram of the decoupled translational dynamics of quadrotor

$$v_3^* = \ddot{r}_z + k_{pz}(r_z - z) + k_{dz}(\dot{r}_z - \dot{z}) + k_{iz}\int (r_z - z)\,\mathrm{d}t \tag{3.53}$$

$$\tilde{\tau}_\psi = \ddot{r}_\psi + k_{p\psi}(r_\psi - \psi) + k_{d\psi}(\dot{r}_\psi - \dot{\psi}) + k_{i\psi}\int (r_\psi - \psi)\,\mathrm{d}t \tag{3.54}$$

$$\begin{aligned}\tilde{\tau}_\phi = \;& Sat_{\phi_1}(\ddot{\phi}^\circ) - Sat_{\phi_2}\Big(k_{\phi_1}\dot{\phi} - Sat_{\phi_3}(\dot{\phi}^\circ) \\ & + Sat_{\phi_4}\Big(k_{\phi_2}\dot{\phi} - Sat_{\phi_5}(\dot{\phi}^\circ) + k_{\phi_1}k_{\phi_2}e_\phi\Big)\Big)\end{aligned} \tag{3.55}$$

$$\begin{aligned}\tilde{\tau}_\theta = \;& Sat_{\theta_1}(\ddot{\theta}^\circ) - Sat_{\theta_2}\Big(k_{\theta_1}\dot{\theta} - Sat_{\theta}(\dot{\theta}^\circ) \\ & + Sat_{\theta_4}\Big(k_{\theta_2}\dot{\theta} - Sat_{\theta_5}(\dot{\theta}^\circ) + k_{\theta_1}k_{\theta_2}e_\theta\Big)\Big)\end{aligned} \tag{3.56}$$

with $e_\phi = \phi - \phi^\circ$ and $e_\theta = \theta - \theta^\circ$

We assume here, that the controller in (3.53) is capable to reject the perturbation of the altitude dynamics z. This assumption could be explained by the fact that the altitude reference is constant and we use a controller with integral action which makes it robust to perturbations [Khalil, 2002].

The design of the $x - y$ controllers will be as follows. In equations ((3.51a), (3.51b), (3.51d), (3.51e)), we take $q_i = [x \;\; y]^T$, $p_i = [\nu_x \;\; \nu_y]^T$, and $u_i = [v_1^* \;\; v_2^*]^T$.

Moreover, we take into consideration the perturbation vector in the $x-y$ directions. Therefore, the translational dynamics could be written as follows:

$$\begin{aligned} \dot{q}_i &= p_i \\ \dot{p}_i &= u_i + \Delta_{xy_i} \end{aligned} \tag{3.57}$$

with $\Delta_{xy_i} \in \mathbb{R}^f$ being the perturbation vector.

The representation of perturbations in a single system could be given by an unknown bounded function with a known bound. However, in a system of systems, another challenge arises in order to represent perturbations. In fact, the presence of interactions between UAVs in the multiple-UAV system of systems causes a propagation of perturbations. In other word, if we have a perturbation in a single UAV, its effect will be propagated as a wave toward the other UAVs in the system. To model such phenomenon, we define the perturbation term in (3.57) as follows:

$$\Delta_{xy_i} = \Xi_i + \sum_{j \in N_i} a_{ij}(q)\, \Xi_j \tag{3.58}$$

with Ξ_i being a local and a direct perturbation on the UAV i, and Ξ_j being a perturbation propagated from a neighboring UAV j.

Assumption 3.2. *Every perturbation term Ξ_i $\forall i \in \mathcal{V}$ is an unknown function, but it has a known upper bound such that, $\|\Xi_i\| \leq \mu_i$ and $\mu_i > 0$.*

Lemma 3.2. *If assumption (3.2) holds, then the overall perturbation Δ_{xy_i} is also bounded.*

Proof. The proof of this lemma is obvious, since from (3.58), Δ_{xy_i} is the sum of bounded perturbations terms. Then we can infer that $\|\Delta_{xy_i}\| \leq \mu_{xy_i}$ with

$$\mu_{xy_i} = \mu_i + \sum_{j \in N_i} \mu_j \tag{3.59}$$

if we assume that all the perturbation terms have the same upper bound, that is, $\mu_i = \mu_j$ $\forall i, j \in \mathcal{V}$, then we get:

$$\mu_{xy_i} = (|N_i| + 1)\, \mu_i \tag{3.60}$$

with $|N_i|$ being the cardinality of the neighboring set N_i. □

Now we are ready to present our distributed control law, applied on each UAV, that ensures the flocking and the navigation of multiple UAVs, as well as the stability of the flock in the presence of perturbations, as follows:

$$\begin{aligned} u_i = \sum_{j\in N_i} \Big[& K_p \Phi_\alpha(\|q_j - q_i\|_\sigma)\mathbf{n}_{ij} \\ & + K_d a_{ij}(q)(p_j - p_i)\Big] \\ & + \beta_i + f_i^\gamma(q_i, p_i, q_r, p_r) + \dot{p}_r \end{aligned} \tag{3.61}$$

with β_i being the Lyapunov redesign control term defined as follows:

$$\beta_i = -\mu_{xy_i}\frac{p_i}{\|p_i\|} \tag{3.62}$$

To prove the efficiency of the control law in (3.61), we start by writing the collective dynamics of all the multiple-UAV system in the moving frame, as introduced in (3.3). Then the collective dynamics will be written as follows:

$$\begin{aligned} \dot{q}^r &= p^r \\ \dot{p}^r &= -K_p \nabla_{q_i^r} V(q^r) - K_d \hat{L}\, p^r - c_1 q^r - c_2 p^r + \beta^r + \Delta_{xy}^r \end{aligned} \tag{3.63}$$

with $\beta^r = col(\beta_1^r, \ldots, \beta_M^r)$ and $\Delta_{xy}^r = col(\Delta_{xy_1}^r, \ldots, \Delta_{xy_M}^r)$.

Now we are ready to present our proposition.

Proposition 3.2. *Consider the multiple-UAV dynamical system in (3.63) where every UAV applies the control law in (3.61). Suppose that the assumption (3.2) holds and taking in consideration lemma (3.2), then the solutions of the multiple-UAV dynamical system converge asymptotically to an $\alpha-$lattice.*

Proof. Consider the Lyapunov function $H(q^r, p^r)$ defined as:

$$H(q^r, p^r) = K_p V(q^r) + \frac{1}{2}\sum_{i\in\mathcal{V}} \|p_i^r\|^2 + \frac{c_1}{2} q^{rT} q^r \tag{3.64}$$

Taking the time derivative of H we get:

$$\dot{H} = K_p \nabla_{q^r} V(q^r) p^r + p^{rT}\dot{p}^r + c_1 q^{rT} p^r \tag{3.65}$$

By replacing $\dot{p}^r$ by its expression in (3.63), we get:

$$\dot{H} = -K_d\, p^{rT}\hat{L} p^r - p^{rT} c_2 p^r + p^{rT}\beta^r + p^{rT}\Delta_{xy}^r \tag{3.66}$$

In the previous equation we have:

$$\begin{aligned} p^{rT}\beta^r + p^{rT}\Delta_{xy}^r &= \sum_{i\in\mathcal{V}} (p_i^{rT}\beta_i^r + p_i^{rT}\Delta_{xy_i}^r) \\ &\leq \sum_{i\in\mathcal{V}} (p_i^{rT}\beta_i^r + \|p_i^r\|\mu_{xy_i}) \end{aligned}$$

replacing β_i^r by its expression in (3.62), we find that

$$p^{r^T}\beta^r + p^{r^T}\Delta^r_{xy} \leq 0$$

We infer that $\dot{H} < 0 \; \forall p^r \neq 0$. Taking $\Omega_c = \{\Lambda : H(\Lambda) \leq c\}$ as a level set of H. Following the proof in [Olfati-Saber, 2006], from LaSalle's invariance principle, every solution starting in Ω_c converges to the largest invariant set in $O = \{\Lambda \in \Omega_c : \dot{H} = 0\}$. For $\dot{H} = 0$, we have $p^r = 0$, hence, every solution of the system (3.63) asymptotically converge to the equilibrium configuration $\Lambda^* = (q^{r*}, \mathbf{0})$ with q^{r*} being the minima of $\frac{1}{2}K_pV(q^r) + \frac{c_1}{2}q^{r^T}q^r$, that is, an $\alpha-$*lattice.* □

The proposed control law in (3.61), is based on the Lyapunov redesign method. The β_i term in this controller reveals an important theoretical and practical problems since it is a discontinuous function of p_i [Khalil, 2002]. In one hand, the division by zero, the existence and the uniqueness of solutions are some theoretical problems with this kind of control. In the other hand, the problem of chattering is one of the main practical drawbacks of such discontinuous controller.

A practical solution of these problems is to relax the definition of β_i to be continuous and defined at zero. Thus, we redefine β_i in (3.62) as follows:

$$\beta_i = \begin{cases} -\mu_{xy_i} \dfrac{p_i}{\|p_i\|} & \text{if } \mu_{xy_i}\|p_i\| \geq \varsigma \\ \\ -\mu^2_{xy_i} \dfrac{p_i}{\varepsilon} & \text{if } \mu_{xy_i}\|p_i\| < \varsigma \end{cases} \tag{3.67}$$

with $\varsigma > 0$.

What we can get from this control law is a multiple-UAV dynamical system with bounded solutions. This result is shown in the following proposition.

Proposition 3.3. *Consider the multiple-UAV dynamical system in (3.63) where every UAV applies the control law in (3.67). Suppose that the assumption (3.2) holds and taking in consideration lemma (3.2), then the solution of the multiple-UAV dynamical system is bounded.*

Proof. Taking again the Lyapunov function $H(q^r, p^r)$, proposed in (3.64), and its time derivative:

$$H(q^r, p^r) = K_pV(q^r) + \frac{1}{2}\sum_{i\in\mathcal{V}} \|p^r_i\|^2 + \frac{c_1}{2}q^{r^T}q^r$$

$$\dot{H} = -K_d\, p^{r^T}\hat{L}p^r - p^{r^T}c_2p^r + p^{r^T}\beta^r + p^{r^T}\Delta^r_{xy} \tag{3.68}$$

for $\mu_{xy_i}\|p^r_i\| \geq \varsigma$ in β^r_i (represented in the moving frame), we have the same result as in proposition (3.2).

However, for $\mu_{xy_i}\|p_i\| < \varsigma$, we have:

$$p^{r^T}\beta^r + p^{r^T}\Delta^r_{xy} \leq \sum_{i\in\mathcal{V}}(-\mu^2_{xy_i}\ \frac{\|p^r_i\|^2}{\varepsilon} + \mu_{xy_i}\|p^r_i\|) \tag{3.69}$$

Then, $-\mu^2_{xy_i}\ \frac{\|p^r_i\|^2}{\varepsilon} + \mu_{xy_i}\|p^r_i\| \leq 0$ when $\|p^r_i\| \geq \frac{\varsigma}{\mu_{xy_i}}$.

Therefore,

$$\dot{H} < 0,\ \forall\|p^r_i\| \geq \frac{\varsigma}{\mu_{xy_i}} \tag{3.70}$$

Moreover, since H is a decreasing function, then $H \leq H_0 = H(q^r(0), p^r(0))$, with $(q^r(0), p^r(0))$ are finite initial conditions that respect the condition in (3.70). Therefore, we can find

$$H(q^r, p^r) \leq b_4\|\Pi\|^2 \tag{3.71}$$

with b_4 is a positive constant, $\Pi = [q^{r^T}\ p^{r^T}]^T$ and $b_4\|\Pi\|^2$ is a class $\mathcal{K}$ function verifying that $b_4\|\Pi\|^2 \geq H_0$.

In addition, from (3.64), we have:

$$H(q^r, p^r) \geq \frac{1}{2}\sum_{i\in\mathcal{V}}\|p^r_i\|^2 + \frac{c_1}{2}q^{r^T}q^r$$

so we can find:

$$b_3\|\Pi\|^2 \leq H(q^r, p^r) \tag{3.72}$$

with $b_3\|\Pi\|^2$ is a class $\mathcal{K}$ function and b_3 is a positive constant. From (3.71) and (3.72) we have:

$$b_3\|\Pi\|^2 \leq H(q^r, p^r) \leq b_4\|\Pi\|^2 \tag{3.73}$$

Therefore, using the inequalities (3.73), (3.70) and theorem (3.1) we infer that the solution of the multiple-UAV dynamical system (3.63) is bounded. □

When ς is too small, the solutions of our multiple-UAV system will approach a neighborhood of the origin, that is an *α-lattice*. However, the control law in (3.67) will approach the discontinuous control in (3.62) and then we will get the chattering phenomenon. Therefore, by using the relaxed control law in (3.67) with ς not too small, we get a *quasi-α-lattice*.

3.6 Conclusion

In this chapter, we have proposed four control laws to control multiple-UAV system in the flocking perspective. The control laws were based on the work of [Olfati-Saber, 2006]. Our interest was to improve the control algorithm in [Olfati-Saber, 2006] in order to be compatible with the nonlinear dynamics of UAV. During this chapter, we worked on a linear approximation as well as a full-nonlinear model of UAV.

The design of the first two control laws were based on the linearization about the origin of the UAVs dynamics. We then proposed our first modification, on the flocking algorithm proposed in [Olfati-Saber, 2006], by introducing tuning gains. We proved analytically that our controller is robust to perturbations in contrast to the flocking algorithm in [Olfati-Saber, 2006]. Then, we improved our control law by introducing a distributed integral action in order to obtain a zero-steady-state error, that is, an α-*lattice* configuration of multiple UAVs.

In the two other control laws, we worked on a complete nonlinear model of UAV. We started by applying a control law that linearizes the translational dynamics of UAV. Then, we used the Lyapunov redesign technique to design a robust flocking control law. Finally, we enhanced this control law to avoid the chattering phenomenon.

Simulation and Real-Time Results

Contents

4.1 Introduction

In this chapter, we present simulation and experimental results of the control laws proposed in the precedent chapters. For the simulation, we use a PC-based simulator of flock of multiple UAVs. This simulator is developed by *Guillaume Sanahuja*, PhD and research engineer at Heudiasyc laboratory, with the context of ROBOTEX project. For experiments, we show the results of the implementation of our control laws on ArDrone2 quadrotors from Parrot. The UAVs evolved in an indoor environment where an Optitrack motion capture system is used.[1]

4.2 Simulator of Multiple UAVs

As a part of ROBOTEX project, Heudiasyc laboratory is equipped by a fleet of UAVs in order to carry out scientific researches on flight formation control. However, flight formation control could be risky, that why the laboratory developed a simulator of fleet of UAVs. The goal of this simulator is to run on a computer, a code identical to that used in the real UAVs, to perform all the program development steps safely.

[1]The installation, operation, support and maintenance of the experimental platform equipments (quadrotors, Optitrack system and the wireless network) are ensured by the technical assistants team and research engineers of Heudiasyc laboratory.

For this purpose, the PC is run under Linux as in the real UAVs. In the simulator, virtual sensors and actuators are connected to a discrete nonlinear model of UAV as in (1.4). As a result, all UAVs' states are calculated at each instant of time. Each UAV in the fleet simulator is an independent computer process. Moreover, UAVs evolve in a 3-D virtual environment, thanks to Irrlicht engine. The program in the simulator is connected to a Graphical User Interface (GUI) base-station control program. The base station records and draws measurements. Moreover, it is used to start and end simulations and to set the parameters of UAVs and control laws. Figure 4.1 shows the architecture of the simulator.

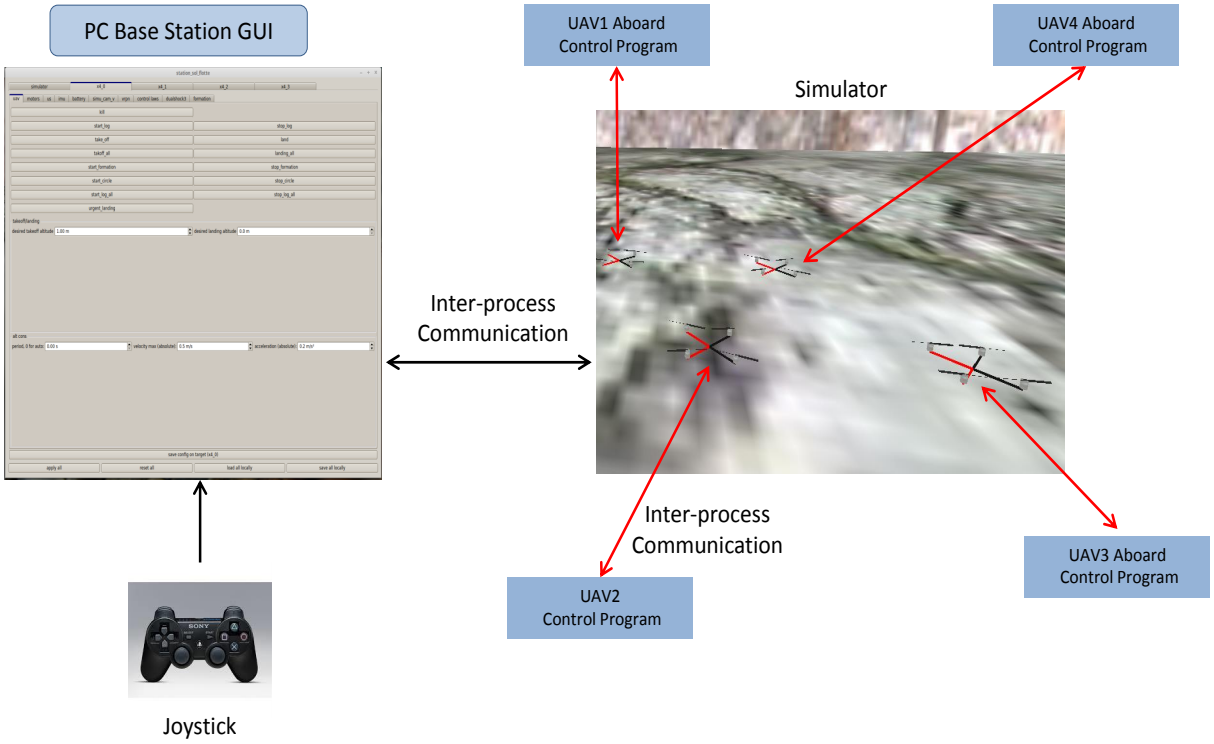


Figure 4.1 – Architecture of the simulator of fleet of UAVs

In the following, we present the base-station GUI, used together in the simulator and in the experimental platform. Figure 4.2 shows the base-station GUI where high-level commands could be sent to one or all UAVs. The same interface is used for real time. Graphs of measured states and useful information such as control laws outputs could be visualized on the GUI, see figure 4.3. GUI is also used to modify parameters and to tune gains on line and send them to the simulated or the real

quadrotors, see figure 4.4.

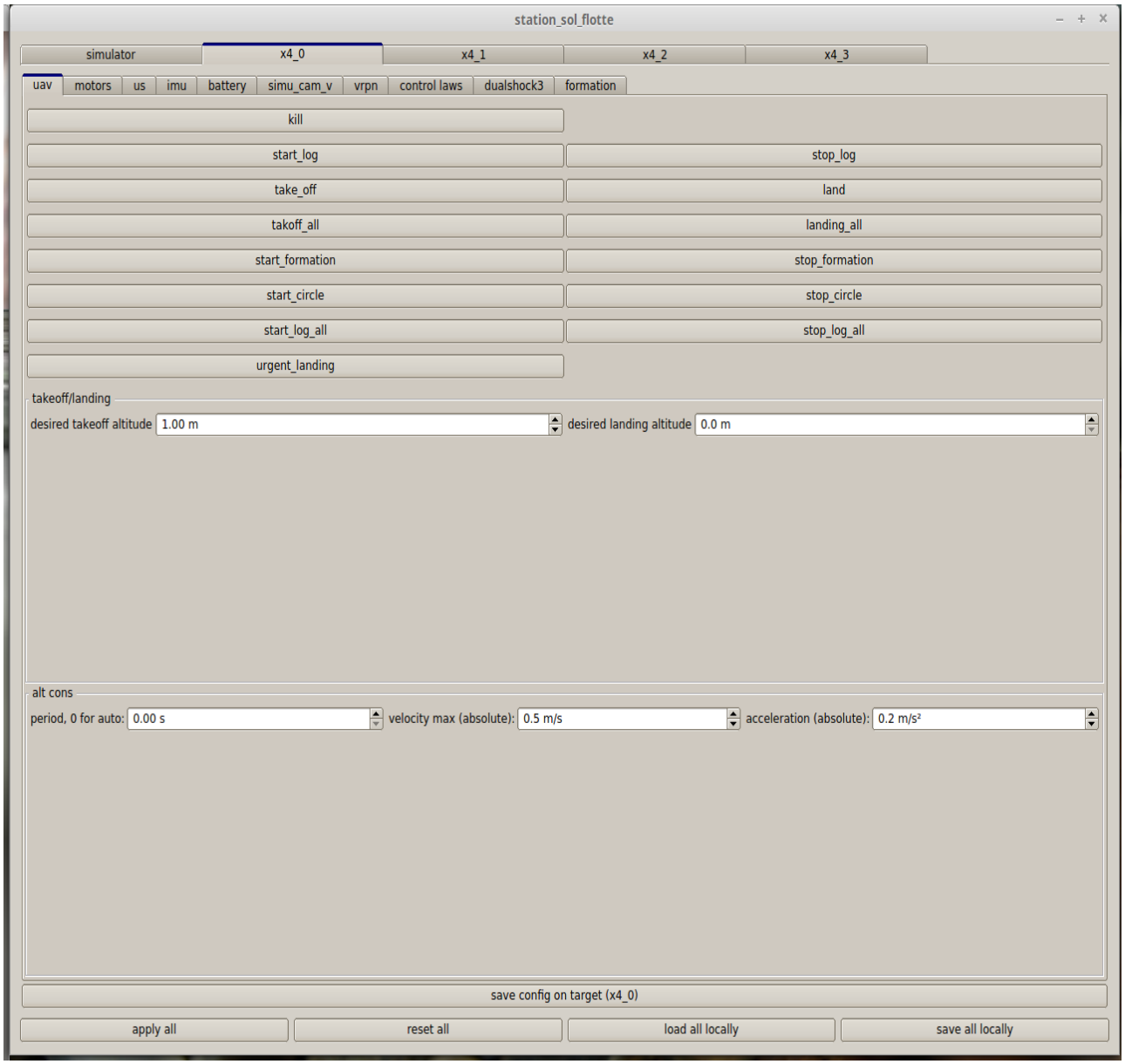


Figure 4.2 – High level commands in the based-station GUI

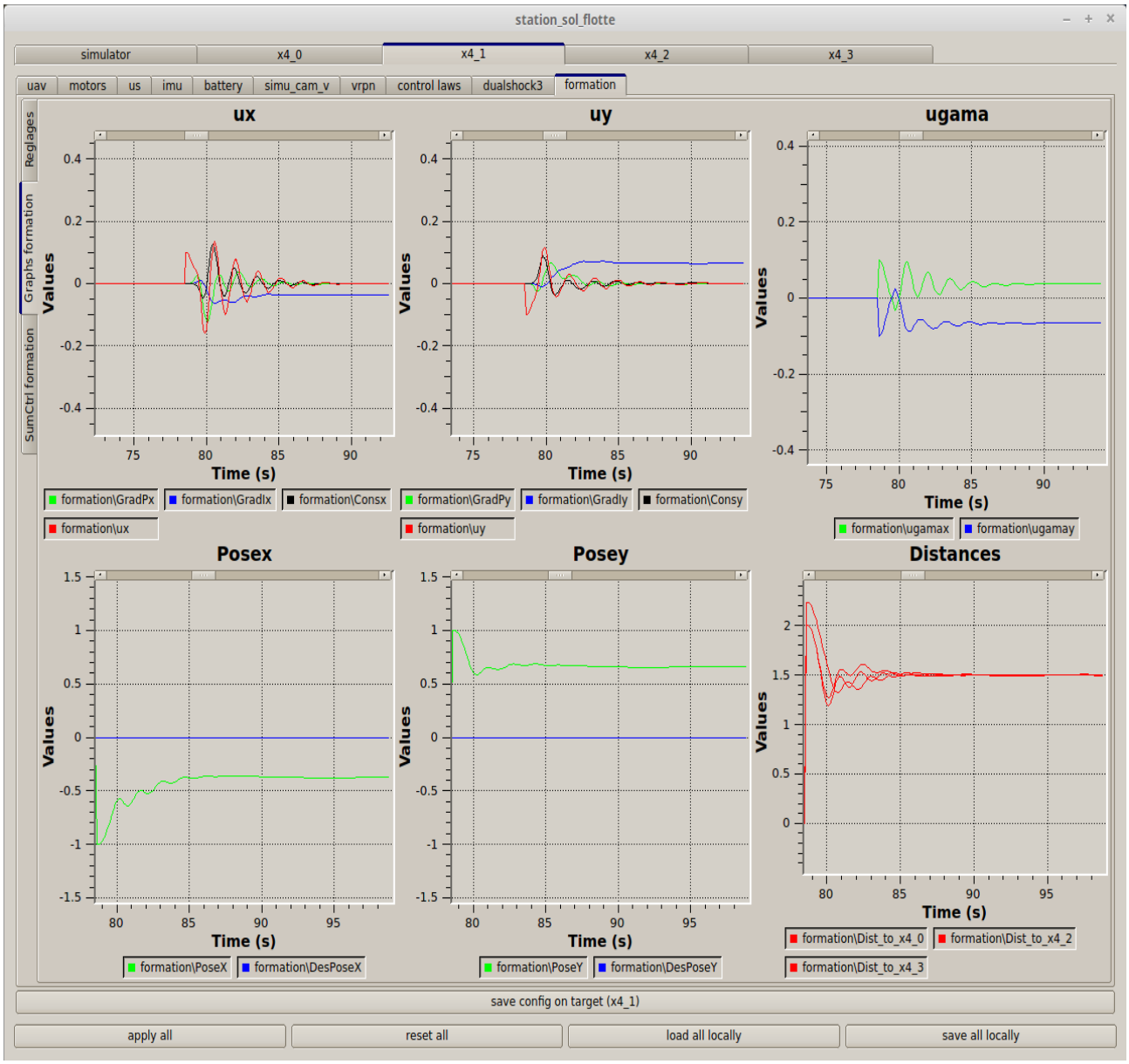


Figure 4.3 – Graphs of measurements in the GUI

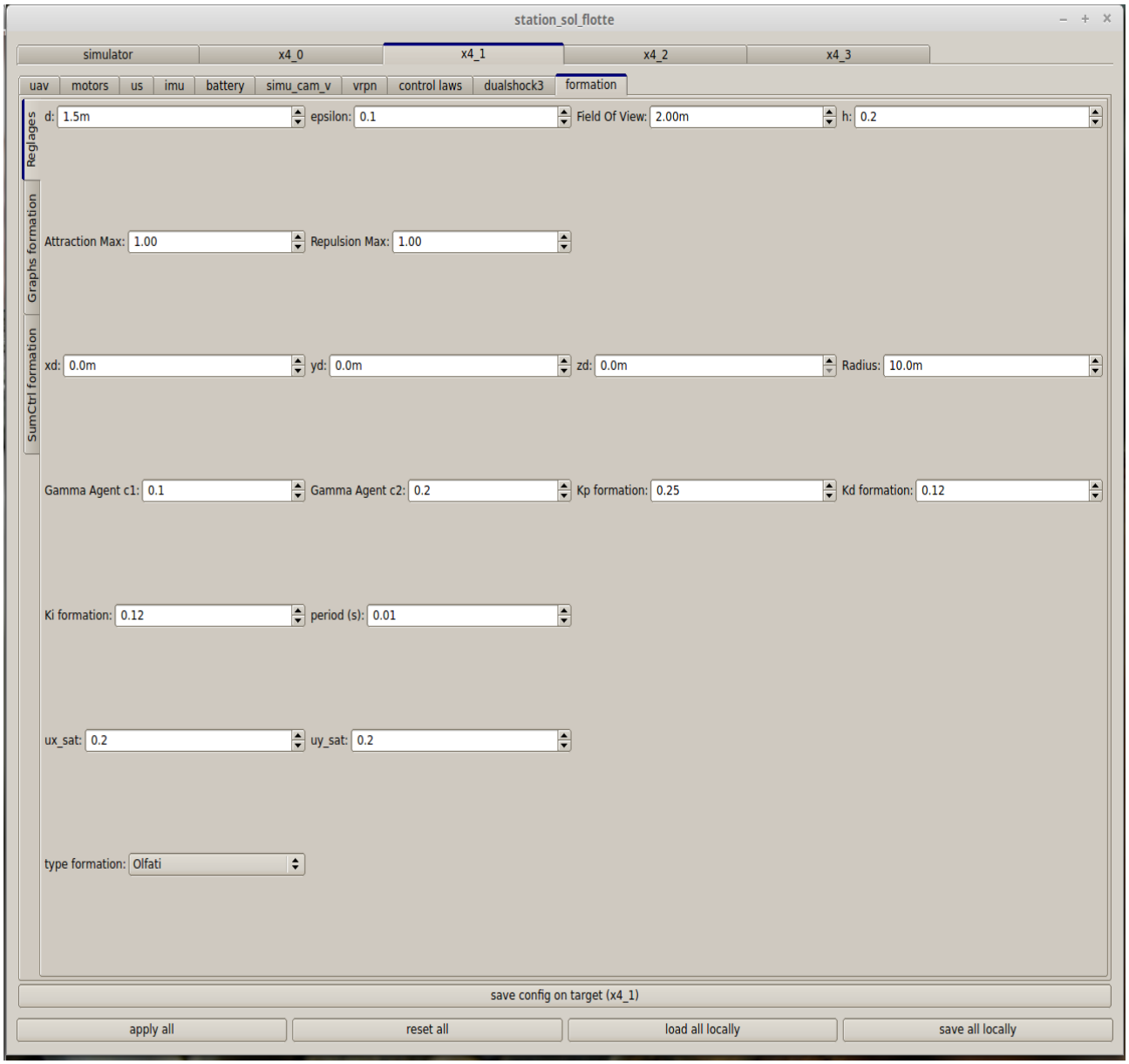


Figure 4.4 – Parameters and gains introduction in the GUI

4.3 Simulation results

In this section, we present different simulation results of our proposed control laws. The simulations are performed in the simulator of the fleet of UAVs, introduced in the previous section. All the tests are performed using a nonlinear model of quadrotors. To show the efficiency of our control laws, different scenarios, such as aggregation and direct or circular navigation of multiple UAVs, are tested with a different number of UAVs. Moreover, we test the scalability, the self-organization and the fault-tolerant properties of our control laws by introducing a sudden accident on a UAV during the flight scenario. This accident consists in a sudden forced urgent landing of a UAV, introduced by the simulator user using the GUI.

In all the scenarios, the UAVs takeoff from their initial positions to a predefined fixed altitude. The yaw angles desired values all over in the scenarios are equal to zero. Then, a flight formation control strategy, that is one of our control laws, is activated and a flight formation scenario is performed. In this work, we have three flight formation scenarios: aggregation, direct navigation and circular navigation.

In the aggregation scenario, the UAVs move and gather around the origin ($x = 0$, $y = 0$). In direct navigation scenario, the UAVs move toward a predefined rendezvous fixed point different from the origin. Finally, the circular navigation consists in UAVs tracking a predefined circular trajectory. In all the scenarios, the predefined trajectory or the rendezvous point is known by all the UAVs.

4.3.1 Flocking by trajectory generation

4.3.1.1 Sum Strategy

In this part, we apply the sum control strategy, introduced in sections (2.5.2) and (2.5.3), on 4 UAVs. First, we show the simulation of 4 UAVs performing an aggregation scenario followed by a circular navigation flight scenarios. Then, we present two separated aggregation and circular navigation scenarios with a simulated urgent landing of one UAV.

Figure 4.5 shows 4 UAVs performing an aggregation and a circular navigation. The UAVs start at their initial position marked by diamonds. Then they aggregate toward the origin to ensure the collision-free formation. After that, the UAVs follow a circular trajectory and then return back to aggregate around the origin. The final positions of UAVs is depicted in the figure by black squares. The overall scenario is performed in a collision-free formation with yaw angles equal to zero.

In order to see clearly the collision-free flight formation, we illustrate in figure 4.6 the interdistances between UAVs during the simulation. The reference desired

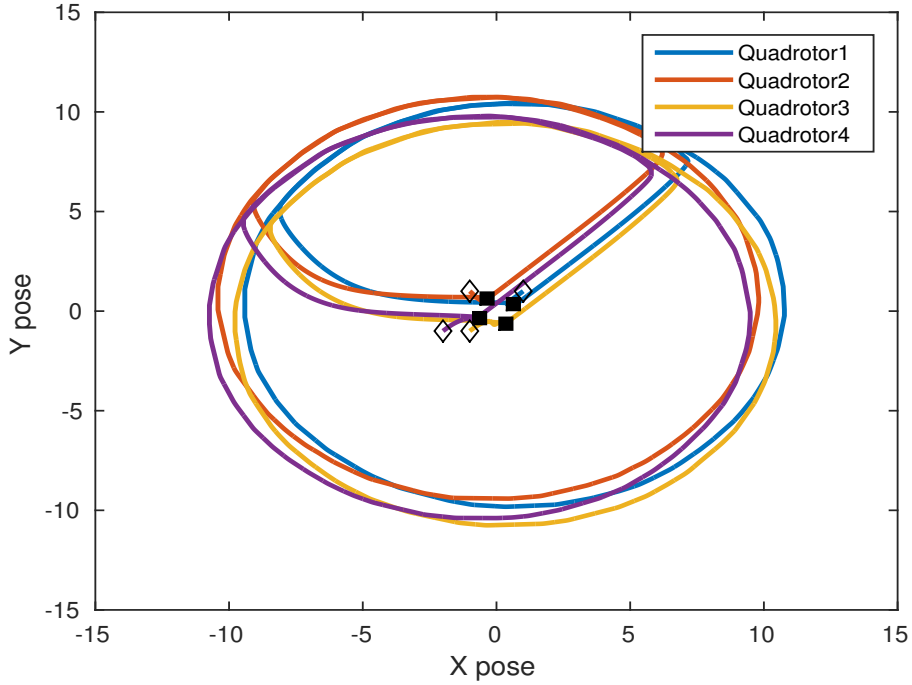


Figure 4.5 – Trajectories of 4 UAVs in aggregation and navigation scenario using Sum strategy

interdistance between every two neighboring UAVs is fixed to $d = 1.5m$. This reference interdistance is depicted by a black line in the figure. As shown in the figure, in the beginning the interdistances are constant, since the UAVs are still in their initial positions. Once the sum control strategy is activated, the interdistances between UAVs converge to the desired value. We notice in the figure that we have some interdistances less than the desired value, for example the distance between UAVs 3 and 4. This result is known in control theory under the name of "steady-state error" and it appears here in the formation control problem. This steady-state error is a drawback of the sum control strategy which will be overcome by another control law that has been proposed and will be presented in the sequel.

In Figure 4.7, we have the result of simulating the aggregation of 4 UAVs while during the simulation, we introduced an urgent landing of UAV 2. At the beginning, the UAVs aggregate toward the origin and constitute a formation with safe interdistances. Then when the UAV 2 lands, the other UAVs reorganize themselves and continue the aggregation separately.

While it is difficult to see the self-organization property in figure 4.7, the sudden interdistances variations between UAVs, in figure 4.8, illustrates more clearly this emergent behavior at time $\approx 44s$.

In figure 4.10, we present the fault-tolerant and the self-organization property of the Sum strategy. 4 UAVs start an aggregation and a circular navigation when

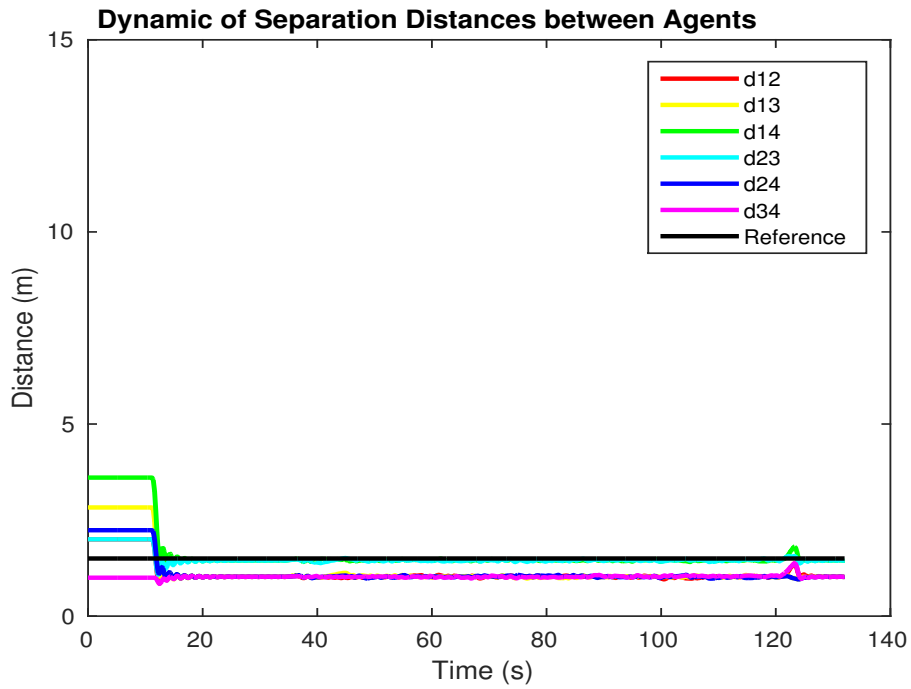


Figure 4.6 – Interdistances between 4 UAVs in aggregation and navigation scenario using Sum strategy

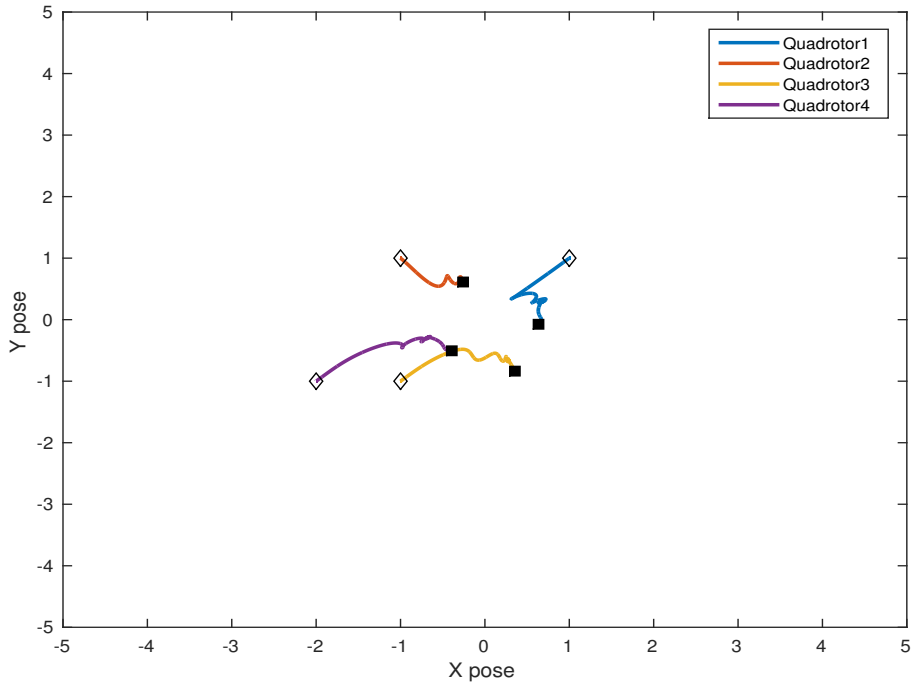


Figure 4.7 – Trajectories of 4 UAVs in an aggregation scenario using Sum control with urgent landing in UAV 2

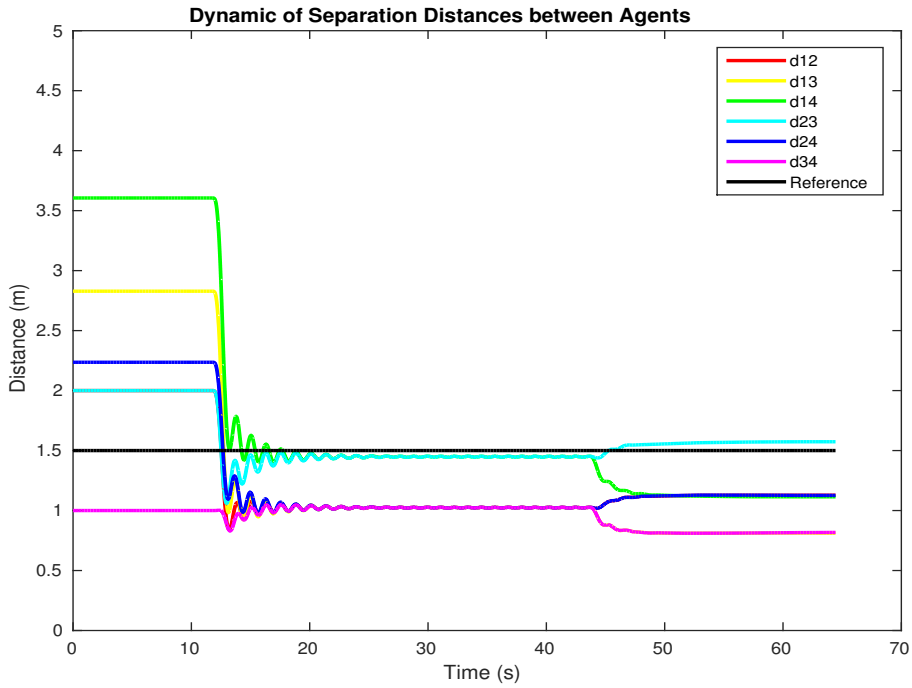


Figure 4.8 – Interdistances between 4 UAVs in an aggregation scenario using Sum control with urgent landing in UAV 2

suddenly the UAV 2 is forced to land. The 3 remaining UAVs continue the navigation and reorganize themselves. We remind here that the initial positions are marked by diamonds and the final positions by black squares.

Figure 4.10 shows clearly the variations of interdistances when the accidental urgent landing happened between 70 and 80 seconds. Since UAV 2 has landed and stopped, the distances between it and the other moving UAVs continue to increase.

4.3.1.2 Average Strategy

In this part, we apply the average control strategy, introduced in sections (2.5.1) and (2.5.3), on 4 UAVs. As in the sum strategy, we show the simulation of 4 UAVs performing an aggregation scenario followed by a circular navigation flight scenarios. Then, we present two separated aggregation and circular navigation scenarios with a simulated urgent landing of one UAV.

In figures 4.11 to 4.16 that show the aggregation and the navigation of 4 UAVs in normal and accidental (with urgent landing) flight, we notice approximately the same results obtained with the sum strategy. The Average strategy succeeded in performing a collision-free formation in the aggregation and the navigation scenarios, figures 4.11 and 4.12. The same drawback of "steady-state error" in interdistances is also noticed in the average strategy. Moreover, in accidental scenarios, the average strategy succeeded in performing self-organized and fault-tolerant aggregation and

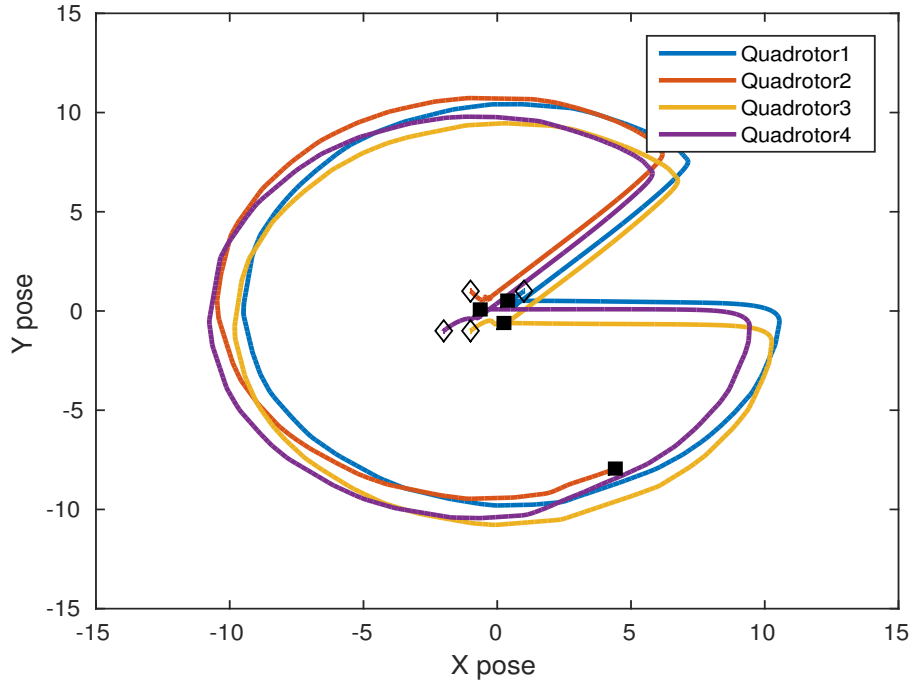


Figure 4.9 – Trajectories of 4 UAVs in a circular navigation scenario using Sum control with urgent landing in UAV 2

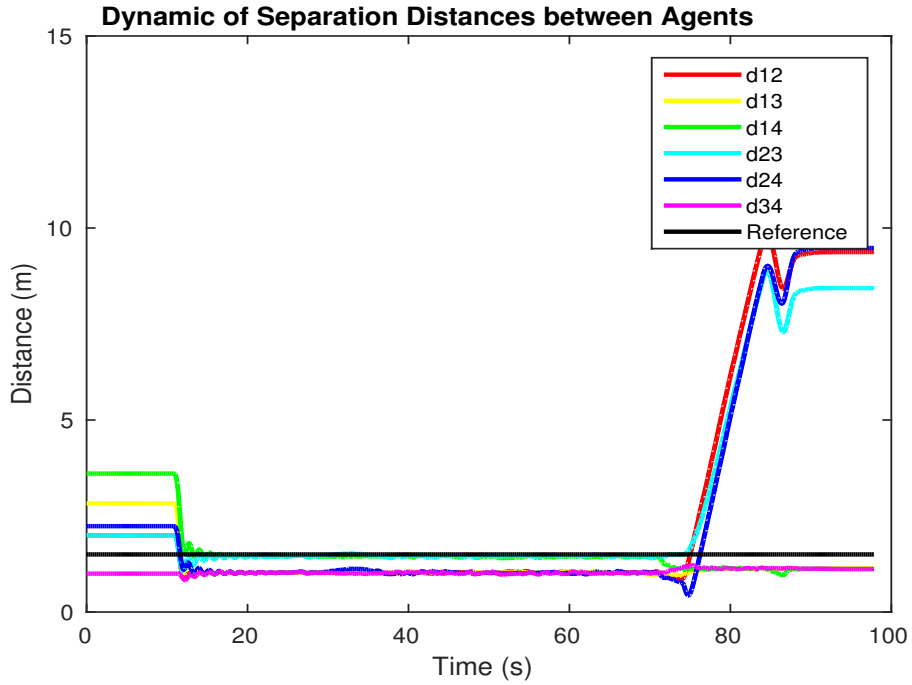


Figure 4.10 – Interdistances between 4 UAVs in a circular navigation scenario using Sum control with urgent landing in UAV 2

navigation.

The only difference that we have noticed when we compared the results of the sum and the average strategy is the smoothness of the UAVs interdistances variation in the average strategy. While in the sum strategy, we can see rapid variations and oscillations when we start the formation, we notice a smoother convergence of interdistances to the desired value in the average strategy (certainly with a steady-state error). This could be explained by the fact that, in the average strategy, each UAV follows an average reference of shifted states (cf. section 2.5) rather than a sum reference. The average operation smooths the large difference between shifted states unlike the sum operation.

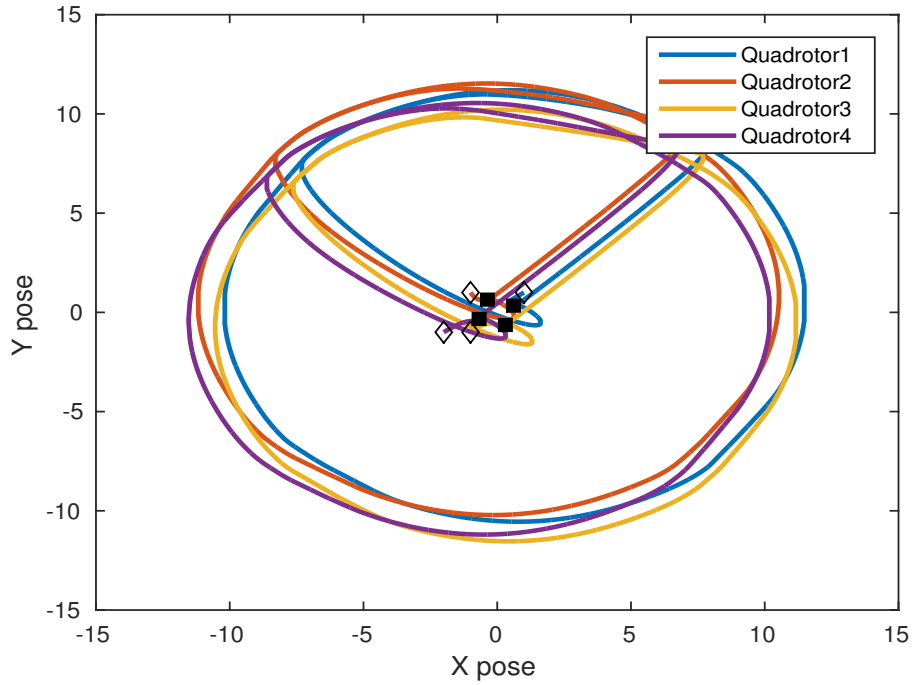


Figure 4.11 – Trajectories of 4 UAVs in aggregation and circular navigation scenario using Average control strategy

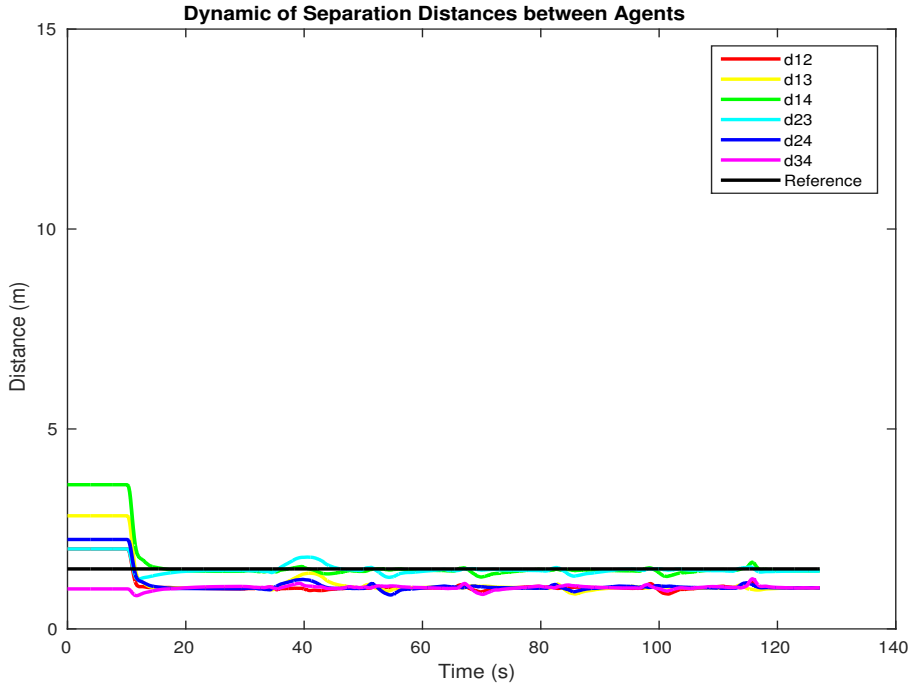


Figure 4.12 – Interdistances between 4 UAVs in aggregation and circular navigation scenario using Average control strategy

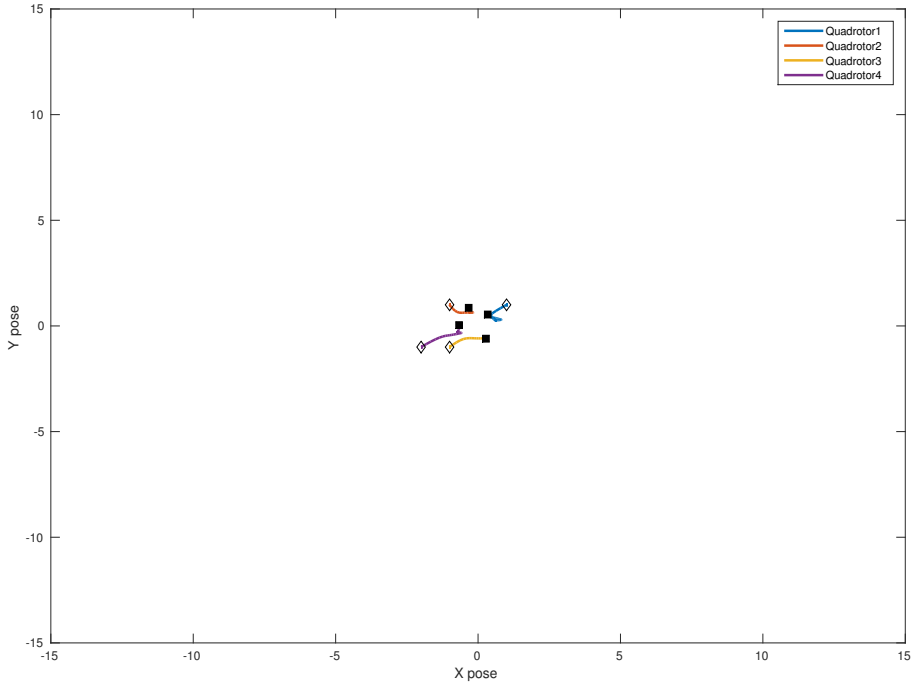


Figure 4.13 – Trajectories of 4 UAVs in an aggregation scenario using Average control with urgent landing in UAV 2

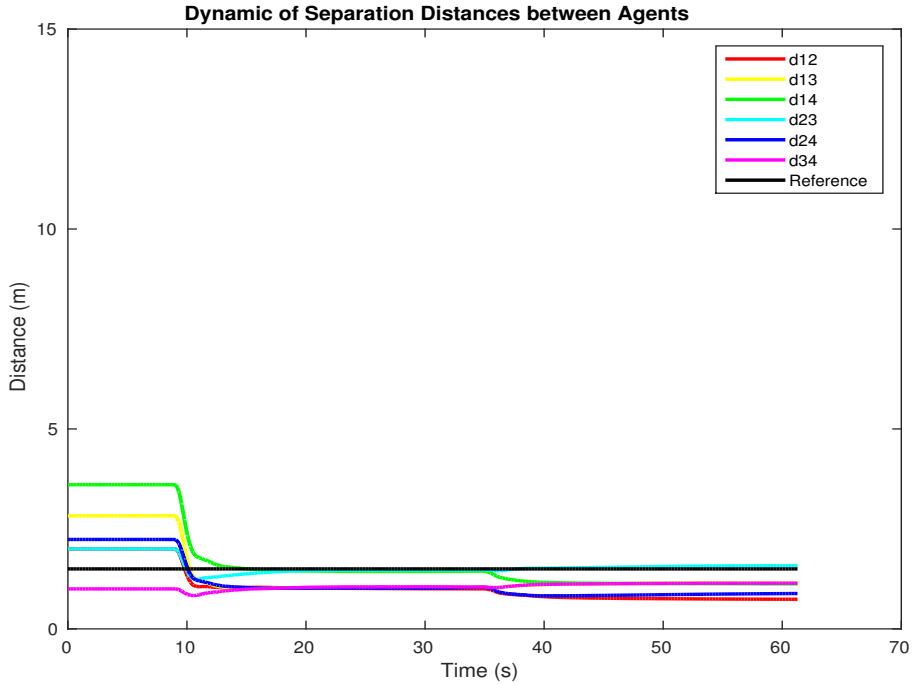


Figure 4.14 – Interdistances between 4 UAVs in an aggregation scenario using Average control with urgent landing in UAV 2

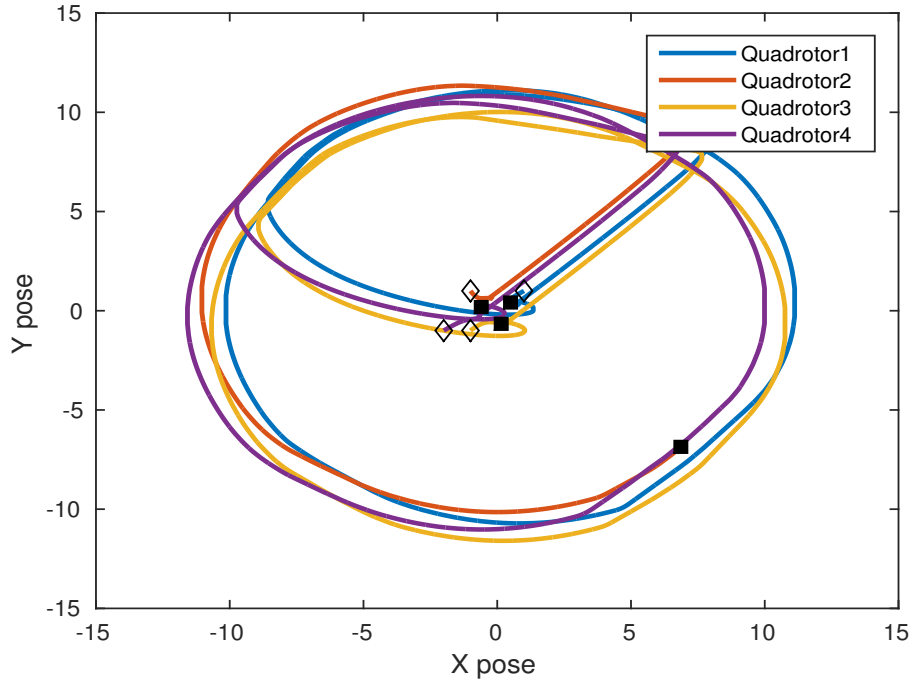


Figure 4.15 – Trajectories of 4 UAVs in a circular navigation scenario using Average control with urgent landing in UAV 2

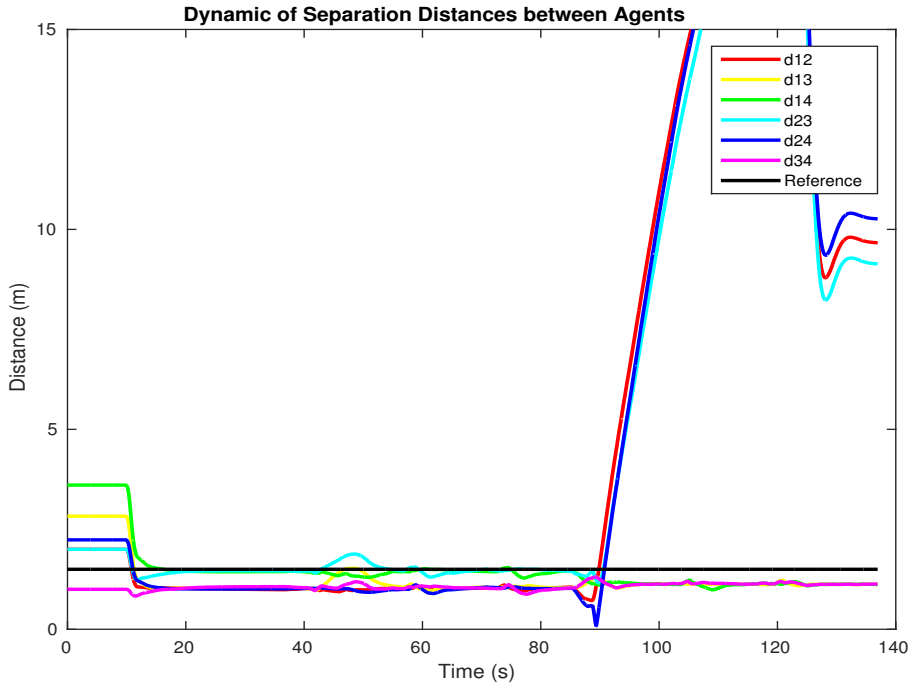


Figure 4.16 – Interdistances between 4 UAVs in a circular navigation scenario using Average control with urgent landing in UAV 2

4.3.2 Consensus-based Flocking

In this section, we show simulation results of multiple UAVs using the flocking control with tuning gains and with distributed integral laws as in (3.15) and (3.43) respectively. In the following simulations, the desired value of interdistances between UAVs is $1.5m$ and the interaction range of each UAV is $2m$. As in the precedent sections, we illustrate several simulations showing the scalability, the precision and the fault-tolerant properties of these control laws.

4.3.2.1 Flocking control with tuning gains

Figure 4.17 shows an aggregation and circular navigation of 6 UAVs using the flocking control with tuning gains. The UAVs start in their initial positions, marked by diamonds, and converge firstly toward the origin. When the formation is stable, a new phase is activated and the UAVs start to track a circular trajectory whilst maintaining the formation. Finally, after a certain time, the UAVs quit the circular trajectory and return back to aggregate around the origin by forming a pentagonal quasi-α lattice pattern, with one UAV at the center.

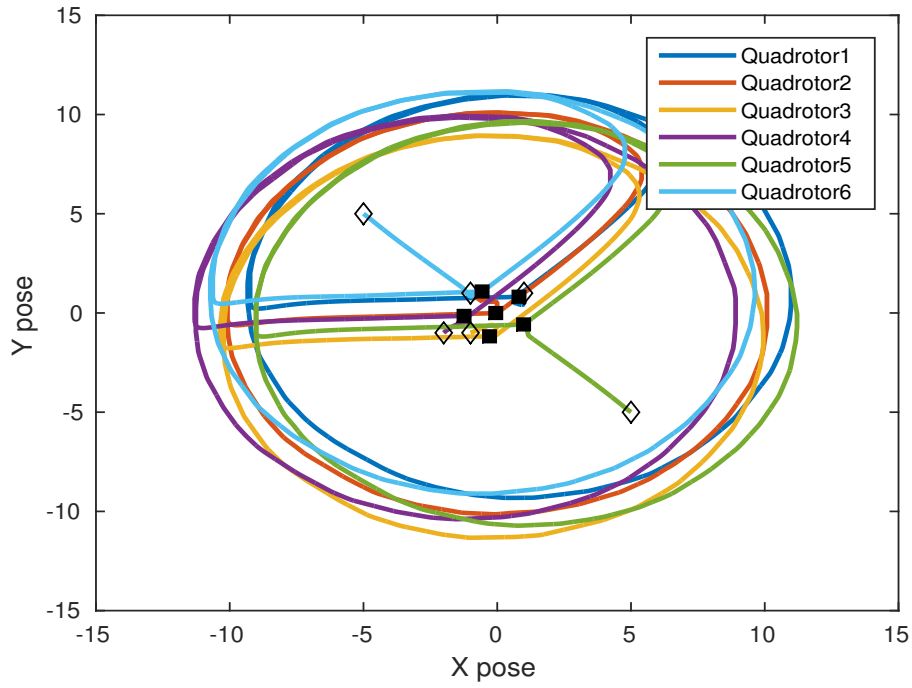


Figure 4.17 – Trajectories of 6 UAVs in aggregation and circular navigation scenario using tuning gains control strategy

The formation of a quasi-α lattice pattern is confirmed in figure 4.17. The distances between neighboring UAVs approach the desired interdistances value $1.5m$. We can notice that the interdistances do not reach completely the desired

reference which means that this control strategy suffers also from a steady-state error drawback.[2] However, the scalability of this control law could be seen in figure 4.17. For example, the distance between UAV 5 and 6 does not converge to $1.5m$, since it is greater than the interaction range $2m$. This means that, using this control law, we can add as much as UAVs to the fleet and the computational power of each UAV will not be affected, since each UAV interacts with a limited number of UAVs in its neighboring range.

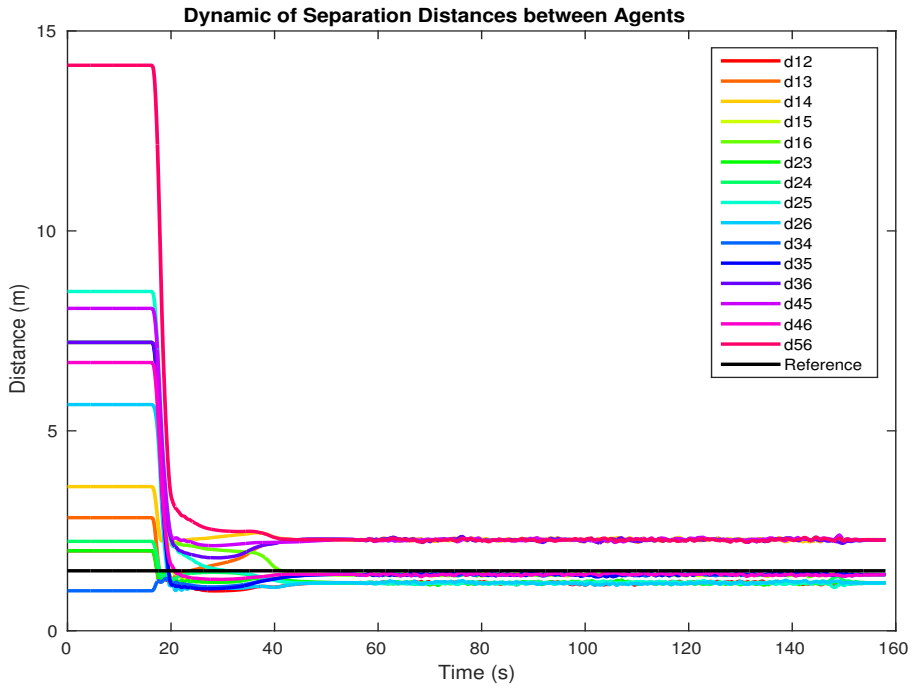


Figure 4.18 – Interdistances between 6 UAVs in aggregation and circular navigation scenario using tuning gains control strategy

In figures 4.19 to 4.22, we illustrate the aggregation and the navigation of 6 UAVs in the presence of a faulty UAV. The control law shows a self-organization and fault-tolerant behavior. In figure 4.21, the 6 UAVs perform an aggregation and a navigation toward a circular trajectory, and when UAV 2 is landed, the other 5 UAVs continue the tracking mission and return back to aggregate around the origin with a different formation pattern.

[2]In fact, since the maximum initial interdistance between UAV 5 and 6 is approximately $14m$, we cannot see clearly the "steady-state error", but if we zoom the figure, we can see it clearly.

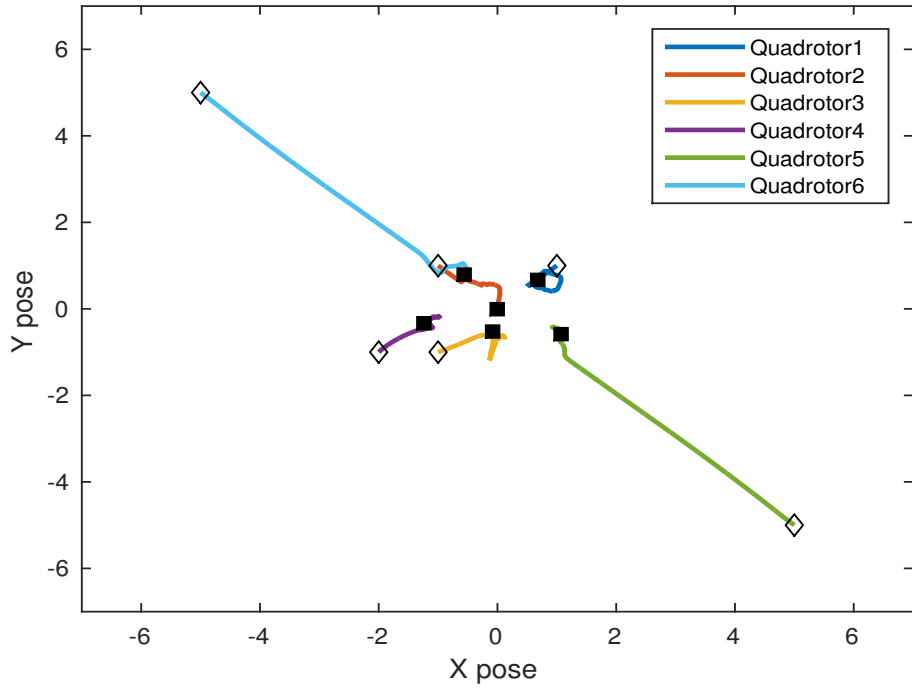


Figure 4.19 – Trajectories of 6 UAVs in an aggregation scenario using tuning gains control with urgent landing in UAV 2

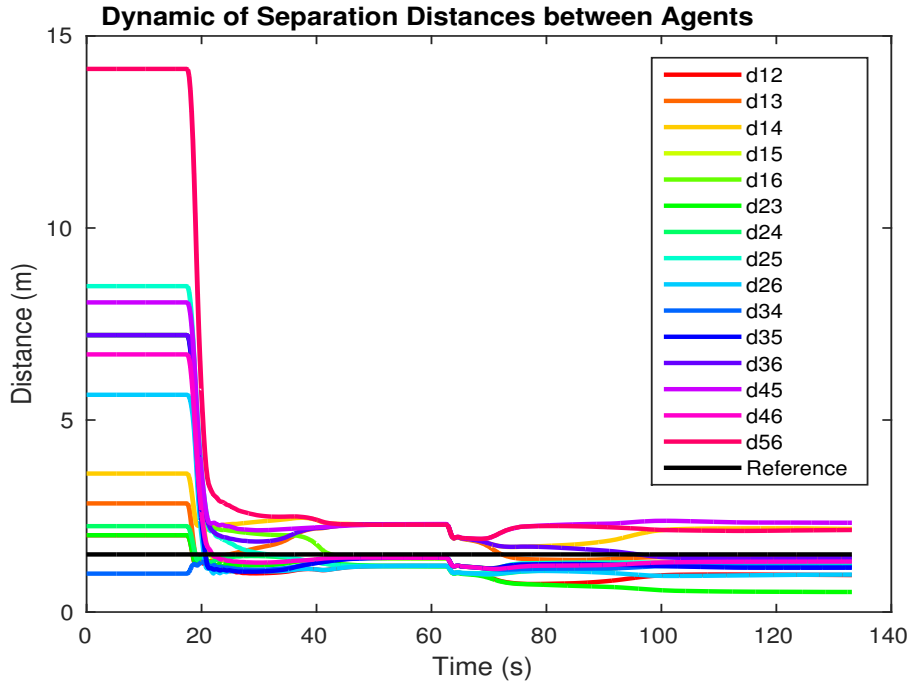


Figure 4.20 – Interdistances between 6 UAVs in an aggregation scenario using tuning gains control with urgent landing in UAV 2

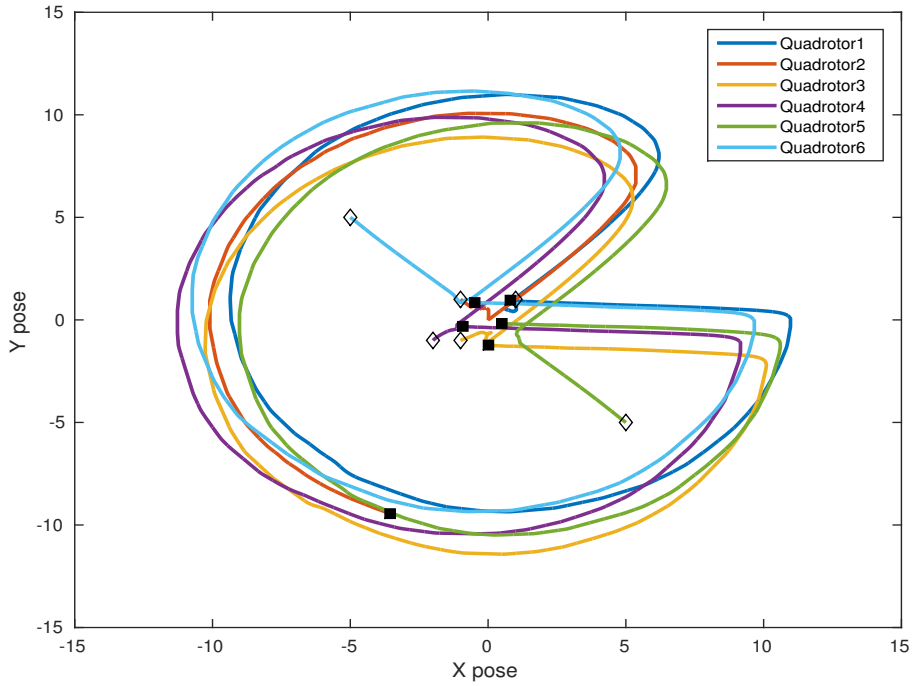


Figure 4.21 – Trajectories of 6 UAVs in circular a navigation scenario using tuning gains control with urgent landing in UAV 2

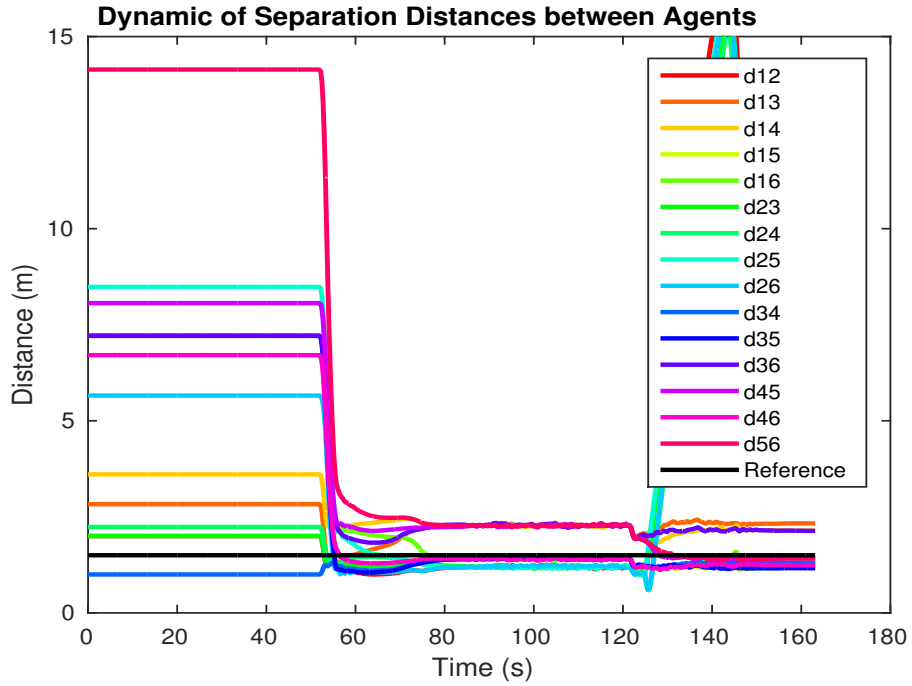


Figure 4.22 – Interdistances between 6 UAVs in a circular navigation scenario using tuning gains control with urgent landing in UAV 2

4.3.2.2 Flocking with distributed integral control

In the previous simulations with different control laws, the main drawback was the "steady-state errors" in the interdistances between neighboring UAVs. In this part, we present the same previous scenarios with the distributed integral control. We keep the same desired interdistance value $1.5m$ and the same interaction range $2m$ to ensure the scalability of this control law. As we will see in the figures, the distributed integral control law overcome the "steady-state errors" problem and ensures, almost, an $\alpha-$lattice pattern together in an aggregation and in a navigation scenarios.

Figure 4.23 shows an aggregation and a circular navigation scenario of 6 UAVs. As seen before, the UAVs perform an aggregation toward the origin and form an $\alpha-$lattice pattern. Then, they follow a circular trajectory while keeping the formation. Finally, they quit the circular trajectory go back in direct navigation toward the origin. We observe here that the final formation, also formed after the aggregation phase, is different from the one seen in figure 4.17 that used the tuning gains control, even if we have the same initial positions. We could say that the presence of integral action renders the interactions between the UAVs more powerful, which prohibits a UAV to go through the other UAVs and to form a pentagonal pattern. However, this observation still needs more study, and the best that we can say is that the observed phenomenon is an emergent unpredictable behavior.

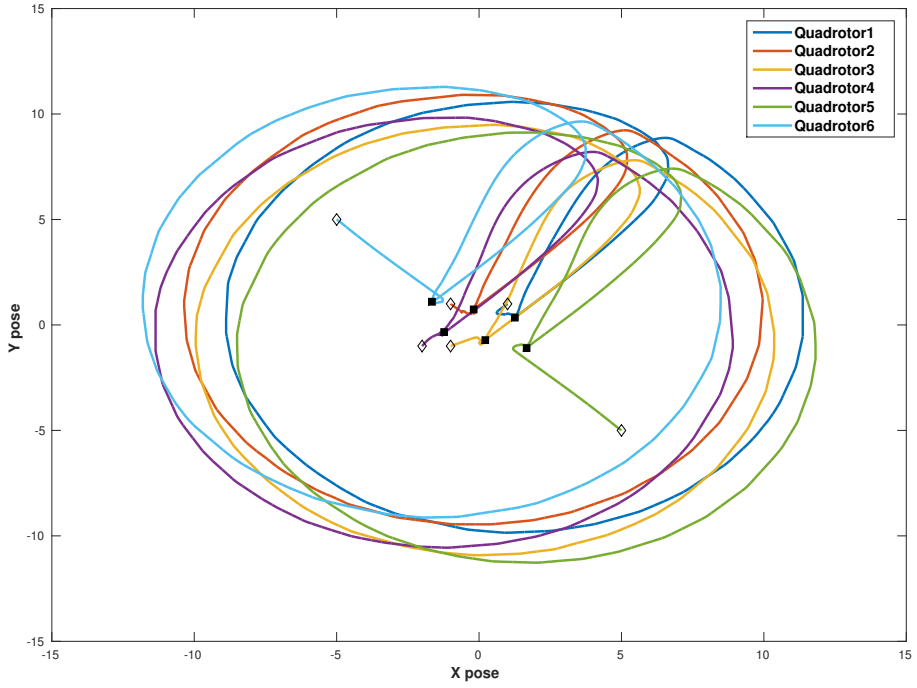


Figure 4.23 – Trajectories of 6 UAVs in aggregation and circular navigation scenario using distributed integral strategy

In figure 4.24, we note the precision of this control law. In fact, all the

interdistances between neighboring UAVs converge to the desired value. This precision continue to persists during different phases of aggregation and navigation scenario. The steady-state errors are completely eliminated and the scalability property is still present as we can see in interdistances between UAVs 5 and 6, and 4 and 5.

Figures 4.25 to 4.28, shows the application of the distributed integral control law in accidental scenarios. From the figures, this control law proves its capability to perform a self-organized and fault-tolerant aggregation and navigation of multiple UAVs, while keeping the precision property.

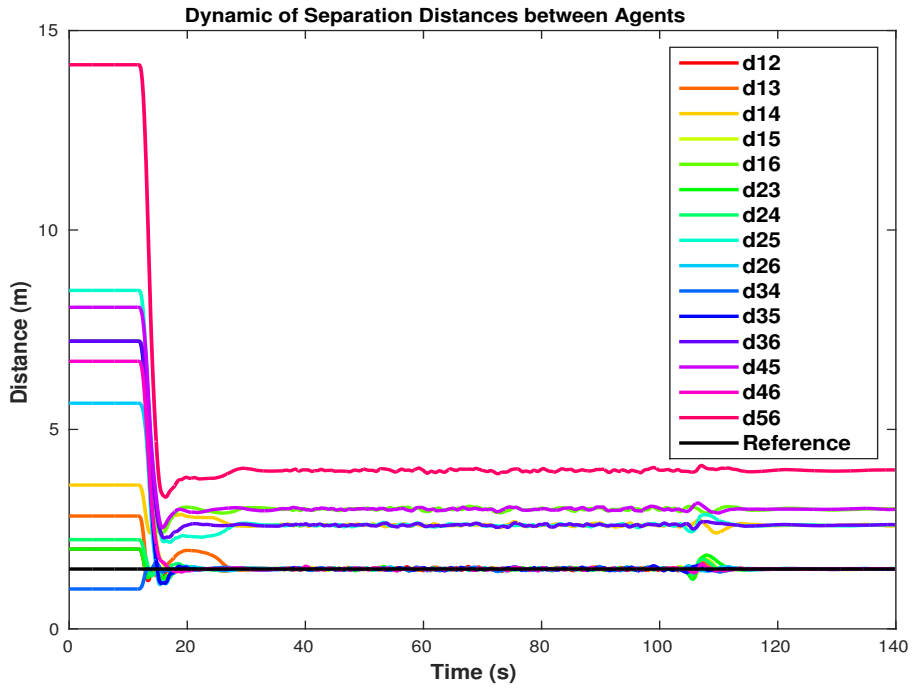


Figure 4.24 – Interdistances between 6 UAVs in aggregation and circular navigation scenario using distributed integral control strategy

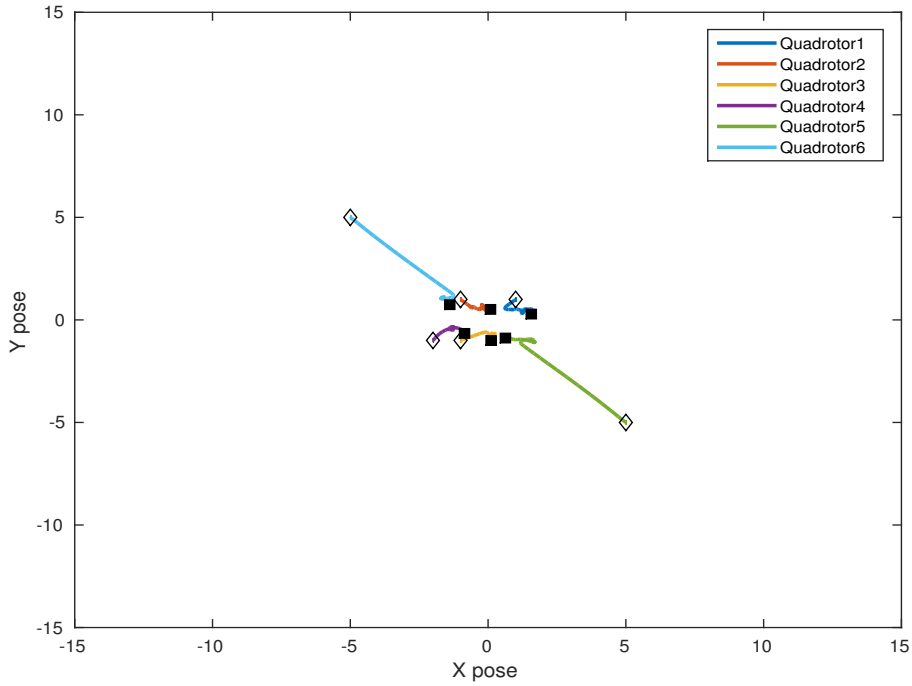


Figure 4.25 – Trajectories of 6 UAVs in an aggregation scenario using distributed integral control with urgent landing in UAV 3

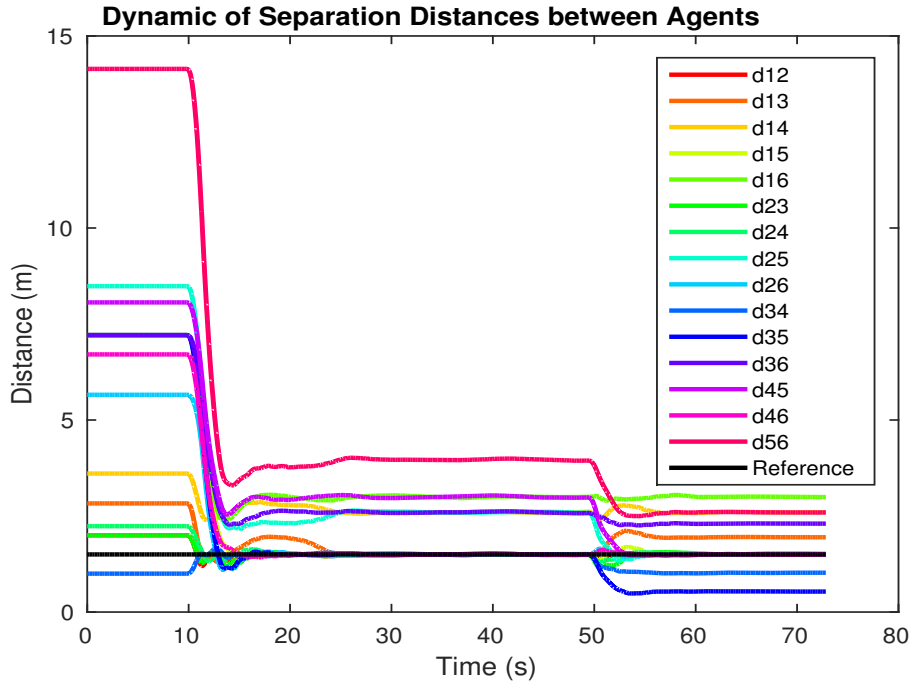


Figure 4.26 – Interdistances between 6 UAVs in an aggregation scenario using distributed integral control with urgent landing in UAV 3

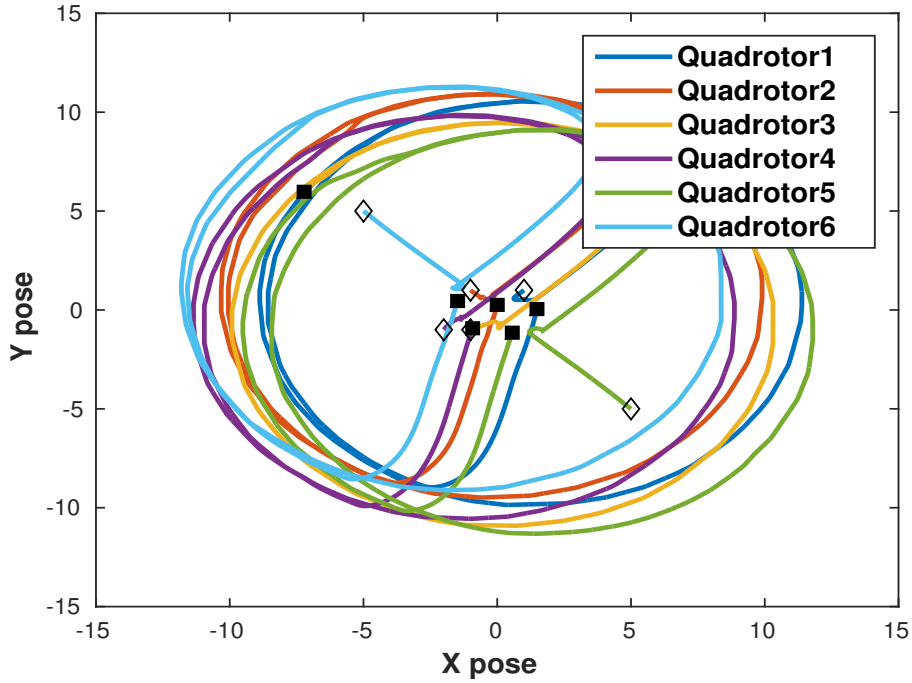


Figure 4.27 – Trajectories of 6 UAVs in a circular navigation scenario using distributed integral control with urgent landing in UAV 3

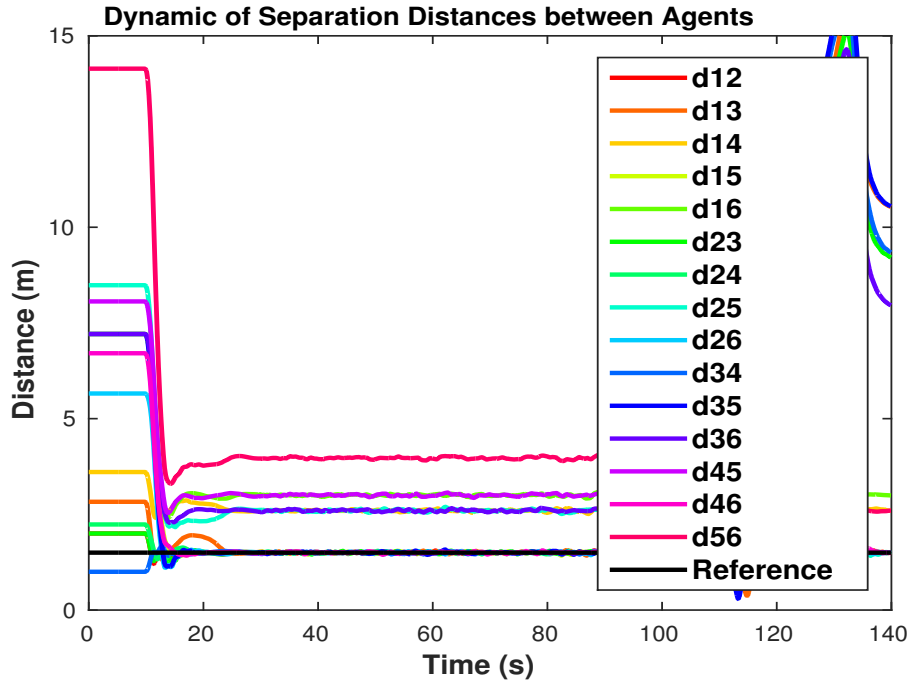


Figure 4.28 – Interdistances between 6 UAVs in a circular navigation scenario using distributed integral control with urgent landing in UAV 3

4.4 Real-Time Experiments

In the following experimental results, our platform is the quadrotor ArDrone2 from Parrot [ParrotArdrone2,], see figure 4.29. Using an SDK provided by Parrot, this platform is designed to be controlled, either from a smartphone, a PC through WiFi, or directly by running a program on the UAV via socket. The utilization of SDK prevents us to use our control laws to stabilize the UAV, since the drone has its own control laws designed by Parrot.

Figure 4.29 – Quadrotor Parrot ArDrone2

We solved this problem through the work of teams from TU Delft University on ArDrone 2 and their autopilot *Paparazzi* [TUDelft,]. They managed to decode the communication protocols between the main processor of the UAV and its sensors and motors. With these protocols, it is possible to directly control the UAV and read the raw data from each sensor. By incorporating these protocols into our own software framework for UAVs, we managed to replace the programs of manufacturer by our own control laws.

The ArDrone 2 is thus mainly used for its material part, whose characteristics are: 1GHz 32 bit ARM Cortex A8 processor, 128 MB RAM, 128MB Flash, WiFi, 3 axis accelerometer, 3 axis gyroscope, 3 axis magnetometer, Pressure sensor, Ultrasound sensors (altitude $< 6m$) and 4 brushless motors. The inertial sensors are used in a complementary filter [Mahony et al., 2008] to estimate the orientation of the UAV.

Experiments are performed in an indoor environment using Optitrack motion capture system [Optitrack,]. The system senses the pose of UAVs at 100 Hz. This information is sent to the UAVs through a Cisco router. Each UAV, then, knows the poses of all UAVs in the flock and can then estimate the velocities. In all experiments, we use the Optitrack frame of reference as our global frame I. Figure 4.30 shows the architecture of our Multiple UAVs platform.

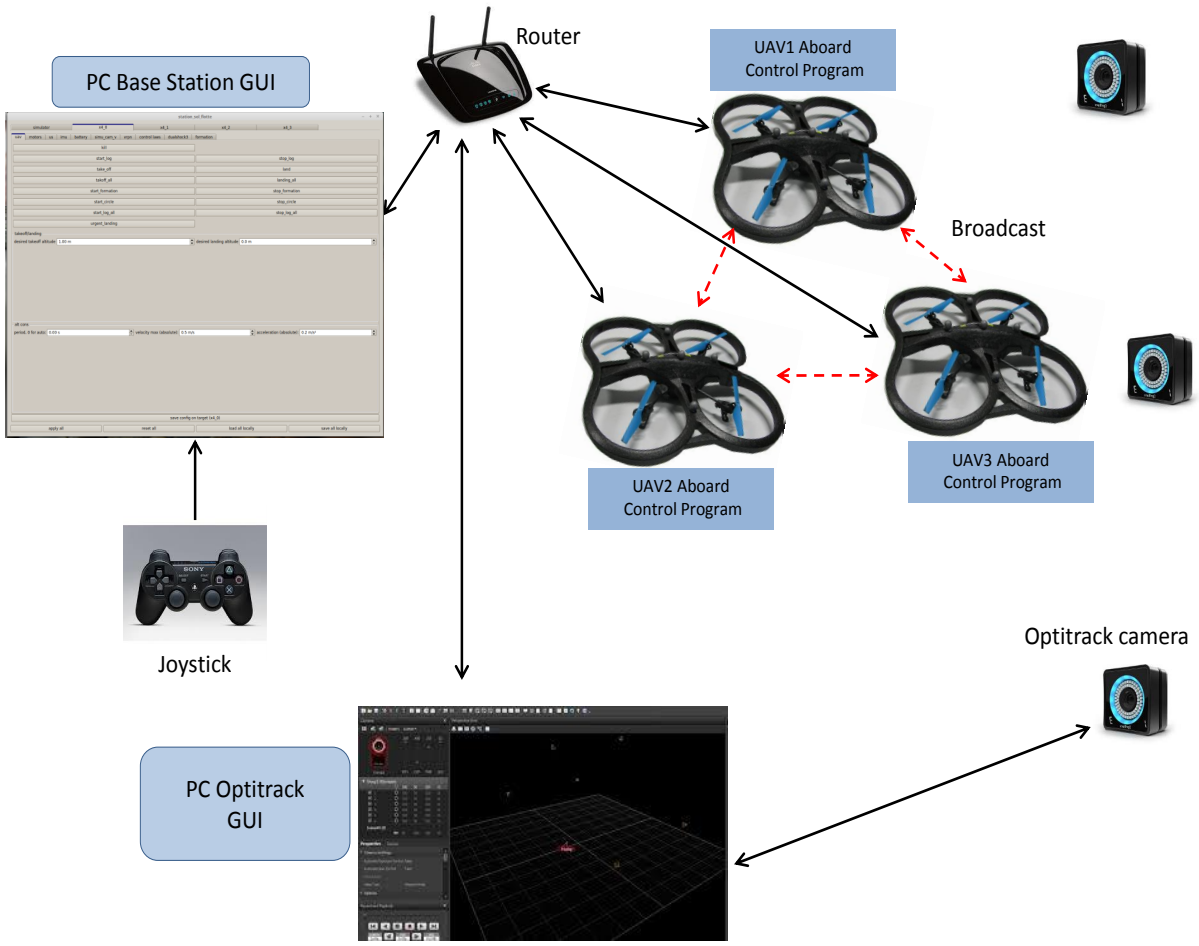


Figure 4.30 – Architecture of the experimental platform. The arrows indicate the communication links between the different components. Discontinuous arrows means occasional communications

We emphasize here that our control laws are run aboard the UAVs. Moreover, our control laws only need the relative distances and directions to the neighboring UAVs. Since we do not have sensors that measure the relative distances to neighbors, we use the Optitrack system as an alternative to extract relative interdistance vectors with neighbors. Thus, we calculate the relative distances, aboard on, by using received positions. Moreover, the $a_{ij}(.)$ function in (3.5) is used to limit the interaction range of UAVs.

4.4.1 Average Strategy

In this experiment, 4 UAVs perform a direct navigation scenario. The UAVs start from their initial positions and navigate to a predefined rendezvous point (-1,1). Figure 4.31 shows the trajectories of the 4 UAVs. Initial and final positions of UAVs are marked by diamonds and black square, respectively.

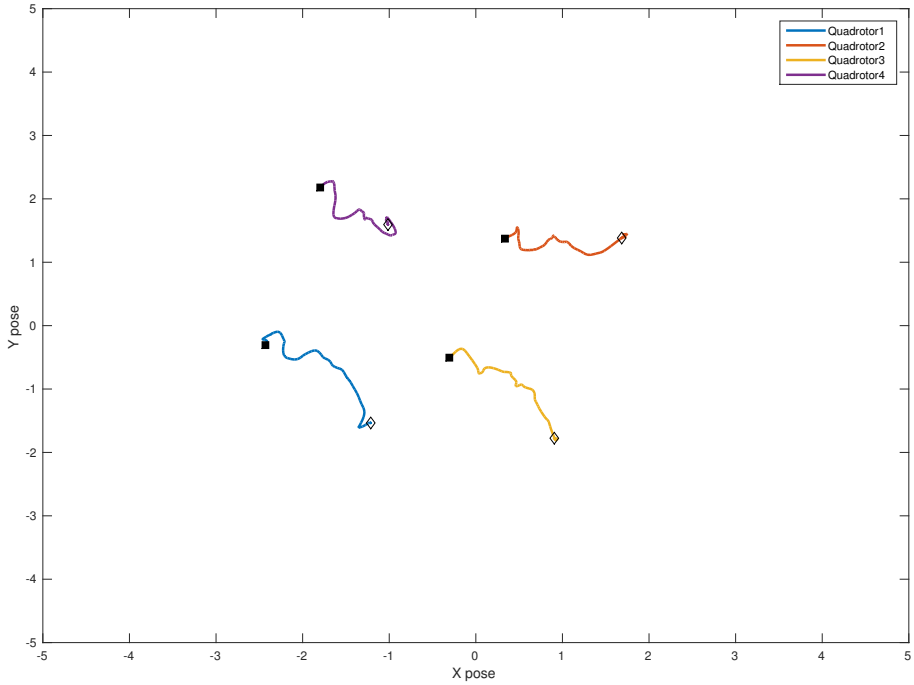


Figure 4.31 – Trajectories of 4 UAVs in a direct navigation scenario using Average control strategy

Figure 4.32 shows the interdistances between UAVs during the experiment as well as the desired interdistance value: $2.5m$. The interdistances between UAVs converge to the desired value but not reach it. Since the average control strategy has a "steady-state-error" problem, as seen in the simulation results, we preferred to increase the desired interdistance in order to avoid any collision between UAVs.

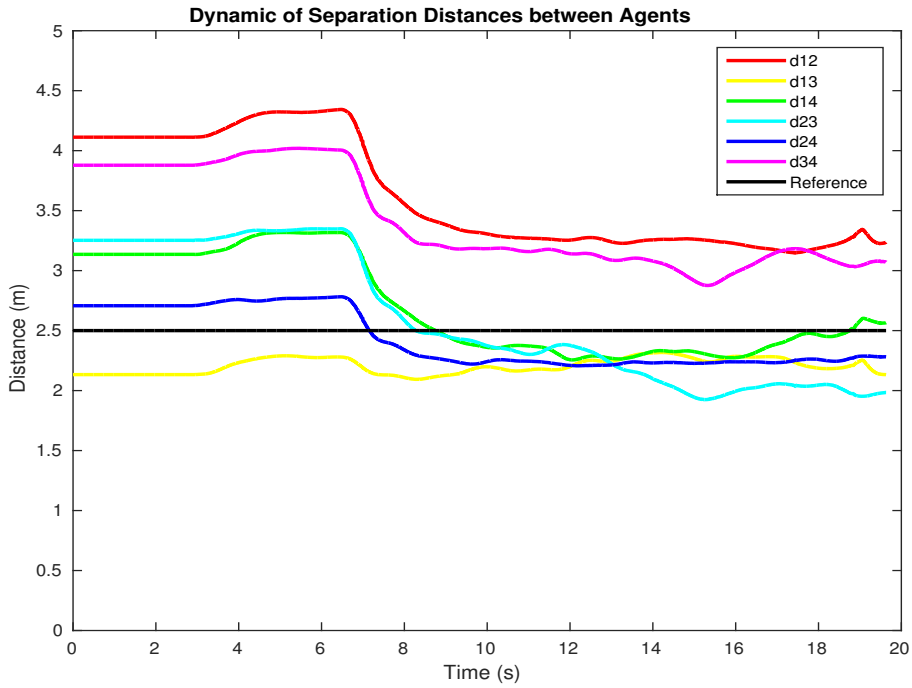


Figure 4.32 – Interdistances between 4 UAVs in a direct navigation scenario using Average control strategy

4.4.2 Consensus-based Flocking

In this section, we present real-time experiments of multiple-UAV flocking. We show the results of three experiments using the control laws in (3.15) and (3.43).

In all of the following experiments, UAVs takeoff from their initial positions to the same defined altitude $r_z = 1m$. The desired yaw angles are set to $r_\psi = 0$ for the whole experiment duration. The formation control law is then launched to form the desired conformation. Finally, the UAVs land after sufficient time.

4.4.2.1 Flocking control with tuning gains

In the first experiment, we use four UAVs to form a *quasi-α-lattice*. We apply our first improved control law in (3.15). The destination point is defined as the origin of the frame I, $q_r = [0\ 0]^T$. In this experiment, we set $K_p = 0.25, K_d = 0.3, c_1 = 0.1, c_2 = 0.2$, $\varepsilon = 0.1$, $h = 0.2$, $c = 2$, the desired distance between neighbors is $d = 1.5$ and the parameters $a = b = 1$.

Figure 4.34 shows the result of using the first complemented control law in (3.15) in the first real-time experiment. The figure exhibits the distances between UAVs over the time. In the experiment, the performance of this control law is improved compared to the one in (3.14). We note, however, a steady-state error in the interdistances between UAVs, i.e. the desired interdistances are not completely reached. This steady-state error could be explained by the presence of continuous

perturbations in the real-time experiment. One of the sources of these perturbations is the downwash of rotor blades.

Figure 4.34 shows the trajectories of UAVs of this experiment. UAVs start at their initial positions designated as black diamonds. Then, they start moving toward the desired destination while they avoid collision with their neighbors. A *quasi-α-lattice* is finally formed.

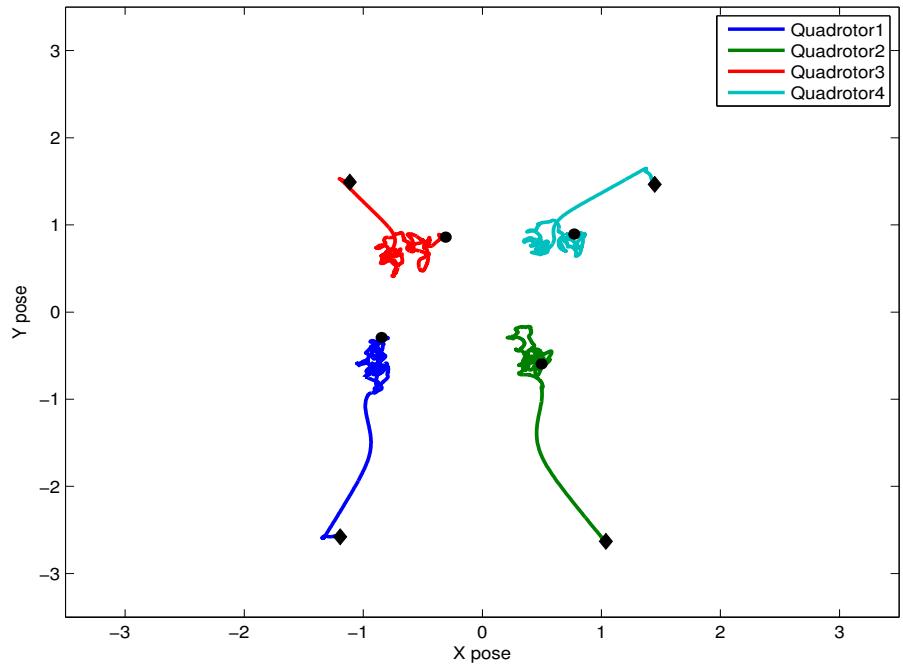


Figure 4.33 – Trajectories of 4 UAVs in an aggregation scenario using tuning gains control strategy

4.4.2.2 Flocking with distributed integral control

In the second experiment, we apply our control law in (3.43). The destination point is designated to be $q_r = [1\ \ 1]^T$. The parameters of this control law is set as $K_p = 0.25$, $K_d = 0.3$, $K_i = 0.09$, $c_1 = 0.1$, $c_2 = 0.2$, $\varepsilon = 0.1$, $h = 0.2$, $c = 2$, the desired distance between neighbors is $d = 1.5$ and the parameters $a = b = 1$.

Figure 4.35 shows the result of the second experiment using the control law (3.43). The UAVs go toward the destination point, and the distances between neighbors converge to the desired value. In this experiment, steady-state errors are eliminated, thanks to the integral action in the alternative control law (3.43). The distance between UAVs 1 and 4 is greater than $d = 1.5$ because this distance exceeds the interaction range $c = 2m$. Figure 4.36 shows the trajectories of the UAVs navigating to the destination point during the experiment.

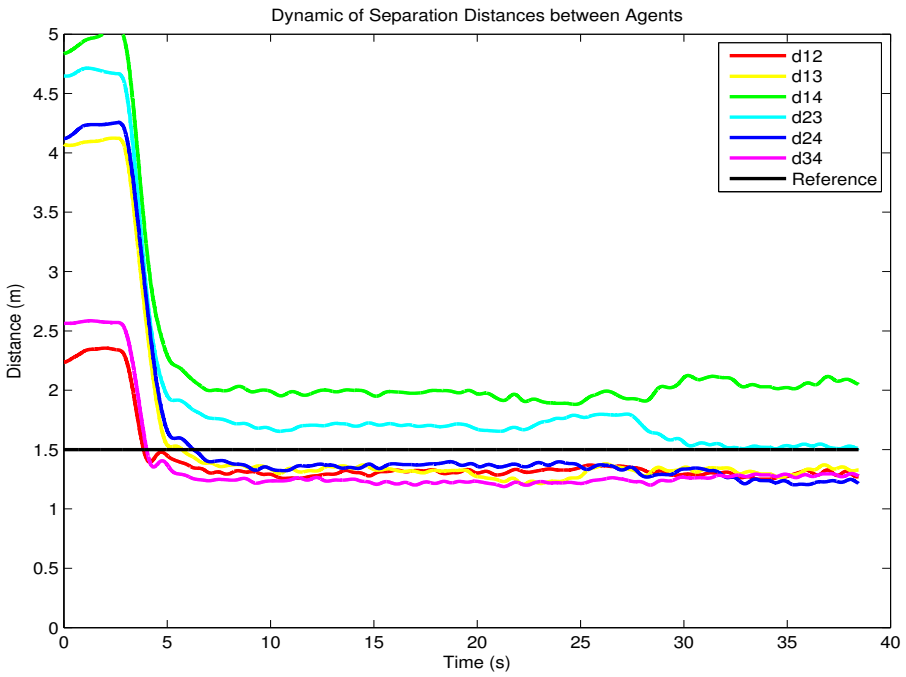


Figure 4.34 – Interdistances between 4 UAVs in an aggregation scenario using tuning gains control strategy

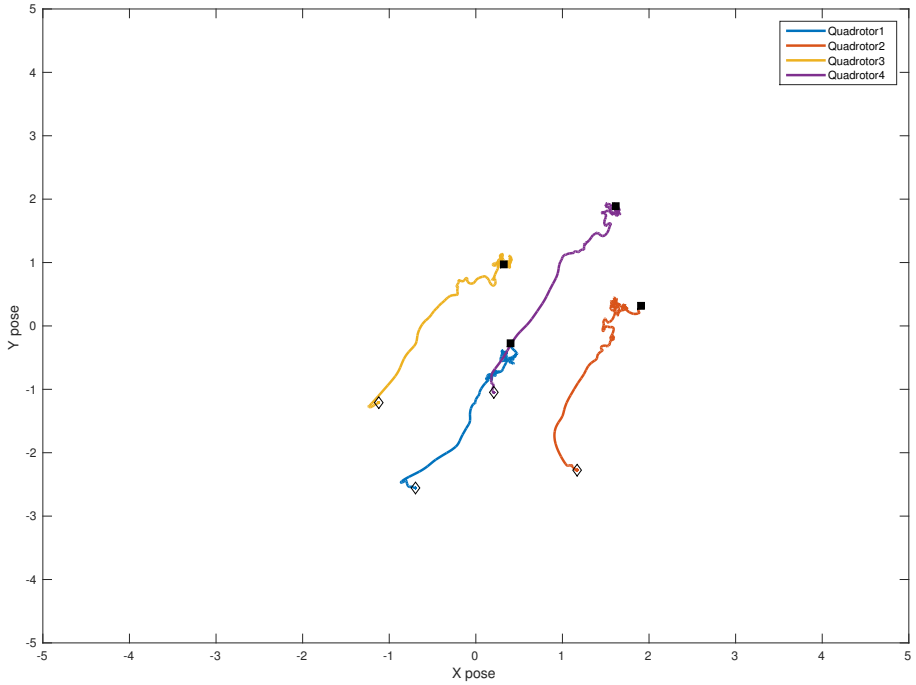


Figure 4.35 – Trajectories of 4 UAVs in a direct navigation scenario using distributed integral strategy

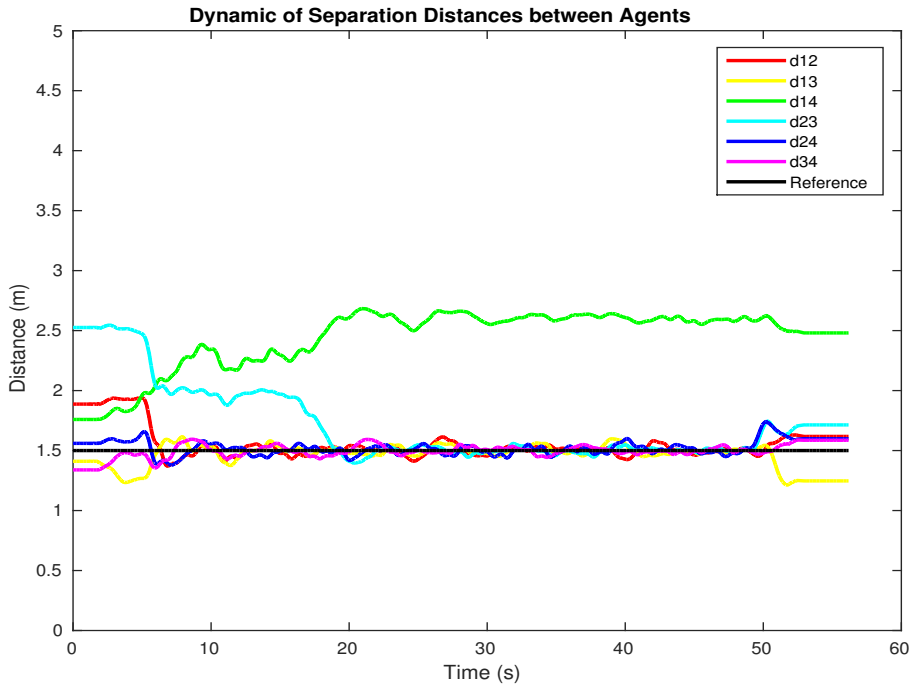


Figure 4.36 – Interdistances between 4 UAVs in a direct navigation scenario using distributed integral control strategy

In the third experiment, we apply the distributed integral control law on 4 UAVs. The objective is to perform an aggregation scenario. As we can see in Figures 4.37 and 4.38, this control law achieves a collision-free aggregation scenario and ensures precise interdistances between UAVs.

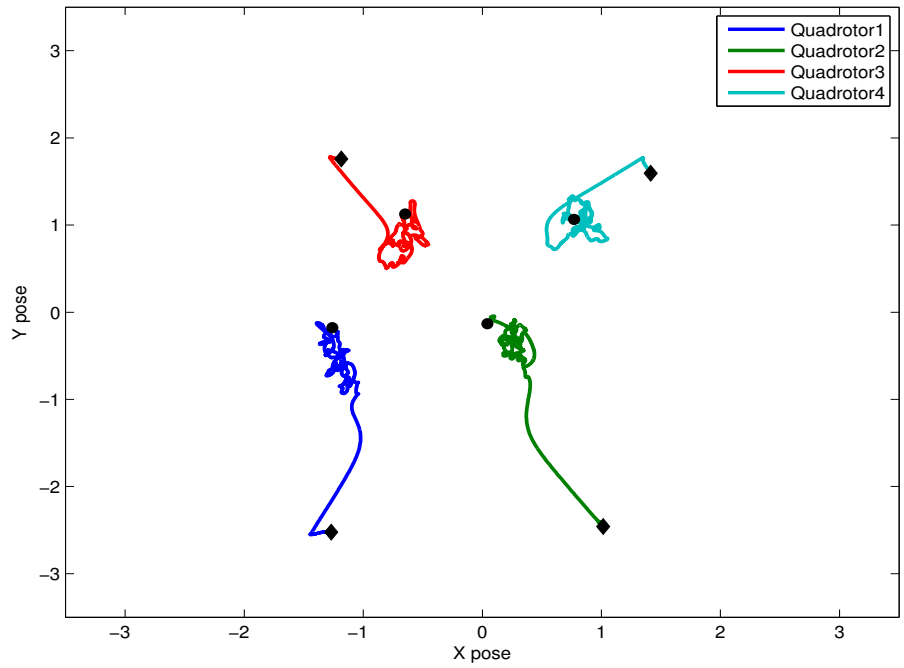


Figure 4.37 – Trajectories of 4 UAVs in an aggregation scenario using distributed integral strategy

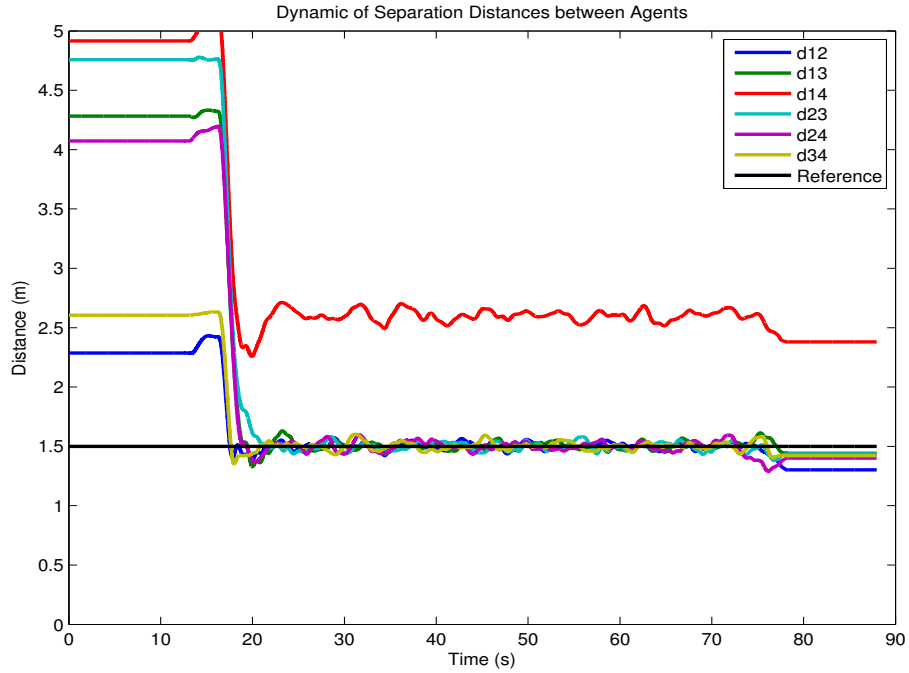


Figure 4.38 – Interdistances between 4 UAVs in an aggregation scenario using distributed integral control strategy

4.5 Conclusion

In this chapter, we show different simulation and experimental results of multiple-UAV flocking by using our proposed control laws in the previous chapters. For the simulation, we use a PC-based simulator of fleet of multiple UAVs. By implementing our control laws on this simulator, we managed to show and to understand the different properties of our control laws. Moreover, the simulator helped us to detect drawbacks in our control laws, such as, "steady-state errors" in the interdistances between UAVs. This important informations guided us to choose the best control law parameters in the experimental work.

The most important properties of our control laws, we noticed in simulation results, are the collision-free formation, the self-organization, the scalability and the fault tolerance. The control law with distributed integral gathers all these properties in addition to a zero-steady-state errors in the interdistances between UAVs. On the contrary, the other control laws share the same drawback of the imprecision of interdistances between UAVs. Moreover, we can conclude from the several simulations of the fault-tolerant property that the distributed architecture and the behavioral-based control used in this work are naturally fault tolerant.

Conclusion and prospects

Contents

In the future, multiple UAVs will be probably widespread as smartphones nowadays. They will be more autonomous, intelligent and cognitive, and present in surveillance, transportation, security and defense, as well as in hobby activities. They could serve in hospitals, police, supermarkets, zoos, and certainly in airports. To make these dreams come true, intensive researches in System of Systems and multiple-UAV control should be carried out. We believe that robustness, fault tolerance, scalability and self organization are some characteristics that researchers have to improve in multiple-UAV system of systems.

In this chapter, we conclude this thesis by reminding the methodology, the lessons learned and their answers, and the difficulties faced during our work. Moreover, we raise the prospects to be continued hoping them will be helpful for our future work and for researchers.

5.1 Working methodology

The objective of this thesis is to design algorithms and techniques to perform a real-time distributed multiple-UAV flight formation control. In the beginning, we have started by literature review of existing works. As a result, theoretical works have been classified and organized. This has allowed us to choose the most useful techniques for our work that we could start with.

In a second time, algorithms have been proposed and simulated based on linear and nonlinear models of quadrotors. In these algorithms, we have focused on the

aggregation and the navigation problems where a flock of multiple UAVs is self-organized to ensure a mission. Unlike the proposed strategies in the literature, our algorithms reduce the complexity of the control law design by proposing a unique control law independent of the number of agents, and dependent only on a generated trajectory issued from neighboring measurements. We also proposed a new behavior which deals with the aggregation of UAVs from a global point of view.

Motivated by the challenges of system of systems, we spotted flocking and consensus algorithms introduced in the literature that could overcome these challenges. Based on these algorithms, we proposed four improved control laws aiming at being compatible with the nonlinear model of quadrotors and experimental works. Moreover, we provided theoretical stability analysis of the collective multiple-UAV system using these control laws and proved their robustness in the presence of perturbations. The control laws have been designed to be run aboard each quadrotor in the flock. By running the control law, each quadrotor has interacted with its neighbors to ensure a collision-free flocking.

During this work, we tested and validated our proposed control laws in simulation and experimental platforms of Heudiasyc laboratory. For simulations, we used a PC-based simulator of flock of multiple quadrotors. For experiments, we implemented our control laws on ArDrone2 quadrotors, which evolved in an indoor environment of Optitrack motion capture system.

5.2 Lessons learned

In this thesis, we sought to answer different questions raised at the beginning. After performing this work, we believe that we can give initial answers to these questions. In the following, we remind the raised questions and propose some answers:

1. The first raised question was, *"How to control and navigate a fleet of UAVs taking into account model nonlinearities and collision avoidance?"* In fact, controlling a fleet of UAVs was not an easy task in this work. With linear model of UAVs, sophisticated control methods are found in the literature. In one hand, with model nonlinearities, two solutions could be considered. The first solution is to linearize the UAV model about its origin, and then propose a control law for this linearized system. The second solution is to use linearization methods, such as feedback or feedforward linearization, and similarly apply appropriate control laws for these linearized systems. In the other hand, collision avoidance between UAVs are solved in this thesis, either by following an average or a sum of shifted state reference, or by using potential

functions.

2. The second question was, *"What are the effects of model uncertainties and exogenous perturbations on the System of Systems?"* The answer of this question is found in the stability analysis of the collective dynamical model of multiple UAVs, see chapter 3. In fact, since system of systems are networked and interactive between them, a local perturbation in one subsystem could propagate to the overall system. This propagation of perturbations could violate the stability of the system of systems, if this issue is not taking in account in the control laws.

3. The third question was, *"How can individuals achieve local rules in a precise and robust way?"* As seen in the simulation and experimental results, the main drawback of the most proposed control laws in this work was the "steady-state errors" in the interdistances between UAVs. This problem was solved by introducing a distributed integral control law.

4. The last question was, *"While a System of Systems is supposed to be scalable, is it fault tolerant?"* The distributed architecture and the behavioral-based control structure used in this work are fault tolerant by nature, as it is shown in the simulation and experimental results. Thus, for a SoS to be fault tolerant, it is preferable to use these techniques. However, we believe that this question still needs more study, since the tested cases covered only a special fault, that is, urgent landing of one UAV in the fleet. We believe that SoS characteristics could be improved by employing advanced fault-tolerant techniques.

5.3 Difficulties

Final results are usually seen as convenient and promising, but they hide a lot of difficulties. In the following, we present the most important difficulties we faced during our work:

- **Communications**: In our multiple-UAV system, communications were a crucial issue. For example, to make UAVs responding to global high level command, such as, "takeoff all UAVs", "landing all UAVs" and "Start formation", these commands are broadcast through socket messages from one principle UAV to the fleet. A failure in receiving these messages leads to a failure in the mission, and sometimes leads to accidents, hence the necessity of employing advanced fault-tolerance techniques.

- **Gains tunning**: In average and sum control strategies, it was relatively easy to deploy a fleet of quadrotors in formation. In fact, it was sufficient to ensure that the controller of each quadrotor can track a given reference, to employ then one of these control strategies. However, with consensus-based algorithms, gains tunning was a difficult task in the simulation and experimental work. We had to start by tunning the gains of one quadrotor, retuning for two and then for three quadrotors to get better results for a number of quadrotors greater than three. Moreover, even if we have the same type of quadrotors, we still need to tune the non flocking gains (gains to stabilize altitude or orientation) for some quadrotors.

- **Power consumption**: We could conclude after a lot of experiments done during this thesis that flocking algorithms are power consuming. A quadrotor in a flocking test, with a battery of 1000 mAh, consumes at least 3 times the consumption of one quadrotor in an ordinary test.

5.4 Prospects

In the following, we present some potential future research prospects in continuation of this thesis:

- **Outdoor experiment**: In this work, real-time experiments were performed in a perfect localization environment. The relative positions of quadrirotors are obtained through fixed and centralized system "Optitrack". A more challenging work will be to develop a fleet of multiple UAVs for outdoor and unknown environment. Thus, for instance, we need to embed, on each quadrotor, sensors capable of providing proprioceptive and exteroceptive information. Examples of such sensors are a GPS, cameras or a laser rangefinder. We believe that the test of SoS algorithms on such a system will reveal more knowledge and questions. It is important to mention here that, at the end of this thesis, we began a first step in this direction by working on an image-based flocking algorithm. However, this work was not yet mature.

- **Flocking of nonholonomic UAVs**: In this thesis, we worked on a type of UAVs that is considered as holonomic robots. However, developing and testing a system of nonholonomic UAVs, like fixed-wing airplane, and taking into account nonholonomic constraints in the flocking control, will be more challenging.

- **Fault-tolerant in multiple UAVs**: As mentioned in this thesis, the fault-tolerance case tested in this work does not cover all possible faults. Communication failure, perturbations, sudden accident in one or more UAV in the flock, etc., are problems that should encourage researchers to employ and to develop advanced fault-tolerance techniques in multiple-UAV system of systems.

- **Obstacle avoidance**: An important problem raised in UAV field is "sense and avoid autonomously", that is for one UAV. A promising research topic could be the navigation and obstacle avoidance of multiple UAVs.

Bibliography

[Anderson and Moore, 1990] Anderson, B. D. O. and Moore, J. B. (1990). *Optimal Control: Linear Quadratic Methods*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[Antonelli et al., 2010] Antonelli, G., Arrichiello, F., and Chiaverini, S. (2010). Flocking for multi-robot systems via the null-space-based behavioral control. *Swarm Intelligence*, 4(1):37–56.

[Bakule, 2008] Bakule, L. (2008). Decentralized control: An overview. *Annual Reviews in Control*, 32(1):87 – 98.

[Bellingham et al., 2002] Bellingham, J., Tillerson, M., Alighanbari, M., and How, J. (2002). Cooperative path planning for multiple uavs in dynamic and uncertain environments. In *Proceedings of the 41st IEEE Conference on Decision and Control*, volume 3, pages 2816–2822.

[Bouabdallah et al., 2004a] Bouabdallah, S., Murrieri, P., and Siegwart, R. (2004a). Design and control of an indoor micro quadrotor. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 5, pages 4393–4398.

[Bouabdallah et al., 2004b] Bouabdallah, S., Noth, A., and Siegwart, R. (2004b). Pid vs lq control techniques applied to an indoor micro quadrotor. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2451–2456.

[Bouabdallah and Siegwart, 2005] Bouabdallah, S. and Siegwart, R. (2005). Backstepping and sliding-mode techniques applied to an indoor micro quadrotor. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2247–2252.

[Bouabdallah and Siegwart, 2007] Bouabdallah, S. and Siegwart, R. (2007). Full control of a quadrotor. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), California, USA*, pages 153–158.

[Bouffard et al., 2012] Bouffard, P., Aswani, A., and Tomlin, C. (2012). Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 279–284.

[Castillo et al., 2004] Castillo, P., Dzul, A., and Lozano, R. (2004). Real-time stabilization and tracking of a four-rotor mini rotorcraft. *IEEE Transactions on Control Systems Technology,*, 12(4):510–516.

[Chiaramonti et al., 2006] Chiaramonti, M., Giulietti, F., and Mengali, G. (2006). Formation control laws for autonomous flight vehicles. In *14th Mediterranean Conference on Control and Automation. MED '06*, pages 1–5.

[Couzin, 2009] Couzin, I. D. (2009). Collective cognition in animal groups. *Trends in Cognitive Sciences*, 13(1):36 – 43.

[Diestel, 2005] Diestel, R. (2005). *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Heidelberg, third edition.

[Dimarogonas and Kyriakopoulos, 2006] Dimarogonas, D. and Kyriakopoulos, K. (2006). A connection between formation control and flocking behavior in nonholonomic multiagent systems. In *Proceedings 2006 IEEE International Conference on Robotics and Automation. ICRA 2006.*, pages 940–945.

[Erginer and Altug, 2007] Erginer, B. and Altug, E. (2007). Modeling and pd control of a quadrotor vtol vehicle. In *IEEE Intelligent Vehicles Symposium, 2007*, pages 894–899.

[Formentin and Lovera, 2011] Formentin, S. and Lovera, M. (2011). Flatness-based control of a quadrotor helicopter via feedforward linearization. In *2011 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, pages 6171–6176.

[Franchi et al., 2012] Franchi, A., Secchi, C., Ryll, M., Bulthoff, H., and Giordano, P. (2012). Shared control : Balancing autonomy and human assistance with a group of quadrotor uavs. *IEEE Robotics Automation Magazine*, 19(3):57–68.

[Fresk and Nikolakopoulos, 2013] Fresk, E. and Nikolakopoulos, G. (2013). Full quaternion based attitude control for a quadrotor. In *2013 European Control Conference (ECC),*, pages 3864–3869.

[Ghandour et al., 2014] Ghandour, J., Aberkane, S., and Ponsart, J.-C. (2014). Feedback linearization approach for standard and fault tolerant control: Application to a quadrotor uav testbed. *Journal of Physics: Conference Series*, 570(8):082003.

[Giulietti et al., 2000] Giulietti, F., Pollini, L., and Innocenti, M. (2000). Autonomous formation flight. *IEEE Control Systems*, 20(6):34–44.

[Godsil and Royle, 2001] Godsil, C. and Royle, G. (2001). *Algebraic graph theory*, volume 207 of *Graduate Texts in Mathematics*. Springer-Verlag, New York.

[Guerrero and Lozano, 2012] Guerrero, J.-A. and Lozano, R. (2012). *UAV flight formation control*. John Wiley-ISTE.

[Guerrero-Castellanos et al., 2011] Guerrero-Castellanos, J.-F., Marchand, N., Hably, A., Lesecq, S., and Delamare, J. (2011). Bounded attitude control of rigid bodies: Real-time experimentation to a quadrotor mini-helicopter. *Control Engineering Practice*, 19(8):790–797. hal-00568075.

[Guerrero Castellanos et al., 2015] Guerrero Castellanos, J. F., Rifai, H., Marchand, N., Cruz-José, R., Mohammed, S., Guerrero Sanchez, W. F., and Mino-Aguilar, G. (2015). Biomimetic-based output feedback for attitude stabilization of rigid bodies: Real-time experimentation on a quadrotor. *Micromachines*, 6(8):993–1022. hal-01184073.

[Guerrero Castellanos et al., 2014] Guerrero Castellanos, J.-F., Téllez-Guzmán, J. J., Durand, S., Marchand, N., Álvarez Muñoz, J., and González-Díaz, V. (2014). Attitude stabilization of a quadrotor by means of event-triggered nonlinear control. *Journal of Intelligent and Robotic Systems*, 73:123–135. hal-00860840.

[Hespanha, 2009] Hespanha, J. P. (2009). *Linear Systems Theory*. Princeton Press, Princeton, New Jersey. ISBN13: 978-0-691-14021-6.

[Hou and Fantoni, 2015] Hou, Z. and Fantoni, I. (2015). Distributed leader-follower formation control for multiple quadrotors with weighted topology. In *10th System of Systems Engineering Conference (SoSE), 2015*, pages 256–261.

[Jamshidi, 2008] Jamshidi, M. (2008). *Systems of systems engineering: principles and applications*. CRC press.

[Johnson and Kannan, 2003] Johnson, E. and Kannan, S. (2003). Nested saturation with guaranteed real poles. In *Proceedings of the 2003 American Control Conference, Colorado, USA*, volume 1, pages 497–502 vol.1.

[Jovanovic, 2004] Jovanovic, M. R. (2004). *Modeling, Analysis, and Control of Spatially Distributed Systems*. PhD thesis, University of California Santa Barbara.

[Kendoul et al., 2007] Kendoul, F., Lara, D., Fantoni, I., and Lozano, R. (2007). Real-time nonlinear embedded control for an autonomous quadrotor helicopter. *Journal of Guidance, Control, and Dynamics*, 30:1049–1061.

[Khalil, 2002] Khalil, H. K. (2002). *Nonlinear Systems*. Prentice Hall.

[Kushleyev et al., 2013] Kushleyev, A., Mellinger, D., Powers, C., and Kumar, V. (2013). Towards a swarm of agile micro quadrotors. *Autonomous Robots*, 35:287–300.

[Lee, 2013] Lee, T. (2013). Robust adaptive attitude tracking on so(3) with an application to a quadrotor uav. *IEEE Transactions on Control Systems Technology*, 21(5):1924–1930.

[Lee et al., 2010] Lee, T., Leoky, M., and McClamroch, N. (2010). Geometric tracking control of a quadrotor uav on se(3). In *49th IEEE Conference on Decision and Control (CDC)*, pages 5420–5425.

[Lewis and Syrmos, 2012] Lewis, F. L. and Syrmos, V. L. (2012). *Optimal control*. J. Wiley, New Jersey.

[Li et al., 2011] Li, Z., Hovakimyan, N., and Stipanovic, D. (2011). Distributed multi-agent tracking and estimation with uncertain agent dynamics. In *American Control Conference (ACC)*, pages 2204–2209.

[Lozano, 2010] Lozano, R. (2010). *Unmanned Aerial Vehicles Embedded Control*. John Wiley-ISTE Ltd.

[López-Araujo et al., 2010] López-Araujo, D., Zavala-Río, A., Fantoni, I., Salazar, S., and Lozano, R. (2010). Global stabilisation of the pvtol aircraft with lateral

force coupling and bounded inputs. *International Journal of Control*, 83(7):1427–1441.

[Madani and Benallegue, 2006] Madani, T. and Benallegue, A. (2006). Backstepping control for a quadrotor helicopter. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3255–3260.

[Mahony et al., 2008] Mahony, R., Hamel, T., and Pflimlin, J.-M. (2008). Nonlinear complementary filters on the special orthogonal group. *IEEE Transactions on Automatic Control*, 53(5):1203–1218.

[Mahony et al., 2012] Mahony, R., Kumar, V., and Corke, P. (2012). Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robotics Automation Magazine*, 19(3):20–32.

[Murray, 2010] Murray, R. M. (2010). *Optimization-Based Control*. California Institute of Technology.

[Ogata, 2010] Ogata, K. (2010). *Modern Control Engineering*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 5th edition.

[Olfati-Saber, 2006] Olfati-Saber, R. (2006). Flocking for multi-agent dynamic systems: algorithms and theory. *IEEE Transactions on Automatic Control*, 51(3):401–420.

[Olfati-Saber and Murray, 2002] Olfati-Saber, R. and Murray, R. M. (2002). Distributed cooperative control of multiple vehicle formations using structural potential functions. In *IFAC World Congress*.

[Optitrack,] Optitrack. Optitrack motion capture systems. `https://www.naturalpoint.com/optitrack/`.

[ParrotArdrone2,] ParrotArdrone2. `http://ardrone2.parrot.com/`.

[Partridge, 1982] Partridge, B. (1982). The structure and function of fish schools. *Scientific American*, 246(06):114–123.

[Reynolds, 1987] Reynolds, C. W. (1987). Flocks, herds, and schools: A distributed behavioral model. In *Computer Graphics*, pages 25–34.

[Richards and How, 2002] Richards, A. and How, J. (2002). Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *Proceedings of the American Control Conference*, volume 3, pages 1936–1941 vol.3.

[Roberson and Stilwell, 2006] Roberson, D. and Stilwell, D. (2006). Decentralized control and estimation for a platoon of autonomous vehicles with a circulant communication network. In *American Control Conference*.

[Sanahuja, 2010] Sanahuja, G. (2010). *Commande et localisation embarquée d'un drone aérien en utilisant la vision*. PhD thesis, Université de technologie de compiègne.

[Schollig et al., 2010] Schollig, A., Augugliaro, F., Lupashin, S., and D'Andrea, R. (2010). Synchronizing the motion of a quadrocopter to music. In *IEEE International Conference on Robotics and Automation (ICRA), Anchorage, Alaska*, pages 3355–3360.

[ScienceEtVie, 2014] ScienceEtVie (2014). Science et vie. Nř 1160.

[Shi et al., 2006] Shi, H., Wang, L., and Chu, T. (2006). Virtual leader approach to coordinated control of multiple mobile agents with asymmetric interactions. *Physica D: Nonlinear Phenomena*, 213(1):51 – 65.

[Tanner et al., 2005] Tanner, H., Jadbabaie, A., and Pappas, G. (2005). Flocking in teams of nonholonomic agents. In Kumar, V., Leonard, N., and Morse, A., editors, *Cooperative Control*, volume 309 of *Lecture Notes in Control and Information Science*, pages 229–239. Springer Berlin Heidelberg.

[Tanner et al., 2007] Tanner, H., Jadbabaie, A., and Pappas, G. (2007). Flocking in fixed and switching networks. *IEEE Transactions on Automatic Control*, 52(5):863–868.

[Tayebi and McGilvray, 2006] Tayebi, A. and McGilvray, S. (2006). Attitude stabilization of a vtol quadrotor aircraft. *IEEE Transactions on Control Systems Technology*, 14(3):562–571.

[Teel, 1992] Teel, A. R. (1992). Global stabilization and restricted tracking for multiple integrator with bounded control. *Systems and Control Letters*, 18:165–171.

[TUDelft,] TUDelft. http://wiki.paparazziuav.org/wiki/AR_Drone_2/getting_started.

[Valbuena Reyes and Tanner, 2015] Valbuena Reyes, L. and Tanner, H. (2015). Flocking, formation control, and path following for a group of mobile robots. *IEEE Transactions on Control Systems Technology*, 23(4):1268–1282.

[Vasarhelyi et al., 2014] Vasarhelyi, G., Viragh, C., Somorjai, G., Tarcai, N., Szorenyi, T., Nepusz, T., and Vicsek, T. (2014). Outdoor flocking and formation flight with autonomous aerial robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3866–3873.

[Wikipedia,] Wikipedia. System: https://en.wikipedia.org/wiki/system.

[Zavala-Río et al., 2003] Zavala-Río, A., Fantoni, I., and Lozano, R. (2003). Global stabilization of a pvtol aircraft model with bounded inputs. *International Journal of Control*, 76(18):1833–1844.

[Zhang, 2006] Zhang, X. (2006). An output feedback nonlinear decentralized controller design for multiple unmanned vehicle coordination. In *American Control Conference*.