Conception de commande tolérante aux défauts pour les systèmes multi-agents

Adel Belkadi

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Conception de commande tolérante aux défauts pour les systèmes multi-agents : application au vol en formation d'une flotte de véhicules autonomes aériens

THÈSE

présentée et soutenue publiquement le 12 Octobre 2017

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention Automatique)

par

Adel BELKADI

Composition du jury

Rapporteurs :	Isabelle FANTONI	Directrice de Recherche CNRS, Univ de Technologie de Compiègne. Heudiasyc UMR CNRS 7253
	Mustapha OULADSINE	Professeur, Université Aix-Marseille Directeur du laboratoire des Sciences de l'Information et des Systèmes (LSIS)
Examinateurs :	Amal El FALLAH Seghrouchni	Professeur, LIP6 - Université Pierre and Marie Curie
	Laurent CIARLETTA	Maître de Conférences, Université de Lorraine LORIA, Campus Scientifique
	Constantin MORARESCU	Maître de Conférences, HDR, Université de Lorraine CRAN UMR 7039, CNRS
Invité :	Pedro CASTILLO	CR CNRS, HDR, Univ de Technologie de Compiègne CNRS UMR 7253 Heudiasyc, Sorbonne Universités
Directeur de thèse :	Didier THEILLIOL	Professeur, Université de Lorraine CRAN UMR 7039, CNRS

Mis en page avec la classe thesul.

Remerciements

Les travaux présentés dans ce rapport de thèse ont été effectués à l'Université de Lorraine entre octobre 2014 et Septembre 2017, au sein de l'équipe CID du Centre de Recherche en Automatique de Nancy (CRAN CNRS UMR 7039). Je remercie par conséquent Monsieur Didier WOLF, directeur du CRAN, ainsi que tous les enseignants-chercheurs, équipes de direction (particulièrement madame Sabine Huraux) et collègues doctorants pour m'avoir accueilli pendant ces trois années et pour m'avoir permis de réaliser ces travaux dans les meilleures conditions.

Le financement de mon contrat doctoral a été soutenu par la Région LORRAINE et la Fédération Charles Hermite, je remercie donc monsieur Pierre Vallois directeur de la FCH, pour son soutien et sa confiance.

Je tiens également à exprimer mes sincères remerciements à mes encadrants, Prof. Didier THEILLIOL et Laurent CIARLETTA. Merci pour vos nombreux conseils, suggestions et recommandations. Merci aussi pour ces réunions et discussions toujours stimulantes qui m'ont encouragé à aller au-delà de mes attentes et m'ont fourni un excellent environnement académique qui était une clé Pour le résultat de nos recherches.

Je remercie chaleureusement les membres du jury qui ont accepté d'examiner minutieusement mes travaux de thèse. Je remercie ainsi Madame Isabelle FANTONI, Directrice de Recherche CNRS au laboratoire Heudiasyc à l'université de Technologie de Compiègne, Monsieur Mustapha Ouladsine, Professeur à l'université Aix-Marseille et Directeur du laboratoire des sciences de l'information et des systèmes (LSIS), Madame Amal El Fallah Seghrouchni, Professeur à l'université Pierre and Marie Curie, Monsieur Pedro CASTILLO, Chargé de Recherche CNRS, à l'université de Technologie de Compiègne et monsieur Constantin MORARESCU, maître de Conférences à l'université de Lorraine. Qu'ils trouvent ici l'expression de ma plus profonde gratitude.

En parallèle de mes travaux de recherche, j'ai eu l'opportunité d'avoir un contrat d'enseignement (DCCE) de trois ans à l'Institut Universitaire de Technologie de Lunéville au département Qualité Logistique Industrielle et Organisation (QLIO). Je tiens donc à remercier tous les membres de l'équipe pédagogique de Lunéville avec qui ce fut un réel plaisir de travailler et plus particulièrement monsieur Alexandre VOISIN, madame Isabelle PERRY et monsieur Hugues HAOUY, sans oublier madame Sigrid HENRY et monsieur Jean-Claude MASSON.

Je remercie encore monsieur Pedro CASTILLO, monsieur Hernán Abaunza González, doctorant à l'université de Compiègne et monsieur Guillaume SANAHUJA, ingénieur de recherche au laboratoire Heudiasyc pour leur collaboration et leur aide dans l'implémentation de nos algorithmes sur la plateforme de drones développée à l'UTC.

Je dédie cette thèse
à mes chers
parents

Résumé

Ces dernières années, l'émergence des nouvelles technologies tels que la miniaturisation des composants, les dispositifs de communication sans fils, l'augmentation de la taille de stockage et les capacités de calcul, a permis la conception de systèmes multi-agents coopératifs de plus en plus complexes. L'un des plus grands axes de recherche dans cette thématique concerne la commande en formation de flottes de véhicules autonomes. Un grand nombre d'applications et de missions, civiles et militaires, telles que l'exploration, la surveillance, et la maintenance, ont alors été développées et réalisées dans des milieux variés (terre, air, eau). Durant l'exécution de ces tâches, les véhicules doivent interagir avec leur environnement et entre eux pour se coordonner. Les outils de communication disponibles disposent souvent d'une portée limitée. La préservation de la connexion au sein du groupe devient alors un des objectifs à satisfaire pour que la tâche puisse être accomplie avec succès. Une des possibilités pour garantir cette contrainte est le déplacement en formation permettant de préserver les distances et la structure géométrique du groupe. Il est toutefois nécessaire de disposer d'outils et de méthodes d'analyse et de commande de ces types de systèmes afin d'exploiter au maximum leurs potentiels. Cette thèse s'inscrit dans cette direction de recherche en présentant une synthèse et une analyse des systèmes dynamiques multi-agents et plus particulièrement la commande en formation de véhicules autonomes. Les lois de commande développées dans la littérature pour la commande en formation permettent d'accomplir un grand nombre de missions avec un niveau de performance élevé. Toutefois, si un défaut/défaillant apparaît dans la formation, ces lois de commandes peuvent s'avérer très limitées, engendrant un comportement instable du système. Le développement de commandes tolérantes aux défauts devient alors primordial pour maintenir les performances de commande en présence de défauts. cette problématique sera traitée dans ce mémoire de thèse et concernera le développement et la conception de commandes en formation tolérantes au défaut dévolu à une flotte de véhicules autonomes suivant différente configuration/structuration.

Abstract

In recent years, the emergence of new technologies such as miniaturization of components, wireless communication devices, increased storage size and computing capabilities have allowed the design of increasingly complex cooperative multiagent systems. One of the main research axes in this topic concerns the formation control of fleets of autonomous vehicles. Many applications and missions, civilian and military, such as exploration, surveillance, and maintenance, were developed and carried out in various environments. During the execution of these tasks, the vehicles must interact with their environment and among themselves to coordinate. The available communication tools are often limited in scope. The preservation of the connection within the group then becomes one of the objectives to be satisfied in order to carry out the task successfully. One of the possibilities to guarantee this constraint is the training displacement, which makes it possible to preserve the distances and the geometrical structure of the group. However, it is necessary to have tools and methods for analyzing and controlling these types of systems in order to make the most of their potential. This thesis is part of this research direction by presenting a synthesis and analysis of multiagent dynamical systems and more particularly the formation control of autonomous vehicles. The control laws developed in the literature for formation control allow to carry out a large number of missions with a high level of performance. However, if a fault/failure occurs in the training, these control laws can be very limited, resulting in unstable system behavior. The development of faulttolerant controls becomes paramount to maintaining control performance in the presence of faults. This problem will be dealt with in more detail in this thesis and will concern the development and design of Faulttolerant controls devolved to a fleet of autonomous vehicles according to different configuration/structuring.

Sommaire

Table des figures

Liste des tableaux

Introduction générale

Un système dynamique multi-agents est un système composé d'une multitude d'entités ou d'agents interagissant entre eux et évoluant dans le temps afin d'effectuer certaines tâches. L'une des grandes sources d'inspiration pour la conception des systèmes multi-agents a été l'étude des comportements sociaux de certaines familles d'insectes et d'animaux communautaires, notamment se déplaçant en formation ou dans une moindre mesure, chassant en meute. Les points communs de ces diverses espèces sont exactement ceux qui caractérisent ce qu'on appelle l'intelligence collective [Hackman11][YaKu09]. On parle également de phénomènes émergents lorsque ces composants arrivent à réaliser des tâches complexes avec un minimum d'intelligence [ViMa11][MaNu06].

Ces dernières années, l'émergence des nouvelles technologies telles que la miniaturisation des composants, les dispositifs de communication sans fils, l'augmentation de la taille de stockage et les capacités de calcul, ont permis la conception de systèmes multi-agents coopératifs de plus en plus complexes [RaKu16][ZhLa13]. L'un des plus grands axes de recherche dans cette thématique concerne la commande en formation de flottes de véhicules autonomes. Un grand nombre d'applications et de missions, civiles et militaires, telles que l'exploration, la surveillance, et la maintenance [HeCo14][ChKu07], ont alors été développées et réalisées dans des milieux variés (terre, air, eau). Durant l'exécution de ces tâches, les véhicules doivent interagir avec leur environnement et entre eux pour se coordonner. Les outils de communication disponibles disposent souvent d'une portée limitée. La préservation de la connexion au sein du groupe devient alors un des objectifs à satisfaire pour que la tâche puisse être accomplie avec succès. Une des possibilités pour garantir cette contrainte est le déplacement en formation permettant de préserver les distances et la structure géométrique du groupe. Il est toutefois nécessaire de disposer d'outils et de méthodes d'analyse et de contrôle de ces types de systèmes afin d'exploiter au maximum leurs potentiels. Cette thèse s'inscrit dans cette direction de recherche en présentant une synthèse et une analyse des systèmes dynamiques multi-agents et plus particulièrement la commande en formation de véhicules autonomes.

Une commande en formation est classée dans la littérature selon trois caractéristiques, son architecture de contrôle (centralisée, décentralisée ou bien distribuée), sa structure (structure en leader-suiveur, virtuelle, et comportementale), et enfin, le type d'approche de contrôle utilisée. Le choix d'un type de commande dépend essentiellement de l'objectif de la mission, de l'environnement et des contraintes de contrôle. Un autre aspect important de la commande en formation concerne la dynamique des véhicules composant la

flotte. On retrouve dans la littérature des commandes développées pour des dynamiques de véhicules linéaires modélisées souvent par un simple et double intégrateur, et d'autre pour des véhicules à dynamique non-linéaires (modèle monocycle, ...). Les preuves de stabilité et de convergence deviennent alors complexes à démontrer. Tous ces aspects seront énoncés plus en détails dans la suite du document.

Contexte et objectifs de la thèse

A la lumière de ce préambule, les problématiques traitées dans ce mémoire de thèse concerneront la synthèse et la conception de commandes pour le vol en formation d'une flotte de véhicules autonomes aériens. Ce sont des problèmes complexes car elles nécessitent des compétences dans plusieurs domaines. L'autre difficulté est que ces problématiques sont traitées de manières différentes par les nombreux chercheurs qui s'intéressent à cette thématique, en fonction de leurs disciplines et de la communauté à laquelle ils appartiennent.

Ce travail de thèse représente l'un des premiers travaux de recherche sur la commande en formation mené au laboratoire CRAN. Notre premier objectif a donc été de nous familiariser avec les problématiques liées à la commande de flotte en effectuant une recherche bibliographique élargie sur le sujet.

La plupart des approches de commande en formation proposées dans la littérature sont développées pour des véhicules évoluant dans des environnements indoor fournissant des conditions de perception et de communication quasi-parfaites. La dynamique des véhicules considérée lors de la conception des lois de commandes est souvent linéaire et représentée dans la plupart des cas par des chaines d'intégrateurs. Cela est loin d'être réaliste et engendrerait de nombreux problèmes dans le cas d'une implémentation directe sur une flotte de véhicules réels. Notre objectif est alors de proposer de nouvelles approches de commande permettant de prendre en considération ces aspects pour une implémentation réelle sur une flotte de quadrirotors.

Toutefois, dans le cas où les performances d'une approche de commande en formation étaient élevées, si un défaut\une défaillance apparaît dans la formation, celle-ci pourrait s'avérer très limitée, engendrant un comportement instable du système. Le développement de commandes tolérantes aux défauts devient alors primordial pour maintenir les performances de commande en présence de défauts. Notre objectif est alors de développer et de concevoir des commandes en formation tolérantes aux défauts dévolues à une flotte de véhicules autonomes suivant différentes configurations/structurations.

Un dernier point consiste, en collaboration avec l'UTC Compiègne, à implémenter les approches de commande proposées sur la plateforme de drones développée par l'équipe de recherche au laboratoire HEUDIASYC. Ceci nous permettra de valider expérimentalement les différents résultats obtenus.

Organisation du manuscrit

Afin de répondre aux objectifs cités précédemment, le manuscrit de thèse est structuré en quatre chapitres :

◇ Le premier chapitre contient les informations essentielles à la compréhension de la commande en formation. Ainsi, les principales classes et approches de commandes de flotte de véhicules autonomes sont illustrées. Ces notions seront ensuite utilisées dans les prochains chapitres pour faciliter leur compréhension.

◇ Dans le deuxième chapitre, une commande fondée sur une approche basée consensus est proposée sur une flotte de véhicules modélisés par des dynamiques en double intégrateur. L'approche représente une extension du modèle développé par [CuDo10], en introduisant un terme pour le contrôle de formation. Les résultats sont ensuite étendus sur une commande d'une flotte de quadrirotors.

◇ Dans le troisième chapitre, une deuxième commande reposant sur une approche basée optimisation est conçue et expérimentée sur une plateforme de drones. Un algorithme d'optimisation, PSO (Particle Swarm Optimization), issu des Métaheuristiques est utilisé dans la génération des trajectoires des UAVs, garantissant la commande en formation. Une nouvelle architecture de commande est proposée. Les expérimentations ont été menées à l'UTC Compiègne, en collaboration avec le laboratoire Heudiasyc.

◇ Le quatrième et dernier chapitre est lié aux problématiques de commandes tolérantes aux défauts. Les approches développées dans le chapitre II et le chapitre III sont alors reprises et étendues dans le cas de présence de défauts. Deux types seront considérés, des défauts liés à la perte d'un ou plusieurs véhicules et des défauts liés à la perte de communication.

Contributions scientifiques

Les divers résultats obtenus au fil de cette thèse ont fait l'objet de plusieurs publications et présentations lors de conférences ou de séminaires :

Revues internationales

- BELKADI, A., OULHADJ, H., TOUATI, Y., et al. On the Robust PID Adaptive Controller for Exoskeletons : A Particle Swarm Optimization Based Approach. Applied Soft Computing, 2017.

- BELKADI, Adel, THEILLIOL, Didier, CIARLETTA, Laurent, ABAUNZA, Hernan, et CASTILLO, Pedro, Design And Implementation Of Distributed Path Planning Algorithm For A Fleet Of UAVs. IEEE Transactions on Aerospace and Electronic Systems. 2017. (soumise)

- ABAUNZA, Hernan, BELKADI, Adel, THEILLIOL, Didier, CIARLETTA, Laurent, et CASTILLO, Pedro. Cylindrical Nested-Saturation Quaternion Control and Distributed PathPlaning for Coordinated Circular Target Tracking on UAV's. 2017. (A soumettre)

Conférences internationales

- BELKADI, Adel, CIARLETTA, L., et THEILLIOL, D. UAVs team flight training based on a virtual leader : Application to a fleet of Quadrirotors. In : Unmanned Aircraft Systems (ICUAS), 2015 International Conference on. IEEE, 2015. p. 1364-1369.

- BELKADI, Adel, CIARLETTA, Laurent, et THEILLIOL, Didier. Particle swarm optimization method for the control of a fleet of Unmanned Aerial Vehicles. In : Journal of Physics : Conference Series. IOP Publishing, 2015. p. 012015.

- BELKADI, Adel, CIARLETTA, Laurent, et THEILLIOL, Didier. UAVs fleet control design using distributed particle swarm optimization : A leaderless approach. In : Unmanned Aircraft Systems (ICUAS), 2016 International Conference on. IEEE, 2016. p. 364-371. Key Bridge Marriott Arlington, USA.

- BELKADI, Adel, THEILLIOL, Didier, CIARLETTA, Laurent, et PONSART J.C, Robust flocking control design for a fleet of autonomous agents. In : Control and Fault-Tolerant Systems (SysTol), 2016 3rd Conference on. IEEE, 2016. p. 1-6.

- BELKADI, Adel, ABAUNZA, Hernan, CIARLETTA, Laurent, et al. Distributed path planning for controlling a fleet of UAVs : application to a team of quadrotors. In : 20th IFAC World Congress, IFAC 2017. Toulouse, France.

Séminaires

- A.BELKADI. Fleet control based on Particular Swarm Optimization : Application to Unmanned Aerial Vehicles. Journées Automatique du GDR MACS. Villeneuve d'Ascq, France, 15-16 Novembre 2016.

- A.BELKADI. Unmanned Vehicles/Multi-Agents Systems/Design of Fault Tolerant Control Methods. Journées Scientifiques Fédération Charles Hermite. Nancy, France, 22 Juin 2017.

Chapitre 1

État de l'art

Sommaire

1.1 Introduction

Ces dernières années, l'émergence des nouvelles technologies a permis la conception de systèmes multi-agents coopératifs de plus en plus complexes. L'un des plus grands axes de recherche dans cette thématique concerne la commande de flotte de véhicules autonomes. Un grand nombre d'applications transverses se rapportent à cette problématique, et ayant comme dénominateur commun la nécessité de se déplacer de façon organisée. Par exemple, un groupe de robots peut être utile pour de la manipulation en équipe ou le transport de gros objets. D'autres applications intéressantes sont l'exploration de surfaces spatiales ou terrestres, la surveillance, le déminage, la recherche et le sauvetage ou encore la cartographie d'environnements. Le déplacement en formation peut également avoir un intérêt énergétique(minimise les coûts de déplacement), environnemental et sociétal (réduire les problèmes de pollution et d'engorgement dans les milieux urbains) dans le domaine des transports, et plus particulièrement la navigation de véhicules mobiles avec une formation qui matérialise un convoi.

La conception et la commande de tels systèmes souvent distribués, restent tout de même complexes et sont confrontées à de nombreuses contraintes liées à des problèmes de communication, d'autonomie et de localisation. L'aspect non- centralisé de ces systèmes ne permet pas également d'avoir une vue globale de l'environnement et rend la détection et la gestion des défauts encore plus difficiles. Il est donc nécessaire et primordial de disposer d'outils et de méthodes d'analyse et de contrôle développés afin d'exploiter au maximum leurs potentiels. Ce premier chapitre présente un ensemble de notions de base essentielles à la compréhension des problématiques liées à la commande des systèmes multi-agents et plus particulièrement à la commande de flotte de véhicules autonomes.

1.2 Définition d'un agent

Les communautés automatique et informatique ont produit diverses définitions d'un agent autonome avec pour chacune ses qualités et ses faiblesses [FrGr97][Roth95]. Néanmoins, toutes ces définitions partagent un ensemble de notions de base : la notion d'agent, son environnement et son autonomie. Selon la définition issue de la communauté informatique citée dans la référence [WoJe95] et qui est suffisamment proche de celle couramment utilisée par la communauté Automatique :

Définition 1. *Un agent est une entité qui peut agir de façon autonome en réponse à des changements de son environnement. Cet environnement présente l'ensemble des éléments (physiques ou logiciels) externes à l'agent.*

Une deuxième définition plus large fournie dans [Ferber99], permet de regrouper les caractéristiques nécessaires de l'agent qui seront adoptées dans ce rapport de thèse :

Définition 2. *Un agent est une entité physique ou virtuelle ayant plusieurs caractéristiques importantes : la réactivité, l'autonomie, la perception et la capacité de communication.*

FIGURE 1.1 – Les systèmes multi-agents

Notons que toutes ces propriétés sont étroitement liées à celles de nombreux animaux de la nature, où chacun d'entre eux perçoit et réagit de façon autonome à son environnement tout en communiquant localement avec ceux qui l'entourent 1.1.

En plus de ces caractéristiques, dans la communauté automatique, un agent est représenté par un modèle mathématique décrivant sa dynamique générale. Ces équations dites alors équations dynamiques, indiquent la relation entre les entrées et les sorties de chaque agent. Dans la littérature, différents modèles ont été utilisés pour représenter la dynamique d'un agent. Considérons un système multi-agent formé par i = 1, ..., N agents. L'état de l'agent i est représenté par $x_i(t) \in R^m$, où m est la dimension du vecteur d'état, et $u_i(t)$ son entrée de commande. Deux grands types de dynamique sont alors disponibles : les modèles linéaires et les modèles non linéaires.

1.2.1 Modèles linéaires

1 – Modèle en simple intégrateur :

Le cas particulier le plus simple pour la modélisation d'un agent dans le cas linéaire est le modèle en simple intégrateur, celui-ci peut être décrit comme suit :

$$\dot{x}_i(t) = u_i(t) \tag{1.1}$$

Différentes applications de commande des systèmes multi-agents utilisent ce type de modèle, notamment dans la commande de rendez-vous [CoMa06][Kha96], commande de formation [MoCa09] et encore beaucoup d'autres [PiKu08][KiSu07]. Dans cette modélisation, les agents se résument en un ensemble de points pouvant se déplacer dans toutes les

directions. Malgré les nombreux résultats théoriques obtenus pour ce type de modélisation, leur application reste tout de même très restreinte.

2 – Modèle en double intégrateur :

Un autre cas particulier de modélisation d'un agent dans le cas linéaire est le modèle en double intégrateur. Celui-ci est largement utilisé dans la littérature. L'une des raisons est que dans de nombreux cas, plusieurs véhicules peuvent être commandés en contrôlant l'accélération de leurs actionneurs. La modélisation d'un agent par double intégrateur peut être décrite par l'équation suivante :

$$\ddot{x}_i(t) = u_i(t) \tag{1.2}$$

Cette modélisation est très utilisée dans la littérature pour décrire la dynamique des agents dans une commande en formation et plus particulièrement dans la commande de flocking [SeDi09][REN08].

3 – Modèle linéaire général :

Le modèle linéaire général d'un agent dans un système multi-agent est décrit comme suit :

$$\begin{aligned} \dot{x}_i(t) &= A_i x_i(t) + B_i u_i(t) \\ y_i(t) &= C_i x_i(t) + D_i u_i(t) \end{aligned} \tag{1.3}$$

Où A, B, C, D sont des matrices de dimensions appropriées. Ce modèle est particulièrement utilisé dans la littérature pour la commande en formation [FaMu04][LiBr04].

Tous ces modèles linéaires offrent d'excellentes propriétés pour l'étude de la stabilité et de convergence, mais sont dans certaines situations trop simples pour décrire la dynamique réelle des agents. Afin de considérer les non linéarités des agents, plusieurs auteurs considèrent des approches avec des dynamiques non linéaires.

1.2.2 Modèles non-linéaires

1 – Modèle monocycle :

Le modèle monocycle est l'une des dynamiques les plus considérées dans le domaine de l'automatique. Celui-ci est un modèle non-holonome utilisé pour décrire différents types de véhicules autonomes (UV) ; mobiles (UGVs), sous-marins (UUVs) ou aériens (UAVs). L'état d'un agent i est décrit par un vecteur $[x_i, y_i, \theta_i]$ où θ_i représente son angle de déviation et (x_i, y_i) ses positions dans le plan. Ce modèle est décrit dans l'équation suivante :

$$\begin{aligned} \dot{x}_i(t) &= v_i(t)cos\theta_i(t) \\ \dot{y}_i(t) &= v_i(t)sin\theta_i(t) \\ \dot{\theta}_i(t) &= u_i(t) \end{aligned} \tag{1.4}$$

Tel que $v_i(t)$ et $u_i(t)$ représentant les entrées de commande du modèle. Ce modèle peut être utilisé dans la commande de formation comme dans [CoMo08][SePa08], le rendez-vous [DiKy07], ou bien le flocking [KwMa10][CoMo06].

2 – Modèle non linéaire général :

Le modèle non-linéaire général est décrit par les équations suivantes :

$$\begin{aligned} \dot{x}_i(t) &= f(x_i(t), u_i(t)) \\ y_i(t) &= g(x_i(t), u_i(t)) \end{aligned} \tag{1.5}$$

Où f et g sont des fonctions non linéaires des vecteurs $x_i(t)$ et $u_i(t)$. Plusieurs applications comme dans [Saif15][JiWa11] utilisent ce modèle.

La conception d'une commande en formation par exemple, dépend essentiellement du type de dynamique considérée pour modéliser les agents de la flotte[ZhSc14][MeKu11]. Celle-ci peut devenir très complexe en terme de preuve de stabilité et de convergence, avec des dynamiques non-linéaires. Une solution intermédiaire souvent utilisée dans le cas d'implémentation réelle est de considérer deux contrôleurs : un contrôleur haut niveau pour la commande de la flotte et un second contrôleur bas niveau local (non-linéaire) pour assurer que l'agent ou le véhicule suive la référence du contrôleur haut niveau.

1.3 Les systèmes multi-agents

En informatique, un système multi-agents (SMA) est un système composé d'un ensemble d'agents autonomes ou partiellement autonomes pouvant être réels ou virtuels, situés dans un certain environnement et interagissant selon certaines relations. Une définition plus large proposée dans [Ferber97] présente un système multi-agents comme :

Définition 3. *Un système multi-agents (ou SMA) est un système composé des éléments suivants :*

- *Un environnement E, c'est-à-dire un espace disposant généralement d'une métrique.*
- *Un ensemble d'objets O. Ces objets sont situés, c'est-à-dire que, pour tout objet, il est possible, à un moment donné, d'associer une position dans E. Ces objets sont passifs, c'est-à-dire qu'ils peuvent être perçus, créés, détruits et modifiés par les agents.*
- *Un ensemble A d'agents, qui sont des objets particuliers ($A \subseteq O$), lesquels représentent les entités actives du système.*
- *Un ensemble de relations R qui unissent des objets (et donc des agents) entre eux.*

- *Un ensemble d'opérations Op permettant aux agents de A de percevoir, produire, consommer, transformer et manipuler des objets de O.*
- *Des opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification, que l'on appelle les lois de l'univers.*

Les agents du système peuvent alors être des agents logiciels, donc virtuels, capable d'interagir entre eux et de coordonner la réalisation d'actions, à l'inverse des agents physiques qui sont des entités opérant dans le monde réel : un véhicule, un avion ou un robot en général. C'est ce concept de système multi-agents qui nous intéresse dans cette thèse et plus particulièrement la commande de flotte de véhicules autonomes.

1.4 La commande de flotte de véhicules autonomes

La commande de flotte de véhicules ou plus généralement la commande multi-robots, suscite l'intérêt de nombreux chercheurs dans le monde appartenant à des communautés différentes, automatique, informatique, robotique. Cet intérêt est dû aux nombreux avantages de ces systèmes qui permettent souvent d'être la solution aux problèmes liées à la commande d'un système unique.

L'étude et la conception d'une commande de flotte restent tout de même complexe. Si plusieurs véhicules doivent se coordonner pour accomplir une mission, celle-ci est plus difficile que dans le cas de la commande d'un seul véhicule, car en plus des décisions individuelles, il faut prendre en considération les interactions et les combinaisons des actions entre les agents. Une grande partie des recherches concernant la commande de flotte se concentre donc sur l'accomplissement d'une seule tâche à la fois. Quelques exemples de ces recherches concernent :

a- La commande de formation (Formation Control) : La commande de formation représente un aspect très important dans la commande de flotte de véhicules autonomes. Dans de nombreuses applications, les véhicules sont amenés à suivre une trajectoire de mission tout en maintenant une configuration spatiale désirée [SuMo14][JiEg07]. Ce déplacement en formation présente alors de nombreux avantages, notamment une meilleure robustesse et efficacité du système, une meilleure capacité de reconfiguration, une grande flexibilité de la structure ainsi qu'un coût minimal [SuMo14][JiEg07]. La configuration des véhicules quant à elle pourrait être régulière ou irrégulière comme observé avec les nuées d'oiseaux ou avec les bancs de poissons [TaJa07].

La figure 1.2 montre un exemple d'application de la commande en formation sur un convoi de camions [LiMa16]. La commande de convoi est un moyen de réduire considérablement la consommation de carburant pour les véhicules suiveurs car la traînée d'air est réduite lorsque l'écart inter-véhicule entre les camions est réduit. Dans cet article, une étude expérimentale est menée pour déterminer comment le trafic peut affecter une manœuvre de fusion de deux camions essayant de former un peloton sur une voie publique pendant les heures de pointe.

FIGURE 1.2 – Trois camions Scania ont été conduits de Sodertalje, en Suède, à Rotterdam, aux Pays-Bas, au cours du défi européen de commande de convoi

La commande de convoi a aussi un intérêt environnemental et sociétal. La figure 1.3 montre la commande d'un convoi de 8 véhicules sur autoroute. Le programme PATH (Partners for Advanced Transit and Highways) est un programme ayant pour mission principale d'appliquer les technologies de pointe pour augmenter à la fois la capacité et la sécurité des autoroutes, tout en réduisant les problèmes de congestion du trafic, de pollution atmosphérique et de consommation énergétique.

FIGURE 1.3 – Convoi de 8 voitures réalisé dans le cadre du projet PATH

b- Le déploiement et la commande de couverture (Coverage Control) : Le but pour ce type de commande est de concevoir une flotte de véhicules capable de se déployer sur une surface de la manière la plus optimum possible [LeDi15][CoMa04]. On retrouve cette approche dans les missions de surveillance ou de recherche [BuCo09]. La figure 1.4 montre un exemple d'application où une flotte de drones est déployée dans

un environnement pour couvrir une zone d'intérêt par des caméras de surveillance embarquées [SaVo16]. Le champ des caméras est visualisé par les pyramides blanches. Les courbes jaunes représentent les trajectoires des drones et les lignes rouges représentent les liens de localisation relatifs.

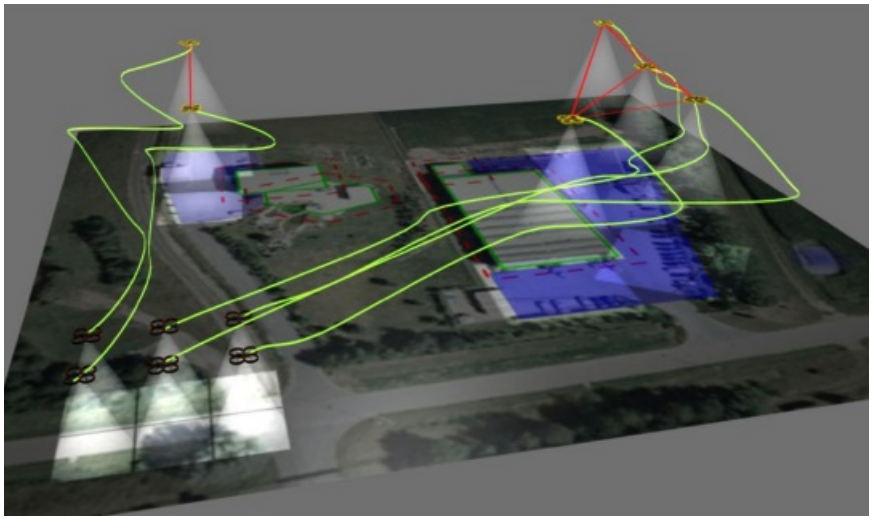

FIGURE 1.4 – Flotte d'UAVs déployée pour couvrir une zone d'intérêt pour la surveillance

Un autre projet très ambitieux concerne la commande d'une flotte de véhicules électriques sur le réseau urbain. L'idée principale de ce projet "VALET" est de concevoir et de développer un système intelligent de redistribution automatique de voitures en autopartage. Il s'agit donc de récupérer les véhicules stationnés de façon aléatoire par les utilisateurs dans le réseau, une fois les véhicules collectés et assemblés en convois, l'objectif est alors de guider ces pelotons vers leur zone de stationnement réservé ou bien à leurs parcs de stationnement respectifs. Les pelotons sont alors démantelés, chaque véhicule se voit attribuer une place de parking vers laquelle il doit se diriger puis se garer de manière autonome 1.5.

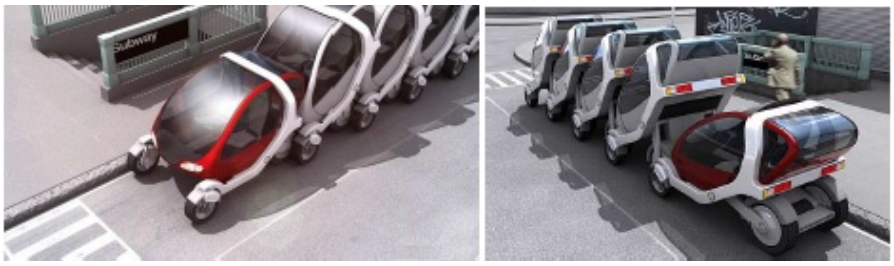

FIGURE 1.5 – Déploiement de véhicules électriques sur un réseau urbain

c- Swarming et Flocking : Le swarming (essaim) désigne un comportement collectif exposé par des entités, en particulier des animaux, de taille similaire qui s'accumulent au même endroit ou se déplacent en masse dans une certaine direction 1.6. Le flocking

désigne plus un mouvement de déplacement d'une flotte en référence aux mouvements des oiseaux 1.7. Le but est alors pour les entités d'arriver à un compromis "un consensus" sur leurs vecteurs vitesse (direction et amplitude) [AhCh12][DaMo11].

FIGURE 1.6 – Swarming

FIGURE 1.7 – Flocking

Il existe encore d'autres problématiques traitées dans la littérature en relation avec la commande de flotte. On retrouve par exemple : **la commande Rendez-vous** ; dans celle-ci il s'agit pour un groupe de véhicules d'arriver à un compromis sur un point de rendez-vous [OlMu07]. Les agents ne disposant uniquement que des informations sur les distances relatives avec les agents de leur voisinage et sans pouvoir se localiser dans l'espace, arrivent à converger vers un consensus sur la position de rendez-vous. D'autres recherches encore s'intéressent aux problématiques liées à l'attribution de tâches **(Task Assignment)** dans le cas de commande de robots coopératifs [MiZa08][ChCa02] et aux problématiques de connectivité et communication dans une flotte, à la synchronisation, aux retards et bien d'autres.

Nous proposons dans les sous-sections suivantes de classer les différentes techniques de commande de flotte selon la structure, l'architecture et l'approche utilisée.

1.4.1 Les architectures de commande de flotte

Dans la commande de flotte, nous pouvons distinguer trois types d'architectures : centralisée, décentralisée et distribuée. Dans ce qui suit, nous présentons successivement ces trois architectures en montrant leurs principaux avantages et inconvénients.

Architecture centralisée

Dans l'architecture centralisée figure 1.8, un seul régulateur permet de commander tous les agents de la formation [BeTi02][RiHo02]. Celui-ci peut être placé à bord d'un agent de la flotte ou bien complètement à distance. Le contrôleur dit centralisé reçoit l'information sur l'état de chaque agent de la formation puis, calcule les entrées de commande appropriées à chacun d'eux et leur transmet ensuite les signaux de commande.

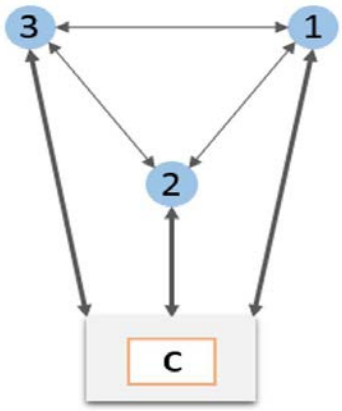

FIGURE 1.8 – Architecture de contrôle centralisée

La conception d'un tel type de contrôleur est assez simple de par la connaissance globale de l'état de tous les agents de la formation, cependant, cette architecture nécessite une puissance de calcul et de communication élevée ce qui impacte l'évolutivité de la formation. L'utilisation d'un seul contrôleur pourrait être aussi un inconvénient dans le cas de présence de défauts.

Architecture distribuée

Dans cette architecture figure 1.9, chaque agent de la formation a son propre contrôleur qui utilise l'information de son état ainsi que celles de ses voisins afin de calculer ses entrées de commande appropriées [HoFa15][TaJa07].

Nous pouvons trouver dans la littérature plusieurs travaux qui utilisent cette architecture de contrôle. Sa robustesse et son évolutivité lui permettent d'être sans aucun doute l'architecture la plus réussie pour le contrôle de formation. Elle reste toutefois peu optimale à cause de la connaissance restreinte de chaque contrôleur.

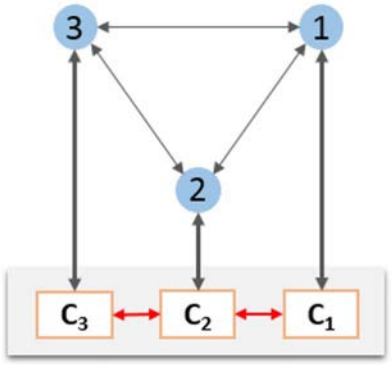

FIGURE 1.9 – Architecture de contrôle distribuée

Architecture décentralisée

Dans cette architecture chaque entité aura son propre contrôleur qui mesure et commande uniquement ses propres états [VaVi14] [Bakul08].

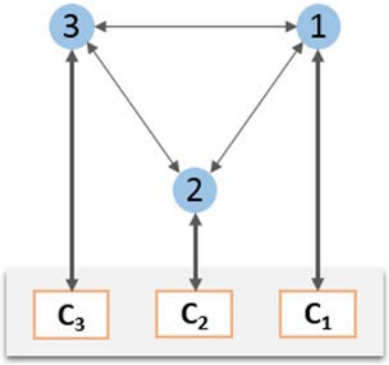

FIGURE 1.10 – Architecture de contrôle décentralisée

Les contrôleurs n'ont alors aucune connaissance sur l'état des autres agents, cette architecture ne peut donc pas être appliquée pour la commande en formation de véhicules autonomes figure 1.10.

1.4.2 Les structures de commande de flotte

Dans la littérature existante, nous pouvons distinguer trois structures de contrôle de formation : la structure leader-suiveur, la structure virtuelle et la structure basée sur le comportement. Nous proposons de décrire ces différentes structures avec leurs principaux avantages et inconvénients.

Structure leader-suiveur

Dans la structure leader-suiveur, les agents de la formation suivent un agent désigné comme le leader. Une trajectoire de mission est définie au sein de ce dernier qui sera suivie par les autres agents dans un déplacement en formation. Deux stratégies ont été mises en œuvre dans cette structure, le mode leader "leader mode", dans laquelle tous les agents de la flotte suivent directement le leader, et le "front mode" [HoFa15], où chaque agent de la formation suit un agent voisin et ainsi de suite jusqu'à atteindre le leader qui lui, conduit toute la formation. Cette structure est simple et largement mise en œuvre dans le contrôle de formation. De plus, l'analyse de stabilité des systèmes utilisant cette structure est relativement simple. Cependant, elle révèle certains inconvénients. L'un d'eux est que toute la formation dépend du leader. Par conséquent si un défaut/défaillance apparait dans celui-ci, toute la formation sera affectée.

Deux solutions d'amélioration sont proposées. Dans la première, le leader est remplacé par un agent virtuel [ShWa06]. En fait, le leader virtuel est une trajectoire de référence envoyée à tous les agents de la formation. Cette solution résout les problèmes de fiabilité et de propagation d'erreur dans le cas d'un leader physique. Dans la deuxième, une solution de leader multiple est proposée pour résoudre le problème de dépendance au leader unique [HoFa15]. Cependant, le problème d'auto-organisation est encore persistant dans cette structure. Aussi, la propagation d'erreur et le problème d'auto-organisation sont des inconvénients majeurs de cette structure.

Structure virtuelle

Dans la structure virtuelle, chaque agent de la formation a sa propre trajectoire à suivre. Les trajectoires globales forment la formation désirée. Les trajectoires sont calculées dans un ordinateur central et envoyées aux agents de la formation. En général, aucune interaction entre agents n'est envisagée.

Cette structure présente certains inconvénients. En effet, comme il n'y a pas d'interaction entre les agents, les collisions ne peuvent être évitées en présence de perturbations. De plus, le contrôle centralisé de cette structure augmente les coûts de calcul et de communication[SoAu10][KuMe13].

Structure basée sur le comportement

Dans la structure basée sur le comportement, chaque agent suit certaines règles pour atteindre la formation, l'objectif est donc décomposé en plusieurs règles de comportements. Cette structure, introduite par [Reynolds87], s'inspire du mouvement collectif des animaux. [Reynolds87] a estimé que chaque individu dans une formation devrait suivre trois règles afin d'obtenir une structure comportementale à savoir : l'évitement des collisions, le consensus en vitesse, et la cohésion. Ces règles conduisent au fait que :

- chaque agent de la formation doit assurer une distance de sécurité prédéfinie avec ses agents voisins ;

- chaque agent essaie de faire correspondre sa vitesse avec les agents proches ;
- chaque agent tente de rester proche de ses voisins.

De plus, chaque agent doit converger vers un objectif global de la formation. Cet objectif pourrait être un point de rendez-vous ou une trajectoire de référence connue de tous les agents de la formation.

Les principaux avantages de la structure comportementale sont l'aspect d'auto organisation (chaque agent doit seulement suivre un ensemble de règles simples), l'évolutivité (observée lorsque le nombre d'agents augmente ou diminue dans la formation), et leur contrôle distribué (comme chaque agent interagit seulement avec ses voisins, ses capacités de calcul ne sont pas affectées. L'interaction avec les voisins montre cet aspect de contrôle distribué). Cette structure comporte néanmoins certains inconvénients, dont l'analyse de stabilité qui est relativement difficile, ainsi que l'incapacité d'assurer une forme géométrique fixe de la formation,seuls des profils en essaim avec des inter-distances fixes entre agents pourraient être formés.

1.4.3 Les approches de commande de flotte

Outre le choix d'une des trois topologies mentionnées pour le contrôle des systèmes multi-agents, nous nous intéressons maintenant aux outils théoriques utilisés pour synthétiser l'action de commande. Nous présentons ci-dessous les idées principales des méthodes basées sur des algorithmes de consensus et d'autres utilisant les techniques d'optimisation.

Approche basée consensus

Lorsque plusieurs agents interagissent et échangent des données, il est nécessaire de s'accorder sur des valeurs communes. Ainsi, le problème du consensus joue un rôle central dans l'étude des systèmes multi-agents et plus particulièrement dans le contrôle de formation. La plupart des articles traitant du problème du consensus utilisent une approche appelée analysis-based approach [ReBe05][SaMu04]. Cette approche permet de déterminer, selon un modèle défini, les conditions suffisantes pour garantir certaines propriétés du système multi-agent telle que la stabilité ou la convergence vers un consensus. Récemment, certains auteurs ont introduit une nouvelle approche appelée design-based method [JiLi14][KaKh09]. Celle-ci se concentre sur la façon de concevoir des contrôleurs permettant d'atteindre les exigences spécifiées. La théorie des graphes représente alors l'outil théorique principal utilisé dans ces approches, où les agents sont représentés par des nœuds dans un graphe et les interactions entre les agents par des arcs entre ces nœuds.

Approche basée sur l'optimisation

Une commande basée sur l'optimisation est un terme général qui se réfère à la conception d'un régulateur en utilisant un critère d'optimisation et des techniques de résolution pour obtenir les paramètres de la loi de commande. L'optimalité étant généralement équivalente à une certaine propriété souhaitée, par exemple, la stabilité, la réactivité ou la robustesse [Murray09] [OlDu09]. Cette définition assez large peut couvrir les commandes

optimales classiques, les techniques basées LMI, ou la commande prédictive. L'avantage avec ces approches dans le contrôle multi-agent est que l'objectif est formulé comme un problème d'optimisation, en fournissant des outils efficaces pour trouver la solution optimale par rapport aux critères considérés [OlGr15][CaBo13]. Une autre méthode connue est la méthode du champ potentiel, dans laquelle une fonction de potentiel scalaire positif est construite de telle sorte que son minimum est obtenu lorsque l'agent atteint son but. En dehors de la configuration idéale, cette fonction évolue de sorte que l'agent peut atteindre l'objectif en suivant le gradient négatif du potentiel [Prodan12][Kodirschek92].

On retrouve également dans cette classe, une méthode d'optimisation utilisée dans la commande des systèmes multi-agents basée sur l'optimisation par méta-heuristique. Les systèmes de contrôle par méta-heuristique, tels que les algorithmes génétiques [DePr02], les algorithmes de colonies de fourmis [DoBi08] ou l'optimisation par essaims de particules [AlMo11][Kennedy10] ont fait l'objet de plusieurs études dans la littérature. Nous nous intéresserons particulièrement sur ce type d'approche dans le chapitre III.

1.5 Communication et interaction dans la flotte

L'interaction entre les agents peut être exprimée en terme de topologie de communication, la théorie des graphes vient alors comme une approche naturelle quand on traite avec des systèmes multi-agents [MeEg10][JaLI03]. L'échange d'informations entre les agents est caractérisé au moyen d'un graphe. Un graphe G est un couple (N, ε) consistant en un ensemble de nœuds $N = 1, ..., n$ et un ensemble d'arcs $\varepsilon \in N \times N$. Lorsque $(i, j) \in \varepsilon$, on dit que i est un voisin entrant de j et j est un voisin sortant de i. Un chemin de i à j est une séquence de nœuds $(i_1, i_2, ..., i_p)$ telle que $i_1 = i$, $i_p = j$ et $\forall k \in 1, ..., p-1\,;\, (i_k, i_k+1) \in \varepsilon$. $p-1$ est alors appelé sa longueur. La distance entre deux nœuds i et j est la longueur du plus court chemin allant de i à j. Un graphe est dit fortement connexe lorsque pour tout couple (i, j), il existe un chemin de i à j dans le graphe. Un sous graphe couvrant de G est un graphe $G' = (N, \varepsilon')$ tel que $\varepsilon' \subseteq \varepsilon$. Un cycle est un chemin de i à i. Un arbre est un graphe sans cycle tel qu'il existe un nœud r tel que pour tout $j \in N\{r$, il existe un chemin de r à j ; un tel nœud est appelé racine de l'arbre. La profondeur D de l'arbre est la longueur maximale de l'ensemble des plus courts chemins de la racine r au nœud i, pour $i \in N$.

Un graphe pondéré G est un triplet (N, ε, A) où (N, ε) est un graphe et A une matrice de $M_n(R)$ telle que $a_{ij} > 0$ ssi : $(j, i) \in \varepsilon$. Un graphe pondéré canonique est un graphe pondéré ayant une matrice A tel que : $a_{ij} = 1$ ssi : $(j, i) \in \varepsilon$ et 0 autrement. Cette matrice A est appelée la matrice d'adjacence du graphe. Le degré entrant d'un nœud i est $\sum_{j \in N} a_{ij}(t)$ noté d_i^-. Le degré sortant d'un nœud i est $\sum_{j \in N} a_{ji}(t)$ noté d_i^+. Un graphe est équilibré lorsque pour tout $i \in N$, $d_i^- = d_i^+$. D^+, D^- sont les matrices diagonales dont le $i - me$ coefficient diagonal est d_i^+ et d_i^-. La matrice Laplacienne du graphe est définie par :

$$L = D^- - A \tag{1.6}$$

La figure 1.11 permet d'illustrer un exemple d'un graphe pondéré :

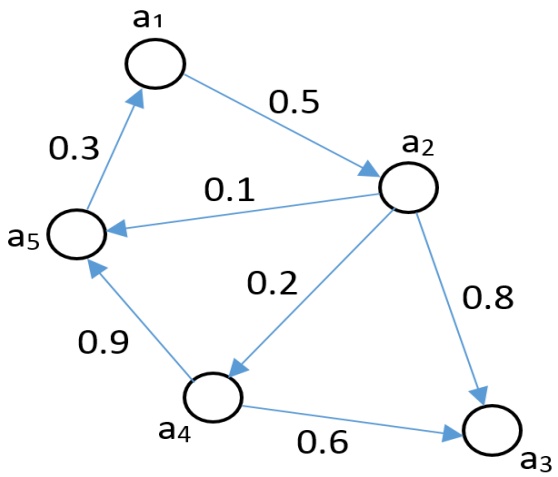

FIGURE 1.11 – Schéma d'un graphe pondéré

Les matrices A, D^+ et D^- sont de dimension 5×5 telles que :

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0.3 \\ 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0.8 & 0 & 0.6 & 0 \\ 0 & 0.2 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0.9 & 0 \end{bmatrix}, D^+ = \begin{bmatrix} 0.5 & 0 & 0 & 0 & 0 \\ 0 & 1.1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.5 & 0 \\ 0 & 0 & 0 & 0 & 0.3 \end{bmatrix}, D^- = \begin{bmatrix} 0.3 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 1.4 & 0 & 0 \\ 0 & 0 & 0 & 0.2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Dans ce cas la matrice Laplacienne est égale à :

$$L = \begin{bmatrix} 0.3 & 0 & 0 & 0 & -0.3 \\ -0.5 & 0.5 & 0 & 0 & 0 \\ 0 & -0.8 & 1.4 & -0.6 & 0 \\ 0 & -0.2 & 0 & 0.2 & 0 \\ 0 & -0.1 & 0 & -0.9 & 1 \end{bmatrix}$$

On s'intéresse spécialement aux spectres de la matrice Laplacienne L, on note les valeurs propres de L par ordre croissant sur leurs parties réelles :

$$Re(\lambda_1(L)) \leq Re(\lambda_2(L)) \leq ... \leq Re(\lambda_5(L)) \tag{1.7}$$

L'ensemble des valeurs propres de L, est noté $Sp(L)$ spectre du graphe G. On appelle $\lambda_2(L)$, dans le cas d'un graphe non dirigé, la connexité algébrique du graphe G [ReBe08]. Celle-ci permet de mesurer la qualité de la connexité du graphe et possède les propriétés suivantes :

- $Re(\lambda_2(L)) > 0$ si le graphe possède un sous graphe couvrant.
- Pour G et G' deux graphes pondérés symétriques de matrice Laplacienne L et L', si G' est un sous graphe couvrant de G alors $\lambda_2(L) > \lambda_2(L')$.

1.6 Détection de défauts et commande tolérante aux défauts

La sécurité d'un système, composée des méthodes de détection et d'isolation des défauts (FDI) et des commandes tolérante aux défauts (FTC), devient aujourd'hui un champ de recherche intensif dans le domaine de l'automatique et qui doit être prise en compte dans la conception des lois de commande.

Le diagnostic de défauts peut être défini comme la procédure qui consiste à détecter, localiser et identifier un composant ou un élément défectueux dans un système dynamique [DeIs01][Patton94]. Trois classes de défauts sont définies dans la littérature :

- les défauts actionneurs qui agissent au niveau de la partie opérative et qui détériorent le signal de commande du système ;
- les défauts capteurs qui sont la cause d'une mauvaise mesure ou estimation de l'état du système ;
- les défauts composants ou systèmes qui eux proviennent du système lui-même et résultent de l'altération d'un composant du système.

Parmi les méthodes de diagnostic de défauts existantes, on retrouve les approches de diagnostic à base de modèle. Dans celles-ci, il est primordial de disposer d'un modèle mathématique le plus précis possible du système. La première étape dans ces approches consiste à générer des indicateurs de défauts (résidus). Ils contiennent des informations sur les anomalies du système à commander. Le principe est de mesurer l'écart entre les mesures des signaux capteurs ou actionneurs, et la valeur théorique fournie par le modèle dans les conditions de fonctionnement nominal [Ripoll99][Henry99]. Les méthodes à l'aide d'observateurs sont très répandues dans la littérature [AlSh08][NaVa08].

L'objectif ensuite de la commande tolérante aux défauts est de déterminer une stratégie de commande qui permet d'annuler, ou au moins de limiter les effets de défauts sur la stabilité et les performances du système. En présence d'un défaut de faible amplitude, une simple commande robuste peut préserver la stabilité et les performances nominales du système à commander, on parle alors de commande tolérante aux défauts passive [NiSt03][Zhou00]. Par contre, en présence de défauts critiques, il est nécessaire de mettre en œuvre une stratégie de commande tolérante aux défauts active. Dans ce cas, on distingue l'accommodation [BlKi03], la reconfiguration [BoMe03][KaVe02] et la restructuration [StGe00] suivant les performances souhaitées après l'occurrence de défaut. Cette approche permet alors de traiter des défauts imprévus mais elle requiert de synthétiser un schéma de FDI permettant de fournir de manière aussi précise que possible une information sur les défauts éventuels ainsi qu'un modèle de défaut du système. Le problème de la commande tolérante aux défauts a été largement abordé ces deux dernières décennies et a fait l'objet d'un nombre important de résultats expérimentaux. Le problème majeur rencontré lors de la conception de telles lois de commandes est que la plupart des techniques FDI sont utilisées comme un outil de surveillance pour détecter et localiser les défauts en boucle ouverte et n'intègrent pas la partie commande.

L'étude des systèmes multi-agents et particulièrement la commande de flotte de véhicules autonomes nécessite, par rapport à la commande d'un système unique, la prise en compte de problématiques supplémentaires et qui peuvent compromettre l'exécution d'une mission donnée, tels que des pertes de communication, des modifications dans la topologie du réseau, des pertes de données entre les agents, l'évitement de collisions, des défauts ou des pertes d'agents dans la flotte ou encore des problèmes de propagations d'erreurs. Tous ces aspects sont considérés dans le cadre de commande de flotte comme des défauts dans le système multi-agents. Bien que l'étude des méthodes FDI et FTC pour un système dynamique unique soit largement connue et développée, peu de résultats se rapportent au concept des systèmes multi-agents. Les notions de sécurité et tolérance aux défauts dans le cadre de la commande en formation seront étudiées plus en détails dans le chapitre IV.

1.7 Conclusion

La commande de flotte de véhicules autonomes représente aujourd'hui un intérêt considérable qui attire de nombreux chercheurs dans le monde. De nombreux travaux ont été réalisés ces dernières années dans le but de développer et de comprendre ce type de systèmes. Nous avons présenté dans ce premier chapitre de thèse un état de l'art sur les principales approches et types de commande de flotte existants dans la littérature. Des notions importantes et essentielles à l'étude de ces systèmes et à la compréhension des prochains chapitres sont également présentées.

Nous nous intéressons dans les prochaines parties à la conception de commandes de flotte distribuées pour le vol en formation d'un groupe d'UAVs. Des problématiques liées à la détection de défauts\défaillances seront étudiées. Nous nous intéresserons particulièrement aux cas de pertes d'agents dans la flotte et aux défauts actionneurs. Des résultats de simulation et expérimentaux seront enfin présentés sur la commande de quadrirotors en flotte.

Chapitre 2

Commande de flotte basée consensus

Sommaire

2.1 Introduction

Un système dynamique multi-agents est un système composé d'une multitude d'entités ou d'agents interagissant ensemble qui évoluent dans le temps dans le but d'effectuer certaines tâches. Lorsque plusieurs agents interagissent et échangent des données, il est souvent nécessaire qu'ils puissent s'accorder sur des valeurs communes, un consensus. Celui-ci peut être un objectif commun à atteindre, une vitesse d'évolution, une distance à respecter, une répartition de la charge de travail, etc. Le processus de consensus apparait dans plusieurs phénomènes naturels, allant des cellules microscopiques aux mouvements collectifs d'animaux vivant en groupe. Un des exemples de mouvement collectif est celui formé par les nuées d'étourneaux. On peut les voir former un amas aérien se déplaçant comme s'il s'agissait d'un unique individu. Le moyen de communication, ici la vision, permet aux oiseaux de s'accorder sur la direction et la vitesse à adopter : il s'agit d'un processus de consensus dont l'objet est la vitesse de vol. Des comportements similaires peuvent être constatés chez les bancs de poissons et les troupeaux de certains mammifères. Ainsi, le problème de consensus joue un rôle central dans l'étude et la conception de commandes en formation.

Nous nous intéressons dans ce chapitre à la commande de flotte basée sur les algorithmes de consensus. Il existe à ce sujet une bibliographie très dense. Notre objectif sera donc de concevoir à partir de modèles existants, une commande de flotte pour le vol en formation d'un groupe d'UAVs. Les drones devront alors se déplacer en formation dans un environnement inconnu en évitant les collisions. Il s'agira également pour les UAVs d'atteindre un consensus sur leurs vitesses d'évolution. Cela permettra entre autres d'accomplir un mouvement de flotte plus fluide et plus robuste aux perturbations. Dans les sections suivantes différents modèles de consensus développés dans la littérature sont présentés. Nous nous intéresserons ensuite à un algorithme en particulier développé dans [CuSm07] et [CuDo10]. Ce dernier fournit de solides résultats mathématiques sur la convergence vers un consensus en vitesse et qui assure simultanément l'évitement de collision entres les agents de la flotte. Une extension de ce modèle est proposée pour la commande d'une flotte de quadrirotors.

2.2 Les algorithmes de Rendez-vous

Il est question ici d'étudier les algorithmes de consensus linéaires. Il s'agit pour un groupe d'agent d'arriver à un consensus sur un point de rendez-vous. Les agents ne disposant uniquement que des informations sur les distances relatives avec les agents de leur voisinage, et sans pouvoir se localiser dans l'espace, arrivent à converger vers un consensus sur la position de rendez-vous.

Le cas particulier le plus simple est de considérer les agents de la flotte comme un ensemble de points pouvant se déplacer dans toutes les directions. Les agents sont alors modélisés avec des dynamiques en simple 1.1 ou double intégrateurs 1.2.

2.2.1 Algorithme de Rendez-vous pour simple intégrateur

Formulation et description

Le cas particulier le plus simple pour la modélisation d'un agent dans le cas linéaire est le modèle en simple intégrateur. Chaque agent de la flotte peut donc être décrit par l'équation 1.1. Afin d'atteindre un consensus sur l'état x_i entre tous les agents, [SaMu04] propose un algorithme de consensus où chaque agent contrôle la dérivée première de son état, telle que :

$$\dot{x}_i(t) = \sum_{j \in N} a_{ij}(t)(x_j(t) - x_i(t)), i \in N \tag{2.1}$$

Où $x_i(t)$ est la position de l'agent i évoluant dans R^d, un espace vectoriel de dimension finie d. $a_{ij}(t) \geq 0$ représente l'influence de l'agent j sur l'agent i au temps t. $N = 1, ..., n$ désigne l'ensemble des labels des agents, où n est le nombre d'agents. L'architecture de commande est donc distribuée du fait que chaque agent i dispose uniquement des informations de ses voisins modélisées par les fonctions $a_{ij}(t)$.

Ce modèle de consensus linéaire résume l'essentiel du processus de consensus dans les mouvements collectifs, où les individus ont tendance à se diriger en direction de leurs voisins pour réduire l'écart qui les sépare. De plus, ce modèle est suffisamment simple pour être analysé en détail.

En exploitant les outils de la théorie des graphes, l'équation de consensus 2.1 peut s'écrire sous la forme suivante :

$$\dot{x}_i(t) = \sum_{j \in N} a_{ij}(t)x_j(t) - \sum_{j \in N} a_{ij}(t)x_i(t), i \in N \tag{2.2}$$

En posant le vecteur $X(t) = [x_1(t), \ldots, x_n(t)]^T$ contenant les états de tous les agents et en introduisant les matrices A et D^- définies dans la chapitre I, on obtient :

$$\dot{X}(t) = A(t)X(t) - D^-(t)X(t) \tag{2.3}$$

$$\dot{X}(t) = (A(t) - D^-(t))X(t) \tag{2.4}$$

alors :

$$\dot{X}(t) = -L(t)X(t) \tag{2.5}$$

Où L est la matrice Laplacienne. Cette écriture simplifiée est couramment utilisée dans l'analyse et la commande des systèmes multi-agents.

Nous supposons dans l'équation précédente 2.5 que les positions des agents évoluent de manière continue dans le temps. Une alternative serait de choisir une évolution discrète avec $t \in N$, ce qui est plus adapté pour l'implémentation temps réel et facilite

souvent l'analyse du comportement du système [ReMo16]. L'équation 2.1 est réécrite sous la forme :

$$x_i(k+1) = x_i(k) + \epsilon \sum_{j \in N} a_{ij}(k)(x_j(k) - x_i(k)) \tag{2.6}$$

on obtient alors :

$$X(k+1) = P(k)X(k) \tag{2.7}$$

Avec $P(k)$ une matrice stochastique $n \times n$. Ce système est considéré comme la représentation en temps discret du système de consensus. A partir de ces développements, en utilisant la structure de la théorie des graphes, l'écriture des systèmes de consensus peut être simplifiée pour obtenir les deux équations 2.5 et 2.7. Ces écritures servent uniquement à l'analyse des systèmes de consensus.

Analyse et étude du système de consensus

Déterminons à partir de l'équation 2.5, les conditions suffisantes sur les poids d'interaction $a_{ij}(t)$ au cours du temps garantissant la convergence de la trajectoire du système vers un état de consensus.

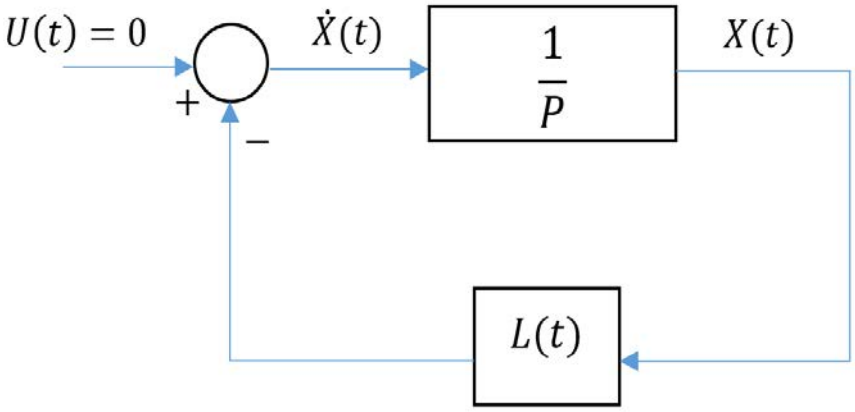

FIGURE 2.1 – Schéma de commande de l'algorithme de consensus

Par définition, la trajectoire du système converge vers un consensus lorsque :

$$\lim_{t \longmapsto \infty} x_i(t) = x^*, \tag{2.8}$$

Où x^* est l'état final commun à tous les agents, appelé valeur de consensus.

L'étude du système consensus doit passer par deux points essentiels, le premier étant la question d'existence et d'unicité de la solution du système, une fois ce point traité, vient la question de convergence du système vers un consensus.

1 – **Existence et unicité**

Soit le système suivant tel que :

$$\dot{x}(t) = f(t, x(t)), t \in R^+ \tag{2.9}$$

Avec $x(0) = x_0 \in R^n$.

Théorème 1. *Si la fonction f est continue par morceaux en t telle que :*

$$\exists l > 0; \forall x, y \in R^n, \forall t > 0, \|f(t, x(t)) - f(t, y(t))\| \leq l\|x - y\| \tag{2.10}$$

Alors $\exists!$ fonction $x(t)$ dérivable sur R^+ solution de l'équation 2.9. [Leip13].

En appliquant le théorème 1 au système de consensus continu 2.1, le résultat suivant est obtenu :

En considérant des poids d'interaction pour tout $i, j \in N$, $t \to a_{ij}(t)$ continues par morceaux, alors le système de consensus 2.1 admet une unique solution dans R [Khalil02]

Dans le cas où les fonctions $a_{ij}(t)$ sont mesurables mais non pas continues, alors le système n'est pas bien défini et sa forme intégrale est préférée :

$$\int_0^t \dot{x}_i(s)ds = \int_0^t \sum_{j \in N} a_{ij}(s)(x_j(s) - x_i(s))ds, i \in N \tag{2.11}$$

$$x_i(t) = x_i(0) + \int_0^t \sum_{j \in N} a_{ij}(s)(x_j(s) - x_i(s))ds, i \in N \tag{2.12}$$

La solution du système est dite solution du système au sens de Carathéodory [FiAr88]. Si pour tout, $j \in N$,les fonctions $t \to a_{ij}(t)$ sont sommables sur tout intervalle borné de R^+, alors la solution au sens de Carathéodory pour le système existe et est unique.

2 – **Convergence vers un consensus**

Comme déjà énoncé, le but de l'approche considérée pour les systèmes de consensus est de déterminer les conditions les moins contraignantes et suffisantes sur les poids d'interaction $a_{ij}(t)$ au cours du temps garantissant la convergence des positions des agents du système vers un état de consensus. Une des conditions suffisantes les plus connues dans la littérature est celle démontrée dans [JaLi03]. Elle consiste à supposer que les poids d'interaction entre les agents sont fortement symétriques $a_{ij}(t) = a_{ji}(t)$ et bornés et que l'union des graphes d'interaction $(N, \varepsilon(t))$ est connexe sur tout intervalle borné. Depuis lors un grand nombre d'extensions et de généralisations ont été établies. Les auteurs dans [SaMu04] ont démontré que le système de consensus en temps continu 2.1 converge vers un consensus si son graphe est équilibré $\forall t \in R^+$ et que ses poids d'interaction $a_{ij}(t)$

soient à valeurs dans $\{0, 1\}$. Les détails de celui-ci sont présentés dans ce qui suit :

Discussion : On considère le système de consensus en temps continu. On suppose que son graphe est équilibré $\forall t \in R^+$ et les poids d'interaction $a_{ij}(t)$ sont à valeurs dans $\{0, 1\}$.

On définit alors un vecteur $x^* = (x_1 + x_2 + ... + x_n)/n$ comme étant la position moyenne initiale des agents.

Comme le graphe est équilibré, la valeur moyenne des positions est invariante au cours du temps. Cela signifie que si la trajectoire converge vers un consensus, la valeur de consensus est nécessairement x^*.

On définit un autre vecteur appelé vecteur de désaccord $\delta(t)$ tel que :

$$\delta(t) = [\delta_1(t)^T, \delta_2(t)^T, ...\delta_1(t)^T]^T \tag{2.13}$$

avec $\delta_i(t) = x_i - x^*$.

On dit que la trajectoire du système converge vers un consensus lorsque :

$$\forall i \in N, \lim_{t\to\infty} \delta_i(t) = 0 \Rightarrow \lim_{t\to\infty} x_i(t) = x^*$$

Démonstration de la convergence vers un consensus :

On pose $V(\delta)$ fonction de Lyapunov du système de consensus tel que : $V(\delta) = \frac{1}{2}\|\delta\|^2$, avec :

$$V(\delta(x^*)) = 0 \tag{2.14}$$

$$V(\delta) > 0, \forall x \in R^n - x^* \tag{2.15}$$

Développement :

$$V(\delta) = \frac{1}{2}\|\delta\|^2 = \frac{1}{2}[(x_1 - x^*)^2 + ... + (x_n - x^*)^2] \tag{2.16}$$

$$V(\delta) = \frac{1}{2}\|\delta\|^2 = \frac{1}{2}[\delta_1^2 + ... + \delta_n^2] \tag{2.17}$$

$$\frac{dV(\delta)}{dt} = \frac{dV(\delta)}{d\delta}\frac{d\delta}{dt} = (\frac{dV(\delta)}{d\delta_1}, ..., \frac{dV(\delta)}{d\delta_n})\dot{\delta}(t) \tag{2.18}$$

sachant que : $\dot{\delta}(t) = -L\delta(t)$, on obtient :

$$\frac{dV(\delta)}{dt} = (\delta_1, ..., \delta_n)(-L)\delta = -\delta^T L\delta \tag{2.19}$$

Proposition : Si L est une matrice Laplacienne d'un graphe pondéré équilibré, alors la matrice $\frac{L+L^T}{2}$ est une matrice Laplacienne d'un graphe pondéré symétrique telle que :

$$\delta^T \frac{L + L^T}{2} \delta = k\|\delta\|^2 \tag{2.20}$$

Avec $k > 0$ la connexité algébrique du graphe

Proposition : Toute matrice carrée L peut être écrite sous la forme : $L = L_s + L_a$, avec $L_s = \frac{L+L^T}{2}$ la matrice symétrique de L et $L_a = \frac{L-L^T}{2}$ la matrice antisymétrique de L.

On peut donc écrire :

$$\frac{dV(\delta)}{dt} = -\delta^T(L_s + L_a)\delta = -[\delta^T L_s \delta + \delta^T L_a \delta] \tag{2.21}$$

Où : $\delta^T L_a \delta = 0$

Preuve : Soit un scalaire $\delta^T L_a \delta$, le transposé de h est h^T tel que :

$$h^T = (\delta^T L_a \delta) = \delta^T L_a^T \delta = \delta^T L_a \delta = -h$$

donc : $h^T = -h \Rightarrow h = 0$

Alors :

$$\frac{dV(\delta)}{dt} = -\delta^T L_s \delta = -[\delta^T \frac{L + L^T}{2} \delta] = -k\|\delta\|^2 \tag{2.22}$$

Avec $k > 0$

On obtient alors que $\frac{dV(\delta)}{dt} < 0$, alors le système est asymptotiquement stable.

Enfin pour constater la vitesse de convergence, on a :

$$\frac{dV(\delta)}{dt} = -k\|\delta\|^2 = -2k\frac{\|\delta\|^2}{2} = -2kV(\delta) \tag{2.23}$$

$$\frac{dV(\delta)}{V(\delta)} = -2kdt \Rightarrow V(\delta) = e^{-2kt}V(\delta(0)) = \frac{1}{2}e^{-2kt}\|\delta(0)\|^2 \tag{2.24}$$

$\Rightarrow \|\delta(t)\|^2 = e^{-kt}\|\delta(0)\|^2$ ce qui monte que la trajectoire converge de manière exponentielle vers le consensus.

L'obtention d'un consensus pourrait éventuellement être classée selon qu'on prenne comme conditions initiales sur les poids d'interaction les notions de symétrie et d'équilibre ou pas. La considération de ces hypothèses facilite l'analyse de convergence et de plus nous permet de relaxer l'hypothèse de connexité des interactions [Moreau04][Cao11].

Les valeurs propres de la matrice Laplacienne du graphe pondéré L sont alors à valeurs réelles positives telles que :

$$0 = Re(\lambda_1(L)) \leq Re(\lambda_2(L)) \leq ... \leq Re(\lambda_n(L)) \tag{2.25}$$

Dans le cas d'un graphe symétrique $\lambda_2(L)$ est appelée la connexité algébrique et permet d'estimer le taux de convergence du graphe L et sa connectivité.

2.2.2 Algorithme de Rendez-vous pour double intégrateur

Un autre cas particulier des systèmes linéaires est le modèle en double intégrateur 1.2. Ce modèle correspond au comportement des véhicules autonomes, car ces systèmes sont commandés dans la plupart des cas par leur accélération et non par leur vitesse. De plus, par des transformations dans les lois de commande, ces systèmes peuvent être réduits à un modèle en double intégrateur. Un des algorithmes de consensus pour ce type de modélisation est représenté dans [XiWa07] dans l'équation suivante :

$$\ddot{x}_i(t) = \sum_{j \in N} a_{ij}(t)(x_j(t) - x_i(t)) + \beta \dot{x}_i(t) \tag{2.26}$$

Où β est un gain positif. Suivant le même développement que précédemment, 2.26 s'écrit sous la forme :

$$\ddot{X}(t) = -LX(t) - \beta \dot{X}(t) \tag{2.27}$$

Le consensus est alors atteint si pour toute valeur initiale $x_i(0)$ et $\dot{x}_i(0)$ on a :

$$\forall i, j \in N, \lim_{t \to \infty} x_i(t) - x_j(t) = 0 \tag{2.28}$$

$$et \lim_{t \to \infty} \dot{x}_i(t) = 0 \tag{2.29}$$

Un autre algorithme de consensus a été développé pour le même modèle d'agent présenté dans l'équation suivante :

$$\ddot{x}_i(t) = \sum_{j \in N} [a_{ij}(t)(x_j(t) - x_i(t)) + \beta(\dot{x}_j(t) - \dot{x}_i(t))] \tag{2.30}$$

ce qui conduit à l'équation :

$$\ddot{X}(t) = -LX(t) - \beta L\dot{X}(t) \tag{2.31}$$

Encore une fois, le consensus est atteint si pour toute valeur initiale $x_i(0)$ et $\dot{x}_i(0)$:

$$\forall i, j \in N, \lim_{t \to \infty} x_i(t) - x_j(t) = 0 \tag{2.32}$$

$$et \lim_{t \to \infty} \dot{x}_i(t) - \dot{x}_j(t) = 0 \tag{2.33}$$

Notons que la valeur du gain β et la topologie du graphe décrivant les communications entre les agents à travers les fonctions $a_{ij}(t)$ affectent directement la convergence vers le consensus. Ce type de dynamique permet d'atteindre un consensus en vitesse des agents du système. Ce cas particulier de consensus est appelé Flocking. Il est par définition inapplicable sur une dynamique en simple intégrateur. Il existe dans la littérature un grand nombre de contributions sur les algorithmes de consensus considérant comme dans [CaIs14][AcIs12][ReBe05] des changement de topologie de la formation, des retards [SeDi08][BlFe08] ou bien des incertitudes [KiSh11]. Dans ce chapitre de thèse nous nous sommes particulièrement intéressés aux algorithmes de flocking garantissant une commande de formation qui assure la cohésion de la flotte tout en évitant les collisions entre les agents.

2.3 Les algorithmes de flocking

Dans la commande de flotte, les véhicules de la flotte doivent se déplacer selon un mouvement cohérent. Ce type de déplacement est appelé flocking. D'après Reynolds [Reynolds87], le premier à avoir proposé un modèle multi-agent décentralisé simple, un mouvement est dit de type flocking lorsqu'il vérifie les trois caractéristiques suivantes :

- la séparation : les agents n'entrent pas en collision ;
- la cohésion : les agents restent à proximité les uns des autres ;
- l'alignement en vitesse : tous les agents observent une vitesse identique.

L'implémentation du comportement de flocking peut être décrite par les équations :

$$\begin{aligned} \dot{x}_i(t) &= v_i(t) \\ \dot{v}_i(t) &= alignement_i(t) + repulsion_i(t) + attraction_i(t) \end{aligned} \tag{2.34}$$

Où $x_i(t)$, $v_i(t) \in R^d$ représentent respectivement la position et la vitesse de l'agent i à l'instant $t \in R^+$.

- Le terme $alignement_i(t)$ est habituellement implémenté par un terme de consensus linéaire sur les vitesses.
- Les termes $repulsion_i(t)$ et $attraction_i(t)$ sont souvent fusionnés et implémentés par le gradient d'un potentiel attractif à longue portée et répulsif à courte distance.

L'objectif de la commande est de déterminer les conditions suffisantes pour la convergence de la trajectoire du système vers un état de consensus dépendant uniquement de la configuration initiale (positions et vitesses) des agents. Les poids d'interaction $a_{ij}(t)$ quant à eux sont des fonctions des positions des agents au temps t.

Nous présentons dans la suite de ce chapitre deux algorithmes de flocking parmi les plus utilisés dans la littérature[Saber06][CuSm07]. Notre contribution majeure porte sur la proposition d'une extension des algorithmes de flocking existants pour la commande en formation des agents de la flotte.

2.3.1 Modèle d'Olfati-Saber

L'algorithme de flocking proposé par Olfati-Saber [Saber06] considère les trois termes de flocking : l'alignement en vitesse, l'attraction et la répulsion. Le terme d'alignement en vitesse est une commande de consensus sur les vitesses, et les termes d'attraction et répulsion sont définis via un potentiel de rappel :

$$\begin{aligned} \dot{x}_i(t) &= v_i(t) \\ \dot{v}_i(t) &= \sum_{j \in N} a_{ij}(t)(v_j(t) - v_i(t)) - \nabla_{xi} V(\|x_i(t) - x_j(t)\|) \end{aligned} \tag{2.35}$$

avec :

$$a_{ij}(t) = 1 si \|x_i(t) - x_j(t)\| < R$$
$$a_{ij}(t) = 0 si \|x_i(t) - x_j(t)\| \geq R$$

Où V est une fonction continue sur R^+ telle que V soit décroissante sur $[0, r]$, croissante sur $]r, R]$, et constante sur $]R, +\infty[$. Avec $r > 0$ la distance souhaitée entre les agents et $R > r$ est le rayon maximal d'interaction.

L'analyse du modèle permet d'obtenir le résultat suivant :

Si la distance maximum entre deux agents est bornée au cours du temps et que l'énergie totale initiale du système ($\sum_{j\in N} \|x_i(0)\|^2 + \sum_{i,j\in N, i\neq j} V\|x_i(t) - x_j(t)\|$) est suffisamment faible alors la trajectoire du système converge asymptotiquement vers un état de flocking : séparation, cohésion et alignement en vitesse.

2.3.2 Modèle de Cucker-Smale

Le modèle de flocking proposé par Cucker-Smale [CuSm07] est un système de commande de consensus pour des agents en interaction, dans lequel chacun met à jour, avec les autres particules, sa vitesse en y ajoutant la moyenne pondérée des différentes vitesses avec des poids de communication dépendants de la distance. Il a été démontré qu'en utilisant le modèle de C-S, les vitesses des particules convergent vers une valeur commune malgré l'absence d'une commande centrale. La commande est implémentée sur un système de type double intégrateur tel que :

$$\begin{aligned} \dot{x}_i(t) &= v_i(t) \\ \dot{v}_i(t) &= \sum_{j\in N} a_{ij}(t)(v_j(t) - v_i(t)) \end{aligned} \tag{2.36}$$

avec comme poids d'interaction :

$$a_{ij}(t) = \frac{H}{(1 + ||x_i(t) - x_j(t)||^2)^\beta} \tag{2.37}$$

Où : $H > 0$ et $\beta \geq 0$ deux paramètres du modèle. La figure 2.2 montre l'évolution des poids d'interaction $a_{ij(t)}$ en fonction des distances $\|x_i(t) - x_j(t)\|$ et des valeurs des paramètres H et β.

Pour estimer les performances du système en alignement en vitesse, on définit :

$$\Lambda(t) = \frac{1}{2} \sum_{i,j\in N} ||v_i(t) - v_j(t)||^2 \tag{2.38}$$

$$\Gamma(t) = \frac{1}{2} \sum_{i,j\in N} ||x_i(t) - x_j(t)||^2 \tag{2.39}$$

Avec : $\Lambda(0) = \Lambda_0$, $\Gamma(0) = \Gamma_0$, et $T = (\frac{1}{2\beta})^{1/2\beta-1} - (\frac{1}{2\beta})^{2\beta/2\beta-1}$

Résultats importants : Les auteurs dans [CuSm07] démontrent que la trajectoire est unique sur R^+ et vérifie la propriété de convergence vers l'alignement en vitesse telle que :

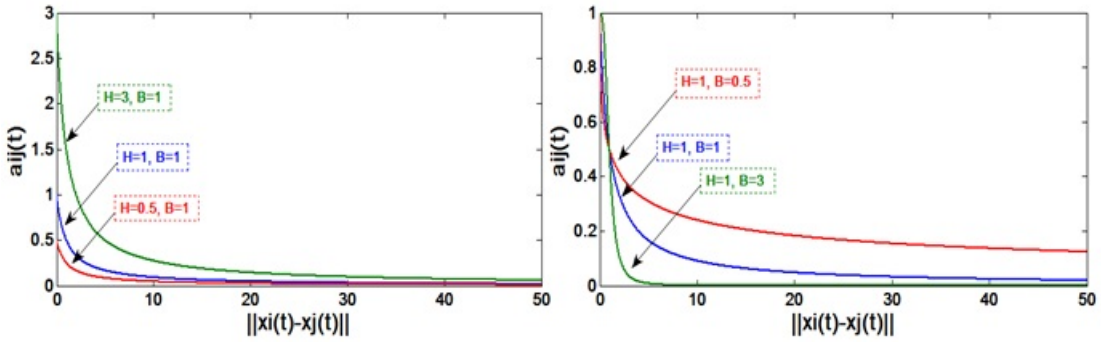

FIGURE 2.2 – Evolution des poids d'interaction $a_{ij}(t)$en fonction des paramètres du modèle

$\exists k_0$, une constante garantissant que :

$$\begin{aligned} \Lambda(t) &\leq \Lambda(0)e^{-\frac{2nH}{k_0\beta}t} \\ \Gamma(t) &\leq k_0 \end{aligned} \tag{2.40}$$

Si une des conditions suivantes est satisfaite :

$$\begin{aligned} &i)\beta < 1/2 \\ &ii)\beta = 1/2 \text{ and } \Lambda_0 < (nH)^2/2 \\ &iii)\beta > 1/2 \text{ and } ((nH)^2/2\Lambda_0)^{1/2\beta-1}T > 2(1+\Gamma_0) \end{aligned} \tag{2.41}$$

Le résultat de Cucker et Smale permet alors de démontrer la convergence du système vers un consensus en vitesse dépendant uniquement de la valeur de β qui détermine le taux et la portée de la communication entre les agents du groupe. Une preuve de convergence simplifiée est proposée dans [HaLi09] en utilisant une fonction de Lyapunov.

2.3.3 Extensions du modèle de Cucker-Smale

De nombreuses études ont suggéré des extensions du modèle de Cucker-Smale pour la commande d'une flotte de véhicules autonomes. Dans [CuDo10], une force répulsive a été utilisée afin de maintenir une distance minimale entre les agents. Des preuves rigoureuses ont été données pour garantir l'anti-collision entre les agents. [Ahn12] et [Park10] proposent une extension du modèle de Cucker-Smale en introduisant des termes d'interaction supplémentaires entre agents afin d'envisager l'évitement de collision et en même temps obtenir des configurations spatiales plus strictes. Un autre travail relatif sur ce sujet peut être trouvé dans [SaMu04], dans lequel un terme de commande découplée avec une fonction potentielle est dérivé pour garantir l'aspect séparation et cohésion entre les agents. En utilisant le terme de commande de vitesse-consensus, ainsi que ce terme de commande découplée, il a été montré que les deux tâches : cohésion de la formation et

évitement de collisions sont toutes deux atteintes.

Parmi les références présentées, [CuDo10] est le seul modèle qui fournit des résultats mathématiques solides sur la convergence vers un consensus en vitesse, et assure simultanément un évitement de collisions entre les agents. Le modèle proposé est présenté comme suit :

$$\begin{aligned} \dot{x}_i(t) &= v_i(t) \\ \dot{v}_i(t) &= \sum_{j=1}^{n} a_{ij}(t)(v_j(t) - v_i(t)) \\ &\quad + \lambda(t) \sum_{j \neq i} f(||x_i(t) - x_j(t)||^2)(x_i(t) - x_j(t)) \end{aligned} \tag{2.42}$$

Avec :

$$\lambda(t) = \left(\frac{1}{n} \sum_{i>j} ||v_i(t) - v_j(t)||^2 \right)^{1/2} \tag{2.43}$$

$$f(r) = (r - r_0)^{-\theta} \text{ and } \theta > 1 \tag{2.44}$$

- Notons que la convergence vers un consensus en vitesse est équivalent à $\lambda(t) = 0$ avec $\lambda(t)$ une fonction qui permet de modérer la force de répulsion
- La distance $r_0 > 0$ représente la distance de sécurité et la fonction différentiable $f(r)$ satisfaisant aux conditions suivantes :

$$\begin{aligned} &1) \int_{r_0}^{r_0+1} f(r)\,\mathrm{d}r = \infty \\ &2) \int_{r_0+1}^{\infty} f(r)\,\mathrm{d}r < \infty \end{aligned} \tag{2.45}$$

La première condition assure la convergence vers le consensus en vitesse et la seconde l'évitement de collision, telle que :

1. toutes les différences de vitesse entre paire d'agent convergent asymptotiquement vers zéro

$$\forall i, j \in N : \lim_{t \to \infty} (v_i(t) - v_j(t)) = 0 \tag{2.46}$$

2. l'évitement de collision entre les agents est toujours assuré

$$\forall i, j \in N : d_{ij}(t) > r_0 \text{ avec } d_{ij}(t) = ||x_i(t) - x_j(t)|| \tag{2.47}$$

Les auteurs montrent, pour ce modèle modifié, que le consensus en vitesse peut être atteint tout en assurant l'évitement de collisions entre les agents si l'une des hypothèses suivantes est assurée :

$$\beta \leq 1/2 \tag{2.48}$$
$$\beta > 1/2 \text{ et } \frac{nH}{4\beta - 2}\left(\frac{1}{1+2\Gamma_0^2}\right)^{\beta-1/2} > \Lambda_0 + \frac{1}{2}\sum_{i>j}\int_{||x_i(0)-x_j(0)||^2}^{\infty} f(r)\,\mathrm{d}r$$

Et cela, sous condition de respecter une distance minimale entre les agents :

$$||x_i(t) - x_j(t)||^2 > r_0 \tag{2.49}$$

2.4 Modèle de contrôle de formation proposé

La question du contrôle en formation est largement discutée dans la littérature et des preuves mathématiques solides sont établies. Différentes approches sont mises en œuvre pour assurer la convergence d'une flotte de véhicule autonome vers une configuration désirée. Certains auteurs introduisent des forces de liaison entre les agents de sorte que la flotte forme une configuration spatiale équilibrée telle que :

$$\forall i, j \in N : \lim_{t\to\infty} d_{ij}(t) = R_d \tag{2.50}$$

Ou bien en utilisant des termes de commande découplés basés sur une fonction potentielle qui atteint le même résultat. Il convient de noter que ces approches peuvent devenir très dangereuses dans un environnement où les mesures sont moins précises ou en cas de présence de défauts.

Notre approche consiste à définir, en se basant sur les résultats obtenus dans [CuDo10], une extension du modèle de Cucker-Smale qui garantit les critères définis dans 2.46 et 2.47. Notre objectif est d'introduire une commande en formation sur les positions des agents adaptée pour la commande de la flotte dans un milieu extérieur. Le but est de faire converger l'ensemble des agents à l'intérieur d'une surface (ici un cercle dans le cas 2D) sans prendre en considération les distances entre les agents, figure 2.3. Cette contrainte peut être formulée par l'équation suivante :

$$\forall i, j \in N : \lim_{t\to\infty} d_{ij}(t) < R(n, r_0) \text{ avec } R(n, r_0) > r_0 \tag{2.51}$$

avec : $R(n, r_0)$ le rayon du cercle de la surface de formation, dépendant de r_0 et de n le nombre d'agents dans la flotte.

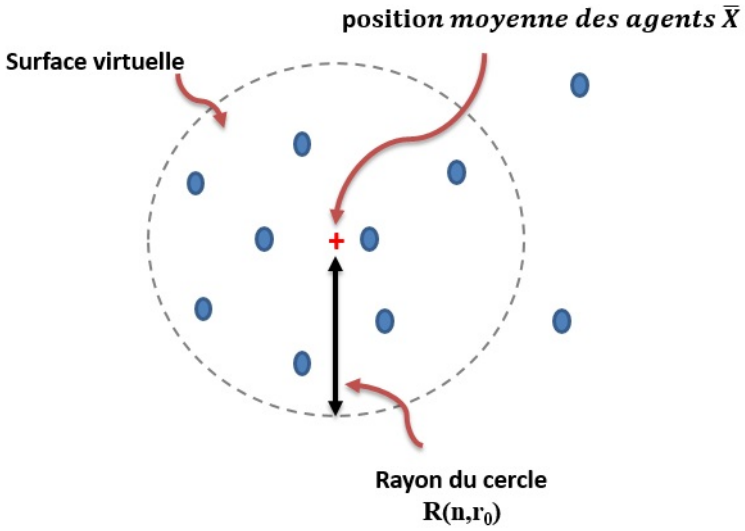

FIGURE 2.3 – Approche de commande en formation proposée

Pour assurer la cohésion de la flotte sans changer la dynamique globale du modèle 2.42, nous proposons d'ajouter une force d'attraction bornée sur les agents se trouvant à l'extérieur de la sphère.

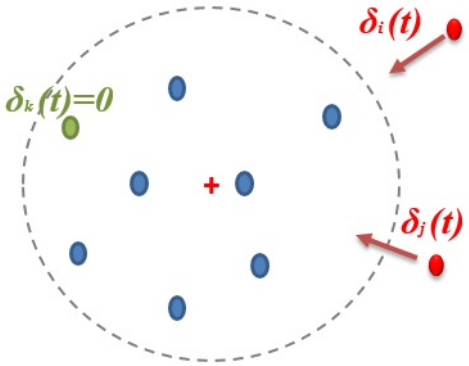

FIGURE 2.4 – Force d'attraction pour la commande de formation

La nouvelle équation de contrôle peut être réécrite comme suit :

$$\begin{aligned} \dot{x}_i(t) &= v_i(t) \\ \dot{v}_i(t) &= U_s(t) + \delta_i(d_i^*(t)) \end{aligned} \tag{2.52}$$

avec :

$$U_s(t) = \sum_{j=1}^{n} a_{ij}(t)(v_j(t) - v_i(t)) + \lambda(t) \sum_{j \neq i} f(||q_i(t) - q_j(t)||^2)(x_i(t) - x_j(t)) \tag{2.53}$$

Où $d_i^*(t)$ représente la distance entre l'agent i et la position du centre du groupe $\overline{X}(t)$ définie par le point (x^*, y^*) ($x^* = \frac{1}{n}\sum_{i=1}^{n} x_i$ et $y^* = \frac{1}{n}\sum_{i=1}^{n} y_i$) et une fonction $\delta_i(d_i^*(t))$ satisfaisant les conditions suivantes :

$$\begin{aligned} &a) ||\delta_i(d_i^*(t))|| \approx 0 \text{ quand } (d_i^*(t) - R(n, r_0)) \leq 0 \\ &b) ||\delta_i(d_i^*(t))|| \leq \delta_{max} \text{ quand } (d_i^*(t) - R(n, r_0)) > 0 \end{aligned} \tag{2.54}$$

Notant que la force d'attraction s'exerce uniquement sur un agent lorsqu'il se trouve en dehors du groupe. La fonction $\delta_i(d_i^*(t))$ peut être définie comme suit :

$$\delta_i(d_i^*(t)) = \overline{\delta_i}(d_i^*(t)) sat\left(\frac{x^*(t) - x_i(t)}{d_i^*(t)}\right) \tag{2.55}$$

avec :

$$\overline{\delta_i}(d_i^*(t)) = \frac{H_c}{2}\left(sign(d_i^*(t) - R(n, r_0)) + 1\right) \tag{2.56}$$

Où H_c est défini comme un paramètre constant positif du modèle, figure 2.5.

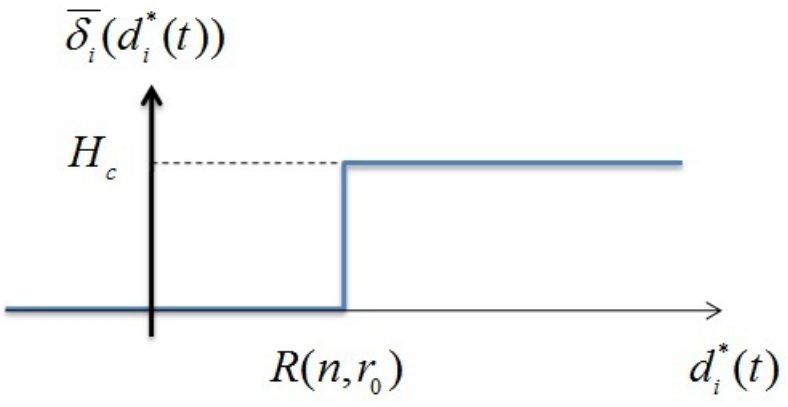

FIGURE 2.5 – Force d'attraction $\delta_i(d_i^*(t))$ considérée

La fonction $\delta_i(d_i^*(t))$ est considérée comme une perturbation active et saturée. Le choix de la variable H_c et des paramètres de saturation doivent donc être suffisamment petits pour ne pas détériorer le signal de commande.

2.5 Résultats de simulation

2.5.1 Implémentation de la commande sur un modèle en double intégrateur

Afin de valider les résultats obtenus, le contrôleur développé dans la section précédente à été implémenté en simulation sur Matlab sur douze agents modélisés par des dynamiques en double intégrateur. Les paramètres de commande considérés sont illustrés dans le tableau 2.5.1. Les valeurs de H et β sont choisies selon les équations 2.48 afin de garantir le consensus en vitesse et l'anti-collision entre les agents. Le coefficient H_c est déterminé expérimentalement telle que sa valeur est suffisamment petite pour ne pas affecter la convergence vers le consensus.

H	β	H_c	r_0
1	0.4	0.1	1

TABLE 2.1 – Paramètres de simulation

Nous considérons dans cette première simulation que les véhicules autonomes ont initialement des vitesses différentes, et sont initialisés à des positions aléatoires en garantissant une distance minimale entre chacun des agents supérieure à la distance de sécurité r_0.

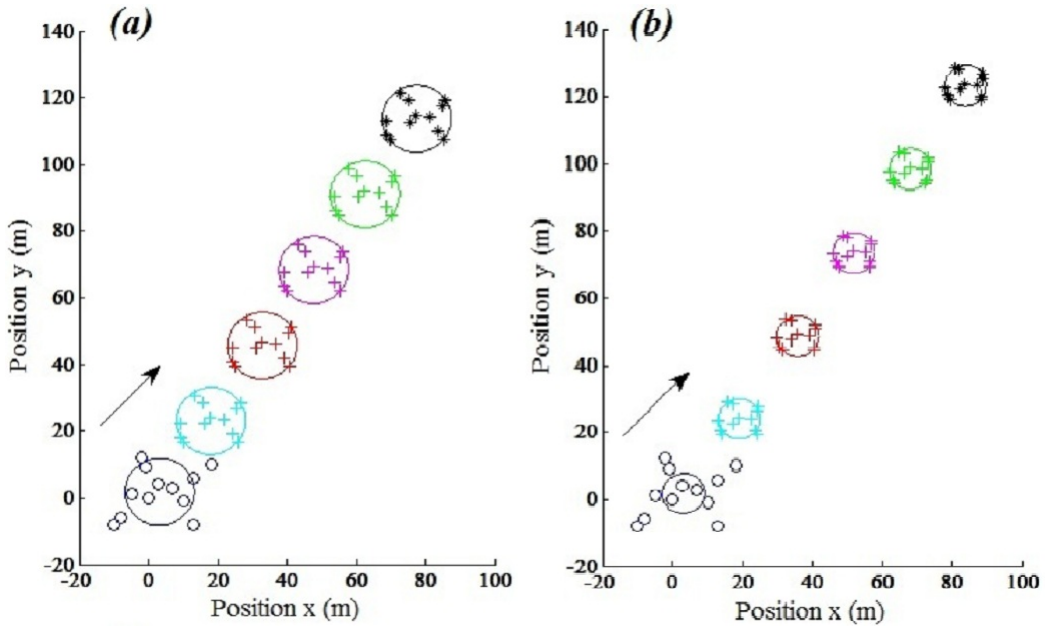

FIGURE 2.6 – Le mouvement de flocking pour douze agents autonomes

La figure 2.6 montre deux cas de mise en œuvre du modèle de flocking en considérant deux valeurs différentes du rayon du groupe avec $R1(n, r_0) = 10$ et $R2(n, r_0) = 6$. Notons

que les agents sont capables de converger dans les deux cas à un voisinage délimité par le cercle de rayon $R(n, r0)$.

-

Pour évaluer la convergence de la flotte, la figure2.7 et la figure2.8 présentent les vitesses d'évolution $v_x(t)$ et $v_y(t)$ de tous les agents pour les deux précédents cas.

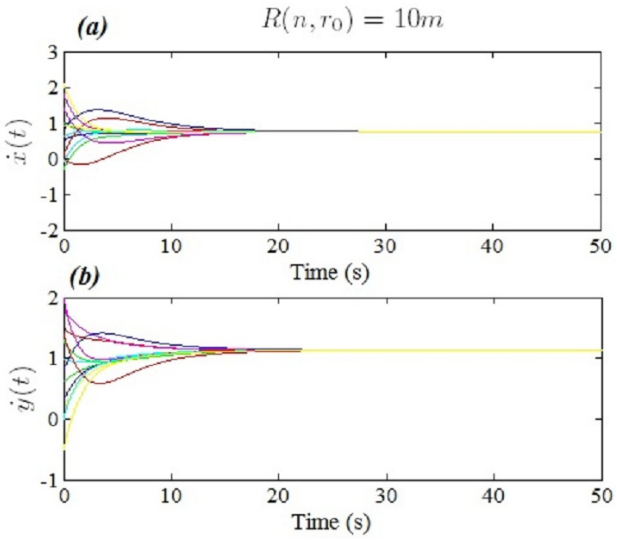

FIGURE 2.7 – Les vitesses ($composante - x$, $composante - y$)

Le temps de convergence est estimé à $15s$ lorsque $R_1(n, r_0) = 10$ et à $18s$ quand $R_1(n, r_0) = 6$. Ce résultat est en lien avec l'influence du terme $\delta_i(d_i^*(t))$ qui augmente lorsque $R(n, r_0)$ diminue.

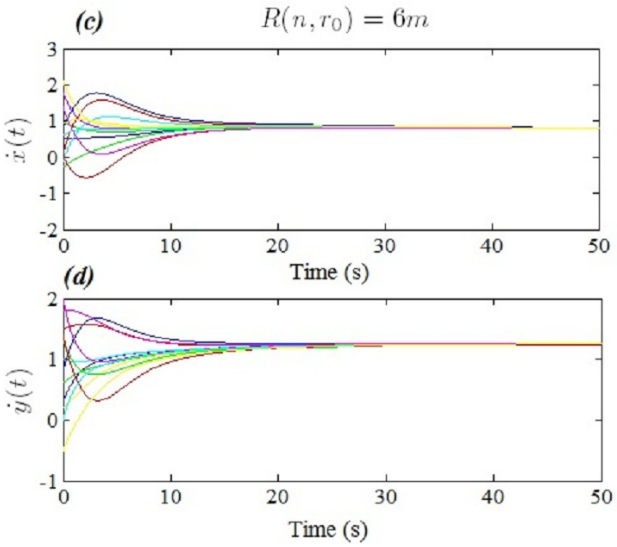

FIGURE 2.8 – Les vitesses ($composante - x$, $composante - y$)

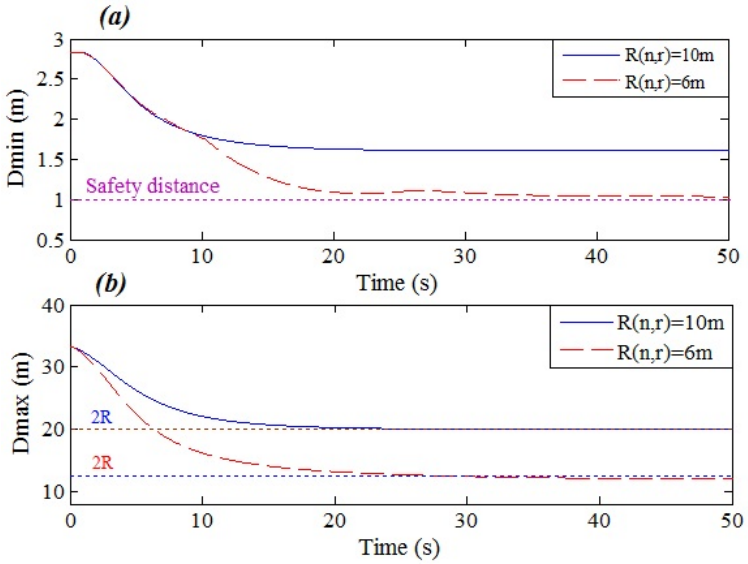

FIGURE 2.9 – Limites des inter-distances entre les agents de la flotte

la figure2.9 permet de visualiser les distances minimales et maximales entre chaque deux agents, tel que :

$$D_{min} = \min_{i \neq j} ||x_i(t) - x_j(t)|| \tag{2.57}$$

$$D_{max} = \max_{i \neq j} ||x_i(t) - x_j(t)|| \tag{2.58}$$

Notons que les distances de sécurité entre les agents sont toujours respectées.

Afin de voir l'influence de la force de rappel sur le comportement de la flotte, nous proposons de simuler l'algorithme proposé dans 2.42 sans le terme de cohésion proposé.

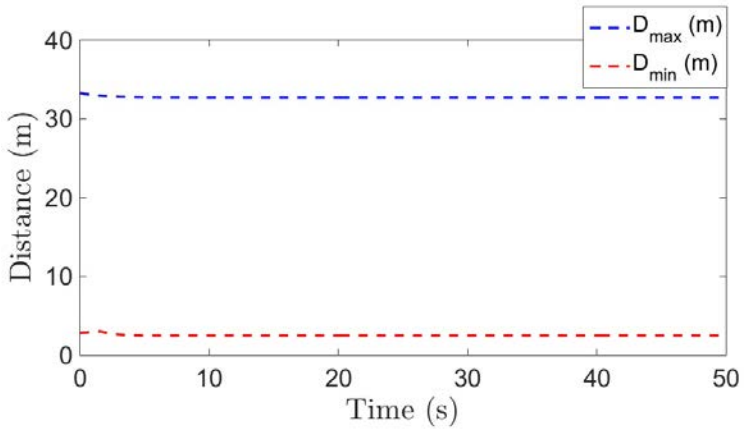

FIGURE 2.10 – Limites des inter-distances entre les agents de la flotte. Cas sans ajout de la force de rappel

Les figures 2.10 et 2.11 montrent respectivement les limites des inter-distances entre les agents et l'évolution des vitesses au cours du temps.

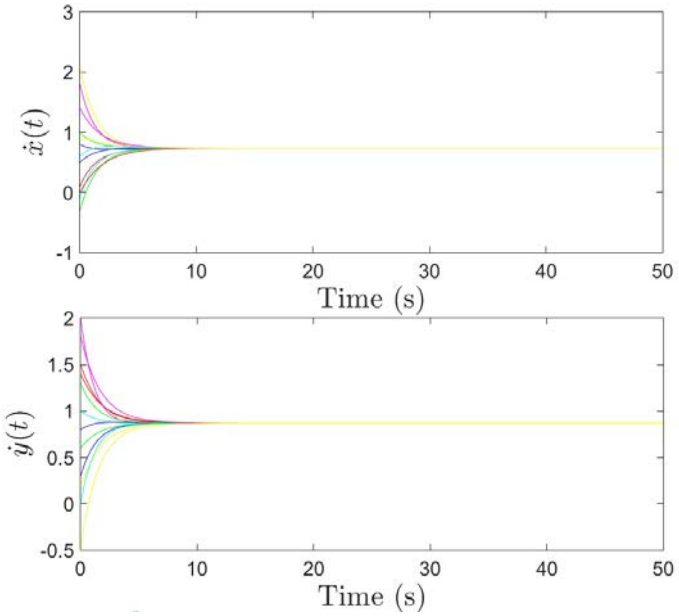

FIGURE 2.11 – Les vitesses ($composante - x$, $composante - y$). Cas sans ajout de la force de rappel

Nous remarquons que le temps de convergence est plus important dans le cas de l'ajout de la force de rappel dans la loi de commande. Néanmoins, même si l'anti-collision est assurée, les distances entre les agents restent très importantes. L'ajout de la force de rappel permet donc la commande en formation de la flotte avec des inter-distances bornées.

Les résultats obtenus dans cette première simulation sont tout à fait conformes aux objectifs définis dans la section précédente. Les performances de la commande de consensus et anti-collisions ne sont pas détériorées par l'ajout de la force de rappel active. Cette stratégie permet également de commander la formation de la flotte sans considérer l'ensemble des inter-distances entre chaque paire d'agents. La surface de formation peut aussi être modifiée grâce au paramètre $R(n, r_0)$.

2.5.2 Implémentation de la commande sur un modèle de Quadrirotor

Les drones aériens ou UAVs (Unmanned Aerial Vehicles) sont des engins volants sans pilote capables de mener à bien une mission de façon plus ou moins autonome. Il en existe de toutes les tailles et de toutes les formes. La fonction principale de ces véhicules aériens est d'accomplir des travaux à risques ou dans des environnements hostiles. Les premières applications ont d'ailleurs été mises en œuvre par les militaires pour des missions de surveillance ou de reconnaissance, sans risque de pertes humaines. Plus récemment, des

applications civiles ont fait leur apparition comme la prévention des feux de forêts, l'inspection des grands ouvrages, la surveillance du trafic autoroutier ou la collecte de données météorologiques.

Afin d'illustrer expérimentalement les approches développées dans ce travail de thèse, nous avons choisi d'utiliser une plateforme de drones composée d'une flotte de quadrirotors.

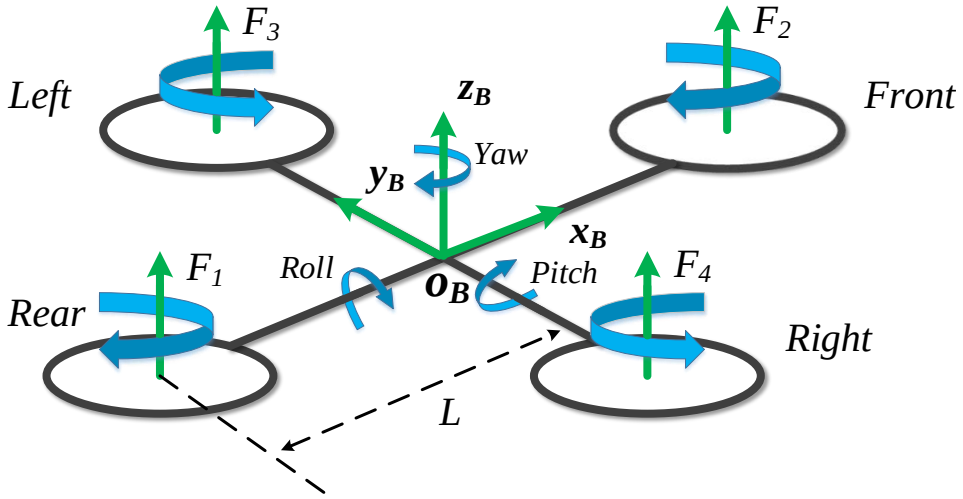

FIGURE 2.12 – Shéma de Quadrirotor

Le quadrirotor est un drone à voilure tournante possédant quatre moteurs disposés en forme de croix de cotés égaux. Sa structure lui offre l'avantage de pouvoir décoller et atterrir verticalement, et aussi d'effectuer des vols stationnaires. Les moments de roulis et tangage sont obtenus en jouant sur les vitesses de rotation de deux moteurs opposés, de plus, deux des quatre rotors ont une hélice à pas inverse ce qui permet d'annuler le couple de lacet.

Les différents couples et forces dus aux moteurs pris en compte dans cette étude sont représentés sur la figure 2.12. Ainsi, chacun des quatre moteurs M_i produit une force F_i et un couple τ_i sur l'axe z. La poussée totale est donc la somme des forces des quatre actionneurs $u = F_1 + F_2 + F_3 + F_4$. Le couple τ_x autour de l'axe x est obtenu par la différence de forces $F_3 - F_4$ et le couple τ_y autour de l'axe y par la différence de forces $F_1 - F_2$. Enfin le couple τ_z autour de l'axe z est obtenu par la somme des couples produits par les moteurs $\tau_1 + \tau_2 - \tau_3 - \tau_4$, M_1 et M_2 tournant en effet dans le sens positif alors que M_3 et M_4 tournent en sens inverse.

Les quadrirotors ont reçu ces dernières années une grande attention des chercheurs dans les domaines de la théorie de la commande et la robotique. En tant que plate-forme

peu coûteuse et facile à assembler, le quadrirotor représente une plate-forme idéale pour valider les approches proposées.

Le modèle dynamique du Quadrirotor peut être calculé à l'aide du formalisme d'Euler-Lagrange [AbZh13][LaYa06] et peut être présenté dans l'espace d'état sous la forme suivante :

$$\begin{aligned}
\ddot{x} &= \frac{(sin\psi sin\phi + cos\psi sin\theta cos\phi)u_z(t) - K_1\dot{x}}{m} \\
\ddot{y} &= \frac{(sin\psi sin\theta cos\phi - cos\psi sin\phi)u_z(t) - K_2\dot{y}}{m} \\
\ddot{z} &= \frac{(cos\theta cos\phi)u_z(t) - K_3\dot{z}}{m} - g \\
\ddot{\phi} &= \frac{u_\phi(t) - K_4\dot{\phi}}{I_x} \\
\ddot{\theta} &= \frac{u_\theta(t) - K_5\dot{\theta}}{I_y} \\
\ddot{\psi} &= \frac{u_\psi(t) - K_6\dot{\psi}}{I_z}.
\end{aligned} \tag{2.59}$$

où $K_i, i = 1, 2, ..., 6$ sont les coefficients de traînée associés à la force de traînée aérodynamique et L est la distance entre le centre de gravité du quadrotor et le centre de chaque hélice. Notez que les coefficients de traînée sont négligeables à basse vitesse. De même, I_x, I_y et I_z représentent les moments d'inertie selon les directions x, y et z.

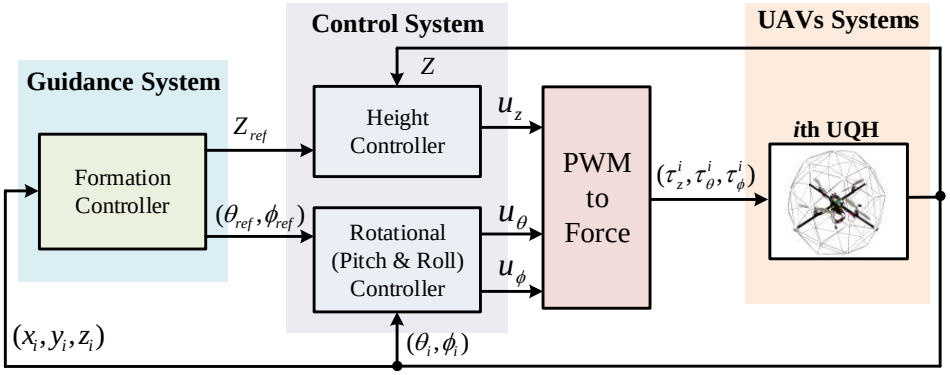

FIGURE 2.13 – Architecture de commande proposée pour chaque agent

De plus, la relation suivante entre les accélérations et les couples peut être formulée comme :

$$\begin{bmatrix} u_z(t) \\ u_\theta(t) \\ u_\phi(t) \\ u_\psi(t) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ L & -L & 0 & 0 \\ 0 & 0 & L & -L \\ C_m & C_m & -C_m & -C_m \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \\ u_4(t) \end{bmatrix}. \tag{2.60}$$

La force de l'hélice et son signal de modulation (PWM) ont la relation suivante :

$$u_i(t) = K_m \frac{\omega_m}{s + \omega_m} u_{ci}(t). \tag{2.61}$$

Pour faciliter la conception du système de contrôle, nous allons utiliser les travaux de recherche existants [LiYu15][ZhCh13]. La simplification du modèle suivant peut être obtenue lorsque $K_m \frac{\omega_m}{s+\omega_m} \approx K_m$. Donc, Eq. (2.61)) peut être réduite comme suit :

$$u_i(t) = K_m u_{ci}(t), \tag{2.62}$$

Où K_m et ω_m sont théoriquement supposés identiques pour tous les moteurs.

Linéarisation du modèle du Quadrirotor

Comme le modèle du quadrirotor est fortement non linéaire, les mouvements de translation et de rotation sont couplés, Afin de faire correspondre la dynamique du quadrirotor avec un modèle double intégrateur pour la conception de l'algorithme de flocking, *L'hypothèse 1* est ainsi conçue pour linéariser sa dynamique.

Hypothèse 1 : Le quadrirotor est supposé être dans un vol quasi-stationnaire ce qui implique :

- $u_z \approx mg$ points vers la direction verticale ;
- les angles de tangage et de roulis sont petits, de sorte que $sin\phi \approx \phi$ and $sin\theta \approx \theta$;
- angle de lacet ($\psi = 0$).

En se basant sur *l'hypothèse 1* et l'équation(2.59), la nouvelle dynamique de translation et de rotation du quadrirotor peut être obtenue sous la forme d'équations (2.63) et (2.64), respectivement :

Dynamique de Translation :

$$\begin{aligned} \dot{x} &= v_x \\ \dot{y} &= v_y \\ \dot{z} &= v_z \\ \dot{v}_x &= g\theta \\ \dot{v}_y &= -g\phi \\ \dot{v}_z &= u_z/m - g. \end{aligned} \tag{2.63}$$

Dynamique de Rotation :

$$\begin{aligned} \dot{\phi} &= \omega_\phi \\ \dot{\theta} &= \omega_\theta \\ \dot{\psi} &= \omega_\psi \\ \dot{\omega}_\phi &= u_\phi(t)/I_x \\ \dot{\omega}_\theta &= u_\theta(t)/I_y \\ \dot{\omega}_\psi &= u_\psi(t)/I_z. \end{aligned} \tag{2.64}$$

Puisque l'algorithme de commande de flocking proposé est conçu sur la base du modèle linéarisé (2.63) du quadrirotor, alors que la simulation/expérience se déroule sur le modèle non linéaire (2.59). Il faut donc inévitablement tenir compte des incertitudes liées à la linéarisation sans provoquer d'oscillations inattendues. Dans cette étude, des gains supplémentaires sont ajoutés aux trois termes de l'équation(2.52) afin de compenser les incertitudes du modèle non-linéaire. Après cela, la loi de contrôle de flocking modifiée devient :

$$\begin{aligned}\dot{p}_i(t) =& q_i(t)\\ \dot{q}_i(t) =& \sum_{j=1}^{n} K_d a_{ij}(t)\Big(q_j(t) - q_i(t)\Big) + K_p \lambda(t) \sum_{j\neq i} f(\big|\big|p_i(t)\\ & - p_j(t)\big|\big|^2)\Big(p_i(t) - p_j(t)\Big) + \delta_i(d_i^*(t)),\end{aligned} \tag{2.65}$$

Où $K_p > 0$ et $K_d > 0$ sont des gains définis par l'utilisateur.

La contrôle (LQR) "linear quadratic regulator", bien connu et largement appliqué pour la commande des systèmes automatiques, peut être une solution appropriée pour la conception du contrôleur interne de chaque quadrirotor [AsWi13]. Par conséquent, dans cette étude, le schéma de contrôle LQR est adopté pour développer la stratégie de commande.

Sans perte de généralité, le modèle linéaire du quadrirotor (2.64) en considérant simplement le mouvement du pitch and roll, peut être réécrit dans la représentation de l'espace d'état suivante :

$$\dot{x}(t) = Ax(t) + Bu(t) + G\omega(t), \tag{2.66}$$

Où $x(t) \in^n$ est le vecteur d'état, $u(t) \in \Re^m$ représente le vecteur d'entrée, $A \in \Re^{n\times n}$, et $B \in \Re^{n\times m}$. $\omega(t) = [g, \omega_d(t)]^T$ comprend l'accélération de la gravité g et les perturbations externes bornées. $\omega_d(t) \in \Re^r$. Dans cette étude, $u(t) = \begin{bmatrix} u_z & u_\theta & u_\phi \end{bmatrix}^T$, $x(t) = \begin{bmatrix} z & \dot{z} & \theta & \dot{\theta} & \phi & \dot{\phi} \end{bmatrix}^T$,

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 & 0 & 0\\ 1/m & 0 & 0\\ 0 & 0 & 0\\ 0 & 1/I_x & 0\\ 0 & 0 & 0\\ 0 & 0 & 1/I_y \end{bmatrix},$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \text{ et } G = \begin{bmatrix} 1 & 0\\ 1 & 0\\ 1 & 0\\ 1 & 0\\ 1 & -1\\ 1 & 0 \end{bmatrix}^T.$$

En tant que mécanisme efficace pour éliminer l'erreur d'état stationnaire, le terme intégral est encore introduit dans la conception du schéma de commande [ZhJi01]. Après avoir incorporé ce terme intégral, le système (2.66) est réécrit comme suit :

$$\dot{x}_a(t) = A_a x_a(t) + B_a u(t) + G_a w_a(t), \tag{2.67}$$

où $x_a(t) = \left[(\int_0^t \epsilon(t)dt)^T, \quad x^T(t)\right]^T$ est le vecteur d'état augmenté, $\epsilon(t) = y_{ref}(t) - y(t)$ est l'erreur entre la référence et la sortie $y(t) = Cx(t)$, $w_a(t) = [\omega^T(t), y_{ref}^T(t)]^T$ contient $\omega(t)$ et la référence $y_{ref}(t)$.

$$A_a = \begin{bmatrix} 0 & -SrC \\ 0 & A \end{bmatrix} \in \Re^{(l+n)\times(l+n)},$$
$$B_a = \begin{bmatrix} 0 \\ B \end{bmatrix} \in \Re^{(l+n)\times m},$$
$$G_a = \begin{bmatrix} 0 & I \\ G & 0 \end{bmatrix} \in \Re^{(l+n)\times(l+r)},$$

$S_r \in^{l\times p}$ est utilisé pour sélectionner les états du système requis.

Ensuite, l'utilisation du contrôleur LQR consiste à concevoir une entrée de commande appropriée $u(t)$ pour la commande du système augmenté pour toutes les valeurs initiales $x_a(t_0)$. Cela peut être réalisé en minimisant la fonction objective suivante [KeKa09] :

$$J = \int_{t_0}^{\infty} (x_a(t)^T Q x_a(t) + u(t)^T R u(t))dt, \tag{2.68}$$

Où $Q \in \Re^{(n+l)\times(n+l)}$ est une matrice symétrique, et $R \in \Re^{(m+l)\times(m+l)}$ est une matrice définie positive.

Le gain de commande K est alors obtenu en résolvant des équations algébriques de Riccati. La commande optimale pour le retour d'état augmenté est définie par :

$$u(t) = -K x_a(t). \tag{2.69}$$

2.6 Simulation

Afin d'illustrer l'efficacité de la méthode de contrôle de flocking proposée, des simulations numériques sont réalisées sur un groupe de 12 quadrirotors modélisés par des modèles non-linéaires.

Initialement, les positions et vitesses des UAVs dans la flotte sont initialisées aléatoirement en respectant les conditions définies précédemment.La distance de sécurité entre les UAVs est fixée à $1m$. Les valeurs des paramètres de commande de la flotte sont sélectionnées comme suit :

$H = 1$, $\beta = 0.4$, $H_c = 0.1$, $r_0 = 1m$, $K_p = 0.5$, $K_d = 0.7$.

Les gains pour le contrôleur de formation sont calculés comme suit :

$$K_e = [99.9995, 0, 0, -60.6848, -13.4133, 0, 0, 0, 0;$$
$$0, 9.9998, 0, 0, 0, -6.6054, -1.1816, 0, 0;$$
$$0, 0, 9.9998, 0, 0, 0, 0, -6.6054, -1.1816]$$

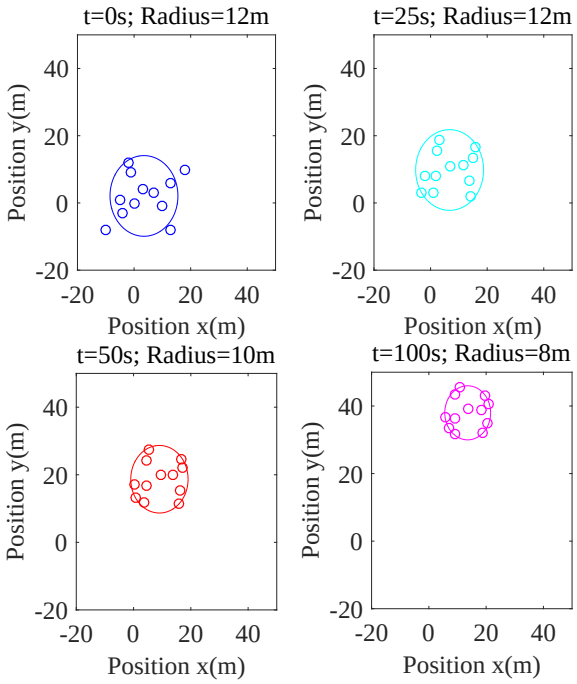

FIGURE 2.14 – Le déplacement en formation de la flotte

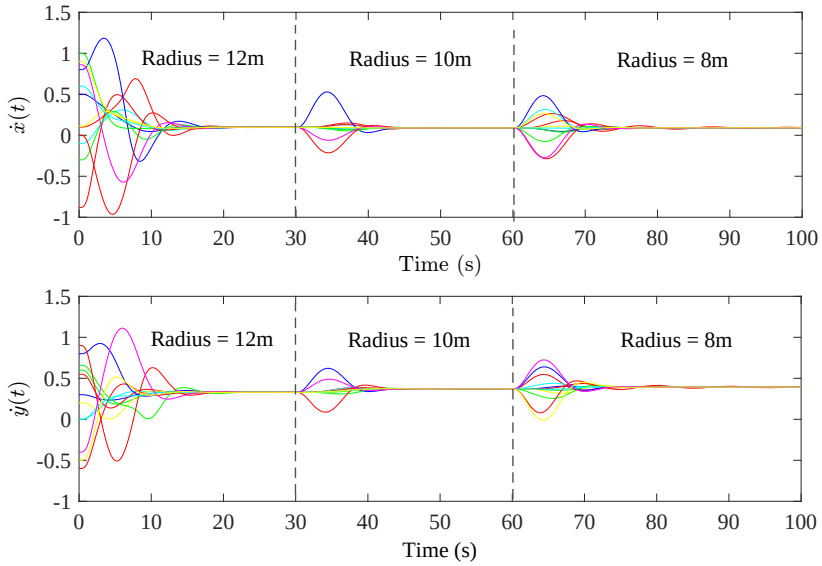

FIGURE 2.15 – Évolution des vitesses des drones au cours du temps

Comme illustré dans la figure 2.14, Le rayon de la flotte est d'abord choisi égal à $R(n, r_0) = 12m$, puis modifié à la valeur $R(n, r_0) = 10m$ et $R(n, r_0) = 8m$ aux instants $30s$ et $60s$ respectivement.

Les résultats de simulation démontrent que les UAVs convergent vers un déplacement en formation en respectant un positionnement à l'intérieur d'une surface de cercle défini par un rayon $R(n, r_0)$.

La figure 2.15 illustre l'évolution des vitesses de tous les quadrirotors au cours du temps. On peut observer que tous les véhicules convergent à la même vitesse au cours d'environ $20s$.

Pour étudier de manière plus claire la performance de la flotte d'UAVs, la figure 2.16 montre les distances minimales et maximales entre chaque deux agents, ces distances sont calculées par les équations :

$$\begin{aligned} D_{min} &= \min_{i \neq j} ||x_i(t) - x_j(t)|| \\ D_{max} &= \max_{i \neq j} ||x_i(t) - x_j(t)||. \end{aligned} \tag{2.70}$$

A partir de la Fig. 2.16, la distance de sécurité $(1m)$ entre tous les agents ainsi que la distance maximale tolérée sont toujours garanties.

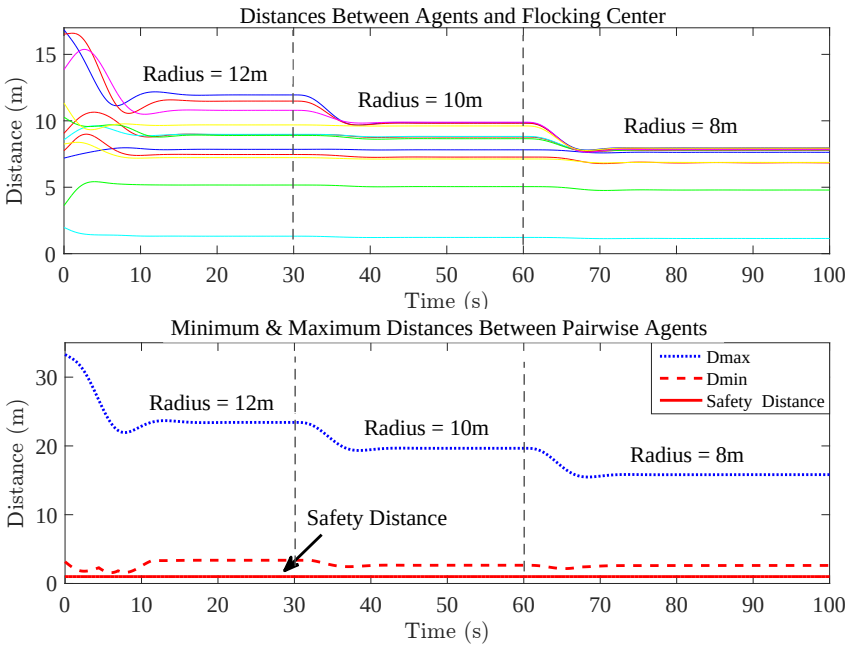

FIGURE 2.16 – Les distances entre UAVs dans la flotte

La figure 2.17 montre la distance moyenne $\Phi(t)$ et les différences de vitesse moyenne $\Psi(t)$ entre les UAVs.

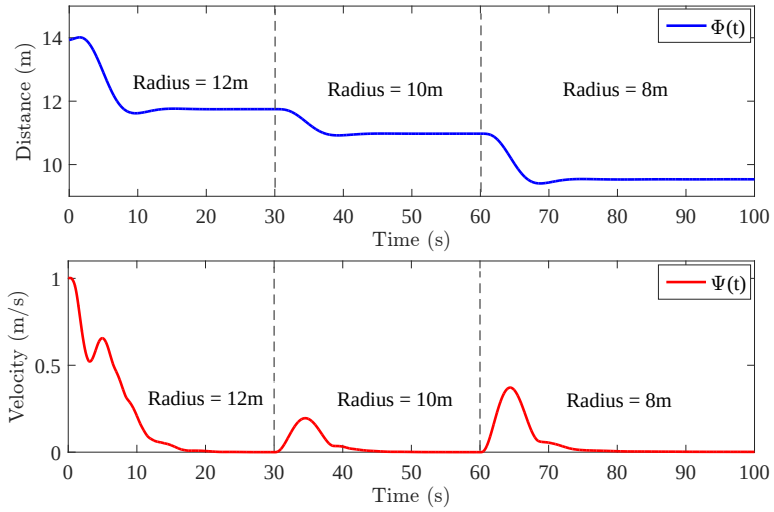

FIGURE 2.17 – La distance moyenne et les différences de vitesse entre les UAVs

$$\begin{aligned}\Phi(t) =& \frac{1}{(n-1)^2}\sum_{i=1}^{n}\sum_{j=1}^{n}||p_i(t)-p_j(t)||,\\ \Psi(t) =& \frac{1}{(n-1)^2}\sum_{i=1}^{n}\sum_{j=1}^{n}||q_i(t)-q_j(t)||.\end{aligned} \tag{2.71}$$

On voit clairement que la commande de formation, et le consensus en vitesse sont garantis.

2.7 Conclusion

Nous avons proposé dans ce deuxième chapitre une extension du modèle de flocking proposé dans [CuDo10][CuSm07]. Notre approche consiste à introduire dans la loi de commande de chaque agent une force de rappel en direction d'un centre virtuel, permettant ainsi de gérer la surface d'évolution de la flotte. Le centre virtuel est calculé de manière indépendante par chaque véhicule. La commande a été implémentée ensuite en simulation sur une flotte de quadrirotors. Des gains sont ajoutés à la loi de commande en raison des dynamiques non linéaires des véhicules.

La commande présentée dans [CuDo10] offre des garanties quant à la convergence vers le consensus en vitesse et à l'évitement des collisions entre les véhicules de la flotte. Il est important de souligner que la force de rappel introduite dans la loi de commande doit être suffisamment petite pour ne pas affecter ou détériorer ces deux critères. Sachant que cette force intervient uniquement quand un agent détecte qu'il est en dehors de la formation. Il sera intéressant de définir par la suite les limites de ces valeurs.

Dans cette première approche, nous avons considéré le cas d'un graphe complet où tous les agents avaient l'information sur les positions et vitesses de toute la flotte. Dans ce

cas le point virtuel est unique et représente le centre du cercle délimitant l'espace désiré d'évolution de la formation. Nous nous intéressons actuellement au cas où le graphe de communication n'est pas complet. Dans ce cas le centre virtuel n'est pas unique mais dépend du voisinage de chaque véhicule. Des indications sont données dans le chapitre IV à ce sujet. Une commande tolérante aux défauts sera également présentée pour gérer d'éventuels défauts\défaillances. Nous nous intéresserons particulièrement aux cas de pertes d'agents, de défauts actionneurs et aux pertes de communication.

Chapitre 3

Commande en formation basée sur une optimisation par essaim de particules distribuées

Sommaire

3.1 Introduction

En plus des différentes architectures de contrôle utilisées pour la commande de flotte de véhicules autonomes, il existe également différents outils théoriques pour synthétiser l'action de commande. Une première classe de méthodes propose d'utiliser un protocole de consensus visant à atteindre un objectif défini de la flotte. Les algorithmes de Rendez-vous par exemple, sont utilisés dans le cas où cet objectif est une position commune à atteindre par les agents. D'autres, comme les algorithmes Flocking visent à faire converger les véhicules vers une direction et une vitesse commune tout en se déplaçant en un groupe compact et en évitant les collisions. Dans la commande de couverture, l'objectif est de concevoir une flotte capable de se déployer sur une surface de la manière la plus optimale possible. Il convient de noter que dans ces méthodes, la théorie des graphes algébriques est l'outil théorique principal utilisé pour résoudre les problèmes de commande.

Une autre classe de méthodes basées sur des approches d'optimisation, et qui nous intéresse particulièrement dans ce chapitre, couvre une large gamme de méthodes pour la synthèse et la conception de commande de flotte. Une commande basée optimisation est un terme général qui se réfère à la conception d'un contrôleur en minimisant ou maximisant un critère d'optimisation. L'optimalité étant généralement équivalente à une certaine propriété souhaitée, par exemple, la stabilité, la réactivité ou la robustesse. Cette définition assez large peut couvrir les commandes optimales classiques [Alekseev13], les techniques basées LMI [MoBa17][HeNa16], ou la commande prédictive [CaAl13]. L'avantage avec ces approches dans la commande des systèmes multi-agents est que l'objectif est formulé sous la forme d'un problème d'optimisation, en fournissant des outils efficaces pour trouver la solution optimale par rapport aux critères considérés.

Toutes ces approches de commande de flotte offrent des caractéristiques intéressantes du point de vue théorique et particulièrement les preuves de stabilité et de convergence. Néanmoins, lors de la conception de ces commandes, un grand nombre d'hypothèses et de simplifications sont considérées, en relation avec les dynamiques des agents, les non-linéarités, les perturbations, les pertes de communications ...etc. L'application de ces approches dans le cas de commande de véhicules autonomes réels nécessite alors souvent la conception d'un contrôleur local qui prend en compte la dynamique interne de chaque agent [YuWa16][LeDi15]. Les résultats théoriques peuvent alors être affectés ou totalement détériorés.

Nous proposons dans ce troisième chapitre de thèse une nouvelle approche de commande en formation distribuée basée sur une optimisation par essaim de particules, connue en anglais sous le nom de "Particle Swarm Optimization algorithm (PSO)". Cette approche offre de nombreux avantages notamment quant à son aspect générique pouvant s'appliquer sur différents types et dynamiques de véhicules, sa simplicité d'implémentation et de réglage, et également son architecture distribuée et son aspect temps réel [DuSw16][Endelbrecht14]. Afin de valider l'approche proposée, des simulations et des tests expérimentaux ont été réalisés sur la commande d'une flotte de quadrirotors. Les résultats obtenus sont présentés dans la suite de ce chapitre.

3.2 Problèmes d'optimisation difficile et Méta-heuristiques

Plusieurs problèmes dans les domaines de l'automatique et de l'ingénierie peuvent être exprimés sous forme d'un problème d'optimisation. Dans ce cas, une ou plusieurs fonctions objectives sont définies. L'objectif est alors de minimiser ou de maximiser ces fonctions par rapport à des contraintes liées au problème traité. L'ensemble des solutions possibles forment l'espace de recherche. Celui-ci peut être aussi limité par les contraintes du problème. Deux types de contraintes peuvent être distingués, des contraintes dites impératives "dites dures" et des contraintes indicatives "dites molles". Les solutions ne respectant pas les contraintes dures sont rejetées et sont considérées en dehors de l'espace de recherche, alors que les contraintes molles doivent être respectées autant que possible [KlGo14].

Les problèmes d'optimisation rencontrés en pratique sont de plus en plus complexes, et les méthodes classiques ne sont plus aussi efficaces pour traiter ces nouvelles difficultés dites d'optimisation difficile [GiMi16]. De nouvelles méthodes appelées Méta-heuristiques tels que les algorithmes évolutionnaires [Vanaret15], les algorithmes de colonies de fourmis [ZhHu13], la méthode du recuit simulé [RaZi16], la méthode de recherche tabou [TaGo14], l'optimisation par essaim de particules (PSO) [RoTa13] sont apparues à partir des années 1980, avec une ambition commune, résoudre au mieux ces nouveaux problèmes d'optimisation difficile.

Il existe deux familles de problème d'optimisation, les problèmes discrets et les problèmes à variables continues. Dans la littérature deux sortes de problèmes reçoivent l'appellation de problème d'optimisation difficile.

- des problèmes d'optimisation discrets, pour lesquels l'algorithme polynomial exact n'est pas connu ;
- des problèmes d'optimisation à variables continues, pour lesquels l'algorithme permettant de retrouver un optimum global à coup sûr et en un nombre fini de calculs n'est pas connu.

Il existe un grand nombre de méta-heuristiques différentes, allant de la simple recherche locale à des algorithmes complexes de recherche globale, qui sont généralement des algorithmes stochastiques itératifs. Les itérations successives doivent permettre de passer d'une solution de mauvaise qualité à la solution optimale. L'algorithme s'arrête après avoir atteint un critère d'arrêt, consistant généralement en l'atteinte d'un temps d'exécution maximum défini ou une précision demandée. Afin de compléter cette présentation succincte des méta-heuristiques, il est possible de qualifier un problème d'optimisation :

- de problème mono-objectif s'il s'agit d'optimiser une seule fonction objective ou de problème multi-objectif lorsqu'on a affaire à plusieurs objectifs. Dans ce deuxième cas le minimum global n'existe pas, mais on cherche plutôt un ensemble de solutions parmi lesquelles on ne peut décider si une solution est meilleure qu'une autre sur tous les objectifs, on parle alors d'optimales au sens de Pareto, avec des méthodes multimodales où l'on ne recherche pas seulement l'optimum global mais l'ensemble des minimums globaux et locaux ;

- d'optimisation dynamique où la fonction objectif n'est plus fixe mais variable dans le temps.

Le choix de la méta-heuristique la plus efficace, capable de fournir un optimum global au bout d'un temps de calcul minimal, est une étape cruciale dans la démarche de résolution d'un problème d'optimisation difficile. Jusqu'à ce jour, il n'existe pas de méthodes théoriques générales capables de démontrer la convergence d'une méta-heuristique appliquée à des problèmes d'optimisation difficiles sauf sous des hypothèses très restrictives. L'utilisateur de ce type d'algorithmes d'optimisation, confronté aux problèmes d'optimisation difficiles, devrait donc faire appel uniquement à son savoir faire et son expérience. En terme d'exemple, les algorithmes évolutionnaires, connus pour leur capacité à retrouver le minimum global au détriment d'un temps de calcul parfois assez important peuvent être cités. Ils sont très bien adaptés pour des problèmes d'optimisation multimodale. Les algorithmes de colonies de fourmis et d'essaim particulaire « PSO » sont quant à eux plus efficaces dans le cas de problèmes d'optimisation dynamique.

La commande de formation peut donc être considérée comme un problème d'optimisation dynamique où la fonction objectif n'est pas fixe mais variable dans le temps en fonction du nombre d'agents dans la flotte, de la configuration spatiale désirée, des contraintes de communication, et de l'objectif à suivre. L'algorithme PSO "Particle Swarm Optimization" ou algorithme d'optimisation par essaim de particules, nous semble alors être le plus adéquat pour ce type problème. La capacité et l'efficacité de cet algorithme dans la recherche du minimum global, ses caractéristiques de convergence ainsi que sa vitesse d'exécution essentielle pour une implémentation temps réel, sont des atouts solides et permettent de consolider et de justifier notre choix pour cet algorithme [RoTa13][Kennedy11].

3.3 Algorithme d'optimisation par essaim de particules PSO

3.3.1 Description et Analyse

La méthode d'optimisation par essaim de particules (PSO) est apparue en 1995 aux Etats Unis [EbKe95]. Ces deux concepteurs sont Russel Eberhart et James Kennedy. Elle est fondée sur la notion de coopération entre des agents dits "particules", qui arrivent à résoudre des problèmes complexes en s'échangeant des informations. Contrairement aux méthodes évolutionnaires basées sur la compétition et la sélection, les particules dans l'algorithme PSO qui ont de mauvaises prestations ne sont pas éliminées au fil du temps mais tendent à améliorer leurs performances en se servant de leur propre expérience et des informations reçues des meilleures particules de l'essaim.

La formulation de l'algorithme PSO peut être décrite de la manière suivante : sur un espace de recherche à n dimensions, un ensemble de m particules appelées «essaim» sont distribuées de manière aléatoire. L'emplacement d'une particule i dans l'espace de recherche sera défini comme étant sa position $\overrightarrow{x}_i(t)$ à l'instant t, et $\overrightarrow{V}_i(t)$ sa vitesse de

déplacement. A chaque itération, toute particule évalue la qualité de sa position sur une fonction objective $f(x_i)$ définie, et mémorise à chaque fois sa meilleure position $\overrightarrow{p}_i(t)$ trouvée. En même temps, elle communique avec les autres particules pour connaître la meilleure position $\overrightarrow{g}_i(t)$ trouvée par l'ensemble des particules. Ensuite, elle modifie sa vitesse et sa position en fonction de ces informations [ShSi07]. Le mouvement d'une particule i appartenant à l'essaim peut donc être décrit par les équations suivantes :

$$\begin{aligned}\overrightarrow{V}_i(t+1) &= \overrightarrow{a} \odot \overrightarrow{V}_i(t) + \overrightarrow{r}_1 \odot \overrightarrow{b}_1 \odot \big(\overrightarrow{p}_i(t) - \overrightarrow{x}_i(t)\big) + \overrightarrow{r}_2 \odot \overrightarrow{b}_2 \odot \big(\overrightarrow{g}_i(t) - \overrightarrow{x}_i(t)\big) \\ \overrightarrow{x}_i(t+1) &= \overrightarrow{c} \odot \overrightarrow{x}_i(t) + \overrightarrow{d} \odot \overrightarrow{V}_i(t+1)\end{aligned} \tag{3.1}$$

Le symbole $\odot$ défini la multiplication vectorielle élément par élément. $\overrightarrow{a}$ est le coefficient d'inertie, $\overrightarrow{b}_1$ et $\overrightarrow{b}_2$ sont deux vecteurs réels représentant l'intensité d'attraction, $\overrightarrow{r}_1$, $\overrightarrow{r}_2$ deux vecteurs de valeurs aléatoires $\in[0\ 1]$ et $\overrightarrow{c}$, $\overrightarrow{d}$ sont des paramètres du modèle utilisés pour mettre à jour les positions des particules. En pratique, les éléments du vecteur $\overrightarrow{a}$ doivent varier entre des valeurs un peu inférieures à 1 comme il sera démontré dans la suite de cette section. Plus ces coefficients sont proches de 1 meilleure est l'exploration de l'espace de recherche, au détriment, néanmoins, de la vitesse de convergence.

Trois tendances de recherche et de déplacement sont évoquées :

- la tendance «aventureuse» consistant à continuer selon la vitesse actuelle ;
- la tendance «conservatrice» consistant à revenir vers la meilleure position déjà trouvée ;
- la tendance «panurgienne» consistant à s'orienter vers la meilleure particule.

Il apparait dans l'équation 3.1 que les composants des variables $\overrightarrow{x}_i(t)$ et $\overrightarrow{V}_i(t)$ sont mises à jour indépendamment les unes des autres. Le seul lien entre ces dimensions est introduit dans la fonction objective $f(x_i)$ avec les solutions $\overrightarrow{p}_i(t)$ et $\overrightarrow{g}_i(t)$. Sans perte de généralité, l'algorithme décrit dans l'équation 3.1 peut être réduit et analysé dans le cas d'une seule dimension, tel que :

$$\begin{aligned}V_i(t+1) &= aV_i(t) + r_1 b_1\big(p_i(t) - x_i(t)\big) + r_2 b_2\big(g_i(t) - x_i(t)\big) \\ x_i(t+1) &= cx_i(t) + dV_i(t+1)\end{aligned} \tag{3.2}$$

L'algorithme PSO est simple à programmer et à utiliser, et peut être considéré comme étant une méthode itérative du fait que la solution est approchée peu à peu, et "stochastique" car des paramètres aléatoires sont considérés. Il est important de souligner les points suivants :

- l'initialisation de l'essaim dans l'espace de recherche peut soit se faire de manière arbitraire, soit de manière régulière. Dans certains cas une combinaison des deux méthodes est favorisée ;
- l'initialisation des vitesses est aussi aléatoire, mais on fixe une valeur Vmax pour éviter que le système explose ;

- La notion de voisinage non retenue dans notre approche, consiste à définir pour chaque particule un nombre de particules appelées voisinage. Deux grandes méthodes existent, soit un voisinage géographique, qui doit être recalculé à chaque itération, soit un voisinage social «le plus utilisé» défini une fois pour toutes [CLERC02] ;
- Un certain nombre de paramètres doivent être définis par l'utilisateur : la taille de l'essaim, la méthode d'initialisation, le voisinage et les différents coefficients et leurs bornes de variations.

A noter aussi qu'il existe d'autres versions de l'algorithme PSO, avec des modifications et des améliorations quant aux choix des paramètres, des coefficients, des méthodes d'initialisations et des choix du type de voisinage. Des versions adaptatives du PSO telles que l'APSO ont été également proposées dans la littérature pour résoudre certains problèmes d'optimisation particuliers [LuCi08][CaCh04].

3.3.2 Étude dynamique de l'algorithme PSO

Dans la suite de ce chapitre de thèse, pour des raisons pratiques et de simplification, la version déterministe de l'algorithme PSO est considérée [Bai10][Trelea03]. Dans ce cas, les valeurs aléatoires r_1 et r_2 sont remplacées par leurs valeurs moyennes $\frac{1}{2}$ et $c = d = 1$. Après simplification, 3.2 peut s'écrire comme :

$$\begin{aligned} V_i(t+1) &= aV_i(t) + b\big(P_i(t) - x_i(t)\big) \\ x_i(t+1) &= x_i(t) + V_i(t+1) \end{aligned} \tag{3.3}$$

où $P_i(t) = \dfrac{b_1 p_1(t) + b_2 g_1(t)}{b_1 + b_2}$ and $b = \dfrac{b_1 + b_2}{2}$

Pour effectuer une analyse dynamique de l'algorithme, les équations 3.3 sont réécrites sous la forme matricielle suivante :

$$\begin{pmatrix} x_i(t+1) \\ V_i(t+1) \end{pmatrix} = A \begin{pmatrix} x_i(t) \\ V_i(t) \end{pmatrix} + BP_i(t) \tag{3.4}$$

où $\begin{pmatrix} x_i(t) \\ V_i(t) \end{pmatrix}$ est le vecteur d'état du système représentant la position de la particule et sa vitesse, P_i l'entrée du système, $A = \begin{bmatrix} 1-b & a \\ -b & a \end{bmatrix}$ est la matrice dynamique, et $B = \begin{bmatrix} b \\ b \end{bmatrix}$ la matrice d'entrée.

Le point d'équilibre du système, s'il existe, est l'état du système tel que :

$$\begin{pmatrix} x_i(t+1) \\ V_i(t+1) \end{pmatrix} = \begin{pmatrix} x_i(t) \\ V_i(t) \end{pmatrix} \tag{3.5}$$

on obtient alors :

$$\begin{pmatrix} x_i(t) \\ V_i(t) \end{pmatrix} = [I - A]^{-1} BP_i(t) \tag{3.6}$$

et donc :

$$\begin{pmatrix} x_i(t) \\ V_i(t) \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} P_i(t) \tag{3.7}$$

Le point d'équilibre du système est alors l'état du système tel que la particule est positionnée dans P_i avec $x_i(t) = P_i$, et a une vitesse égale à zéro $V_i(t) = 0$.

Il est important de déterminer si une particule i finira par se stabiliser au point d'équilibre (dans le cas où l'algorithme converge) et la façon dont laquelle elle se déplacera dans l'espace d'état. Les résultats standards de la théorie des systèmes dynamiques indiquent que le comportement temporel de la particule dépend des valeurs propres de la matrice dynamique $A(t)$. Les valeurs propres λ_1 et λ_2 sont les solutions de l'équation :

$$det(\lambda I - A) = 0 \tag{3.8}$$

i.e,

$$\lambda^2 - (a - b + 1)\lambda + a = 0 \tag{3.9}$$

Le comportement et l'évolution temporelle d'une particule dépendent des paramètres a et b. L'analyse de l'équation 3.9 permet de déterminer la zone de convergence.

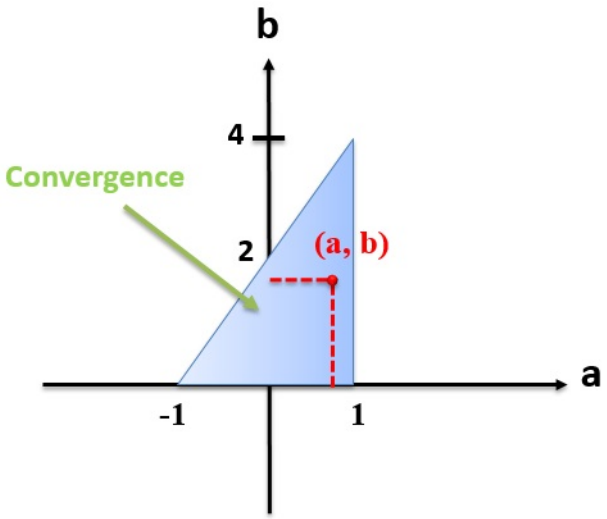

FIGURE 3.1 – Domaine de convergence du PSO

La figure 3.1 illustre l'ensemble des paires de valeurs des paramètres a et b permettant la convergence des particules de l'algorithme PSO vers l'état d'équilibre.

3.4 Approche de commande de flotte proposée

Dans la littérature, un certain nombre de problèmes liés à la commande de flotte de véhicules autonomes sont discutés. Différentes stratégies sont mises en œuvre afin de résoudre au mieux ces problèmes. Celles-ci peuvent être classées en trois approches principales : centralisée, décentralisée et distribuée. Chacune de ces approches a ses propres avantages et inconvénients. Le choix d'une stratégie dépend des contraintes de l'environnement et de l'objectif de la mission. Même si l'approche centralisée est généralement considérée comme le moyen le plus simple de contrôler un système multi-agent, cette approche nécessite un effort de calcul considérable. Une solution à ces problèmes serait d'utiliser l'approche décentralisée ou distribuée. Ces deux approches permettent de simplifier le calcul central en le distribuant sur tous les agents. Dans la commande distribuée, chaque agent est connecté uniquement avec ses voisins les plus proches et possède son propre régulateur qui reçoit l'information des états des autres agents mais aussi les partage avec les autres, alors que dans la commande décentralisée, chaque agent utilise uniquement ses propres informations d'état, même si cet agent échange également l'information avec d'autres voisins. Afin de bénéficier des avantages des deux approches, la robustesse et la tolérance aux défauts des régulateurs distribués, avec l'évolutivité et l'extensibilité des commandes décentralisées, nous proposons dans ce chapitre une nouvelle structure basée sur une architecture de commande décentralisée associée à une planification de trajectoire distribuée.

Toutes les méthodes illustrées ci-dessus sont souvent conçues pour une catégorie spécifique de système où la dynamique est souvent linéaire. L'application de l'une de ces approches est donc fortement liée aux types d'agents ou de véhicules qui forment la flotte. Également la prise en compte des perturbations et l'hétérogénéité du groupe constituent un problème majeur pour les preuves de stabilité, de consensus et autres. L'approche de commande de flotte proposée est donc partiellement indépendante de la dynamique générale des agents, ce qui la rend facilement réalisable sur plusieurs types de véhicules. L'architecture générale est une combinaison d'un contrôleur décentralisé et un planificateur de trajectoire distribué. L'avantage de cette combinaison est de dissocier la partie du contrôleur local de chaque véhicule avec la partie de contrôle de formation. Les trajectoires des véhicules sont générées en ligne et localement à la suite d'une optimisation distribuée de la fonction coût de chaque agent. Cette fonction de potentiel scalaire positif est construite de telle sorte que son minimum soit obtenu lorsque l'agent atteint les objectifs définis, ce sera dans notre cas :

- La convergence à la configuration souhaitée
- Suivi de la cible
- Évitement de collision

Il est supposé que chaque véhicule peut mesurer les positions de tous les agents dans son voisinage grâce à des techniques de vision et/ou des capacités de communication sans fil. Nous considérons le cas sans perte de données, de communication ou interruption.

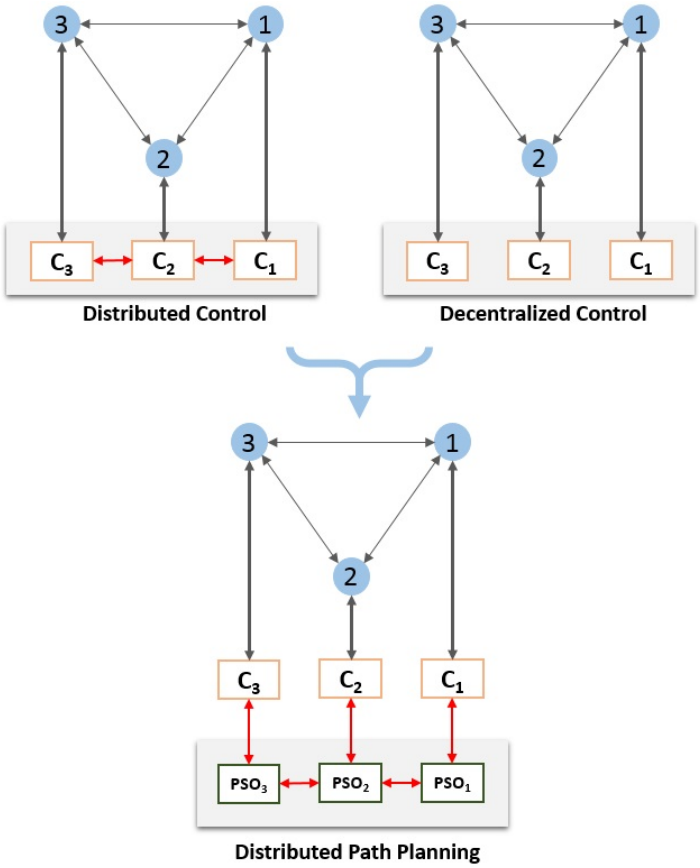

FIGURE 3.2 – Générations de trajectoires distribuées

3.4.1 Formulation de la problématique

En général, un agent se réfère à un système dynamique. Dans ce rapport de thèse, le terme «agent» est interchangeable avec "véhicule autonome". Chaque agent est équipé d'un contrôleur local qui assure la stabilité locale. Une flotte se compose d'un ensemble d'agents $N = 1, ..., n$ où n est le nombre de véhicules. Les informations échangées entre les agents sont modélisées au moyen de la théorie des graphes.

Décrivons le système par un graphique pondéré $G(N, \epsilon(t), A(t))$ Où N et $\epsilon(t) \in N \times N$ sont l'ensemble des nœuds et l'ensemble des arcs respectivement du graphe $G(t)$ et $A(t)$ une matrice de $M_n(R)$ d'éléments dans R tel que : $a_{ij}(t) > 0$ si : $(j, i) \in \epsilon(t)$.

G est supposé être un graphique non dirigé tel que : $\forall(i, j) \in \epsilon(t) \Rightarrow (j, i) \in \epsilon(t)$. Dans ce cas La matrice $A(t)$ est symétrique avec $a_{ij}(t) = a_{ji}(t)$.
Tout d'abord, définissons pour chaque nœud i un ensemble de voisins $\Xi_i(t)$ tel que :

$$\Xi_i(t) = \{j \in N : \|x_j(t) - x_i(t)\| \leq l\} \tag{3.10}$$

l définit la portée du voisinage. $\Xi_i(t)$ s'appelle le voisinage métrique de l'agent i. En-

suite, un deuxième type de voisinage appelé voisinage topologique, où dans ce cas, nous considérons seulement les λ éléments de l'ensemble $\Xi_i(t)$ les plus proches du nœud i. Les λ nœuds représentent un nouvel ensemble $\Xi_i(t)'$ avec $\Xi_i(t)' \subset \Xi_i(t)$.

La valeur de $a_{ij}(t)$, qui représente l'influence entre les agents, dépend de la différence entre $d_{ij}(t)$ la distance entre deux agents i et j avec la distance de sécurité c (où $c > l$).

Les ensembles d'éléments $d_{ij}(t)$ forment la matrice $\partial(t)$ à l'instant t, et d_{ij}^d les distances souhaitées entre les agents forment la matrice ∂_d.

L'objectif de la commande est alors :

1. amener la flotte d'une configuration géométrique initiale $\partial(t_0)$ vers une configuration métrique désirée ∂_d
$$\lim_{t \to +\infty} \partial(t) = \partial_d \tag{3.11}$$
2. Chaque agent doit assurer le suivi des cibles P_d
$$\forall i \in N, lim_{t \to +\infty} d_{ip}(t) = d_{ip}^d \tag{3.12}$$
où $d_{ip}(t)$ et d_{ip}^d sont respectivement la distance réelle et désirée entre l'agent i et le point cible.
3. Évitement de collisions entre les agents
$$\forall i, j \in N, i \neq j : \|x_i(t) - x_j(t)\| > c \tag{3.13}$$

La question est formulée comme un problème d'optimisation en ligne où des fonctions coût scalaires positives $\Lambda_i(t)$ sont construites dans chaque agent i de telle sorte que son minimum est obtenu lorsque la flotte atteint les objectifs définis. L'idée principale est de trouver à chaque étape le meilleur déplacement $h_i^*(t)$ minimisant au mieux la fonction coût, tel que :

$$h_i^*(t) = min(\Lambda_i(t)) \tag{3.14}$$

Et :

$$Y_{id}(t + \tau) = Y_i(t) + h_i^*(t) \tag{3.15}$$

Où $Y_{id}(t + \tau)$ est la trajectoire de référence souhaitée de l'agent i à chaque pas de temps τ.

La figure 3.3 illustre le schéma de contrôle général dans lequel une stratégie de planification de trajectoire distribuée basée sur l'algorithme d'optimisation PSO est proposée pour atteindre les objectifs.

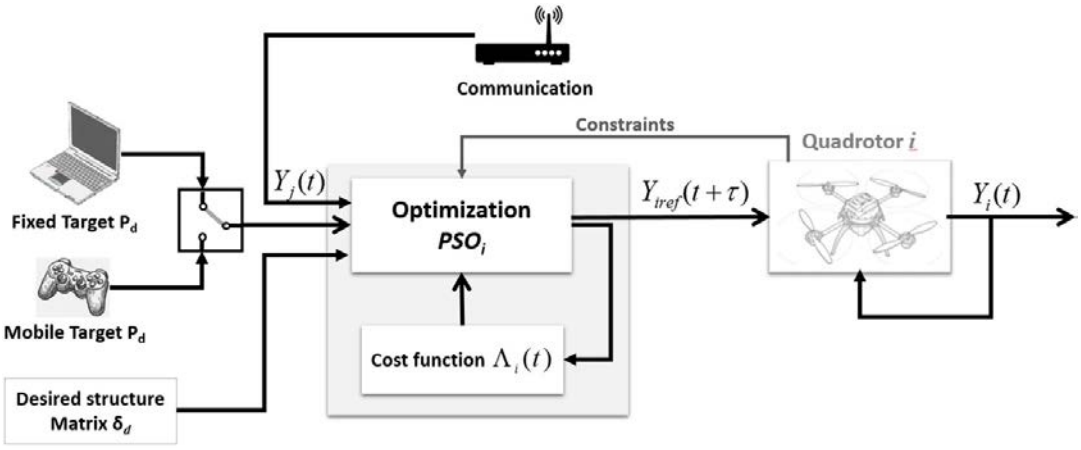

FIGURE 3.3 – General scheme

3.4.2 Élaboration de la fonction objective

Chaque agent dispose d'une fonction coût positif $\Lambda_i(t)$ indépendante des autres. Celle-ci est basée sur les distances entre agents et la position de la cible P_d. La construction de la fonction coût prend également en compte l'évitement de collisions.

Au début, définissons une fonction de coût $\Lambda_i(t)$ pour chaque agent i tel que :

$$\begin{aligned}\Lambda_i(t) &= \rho\Big(\|P_d - [x_i(t) + h(t)]\| - d_{ip}^d\Big) \\ &+ \sum_{j=1}^{q} a_{ij}(t)\Big(\|x_j(t) - [x_i(t) + h(t)]\| - d_{ij}^d)\Big)\end{aligned} \tag{3.16}$$

où :

$$a_{ij}(t) = 1 + exp\left(\frac{c - d_{ij}(t)}{\sigma}\right) \tag{3.17}$$

Avec $i \neq j$, $q = card(\Xi_i'(t))$, $h(t) \in R^2$, et $\rho >> 1$, et d_{ip}^d la distance désirée entre l'agent i et le point cible.

L'objectif principal est de trouver le meilleur vecteur $h_i^*(t)$ pour chaque agent i minimisant la fonction de coût $\Lambda_i(t)$ telle que :

$$\forall i \in N : \lim_{t\to\infty} \Lambda_i(t) = 0 \Rightarrow \tag{3.18}$$

$$\Rightarrow \begin{cases} \lim_{t\to\infty} \partial(t) = \partial_d \\ \forall i \in N, lim_{t\to+\infty} d_{ip}(t) = d_{ip}^d \\ \forall i, j \in N, i \neq j : \|x_i(t) - x_j(t)\| > c \end{cases}$$

Où $h_i^*(t)$ représente le déplacement désiré de l'agent i entre deux instants t et $t + \tau$. La trajectoire de référence à l'instant $t + \tau$ pour l'agent i devient :

$$h_i^*(t) = Y_{id}(t + \tau) - Y_i(t) \tag{3.19}$$

Le choix de la valeur de ρ dépend essentiellement du nombre de voisins de chaque agent et joue un rôle sur le taux de convergence vers la cible. Un compromis doit être trouvé entre le suivi de la cible et le contrôle de formation.

Durant l'évolution des véhicules dans l'espace R^2, l'anti-collision entre les agents doit être assurée. Cette contrainte est introduite dans la fonction de coût $\Lambda_i(t)$ par la fonction $a_{ij}(t)$ qui est d'autant plus grande quand $d_{ij}(t) < c$ et converge vers la valeur 1 quand $a_{ij}(t) >> c$. Chaque agent i est plus susceptible de favoriser les valeurs de $h_i(t)$ qui évite les distances $d_{ij}(t) < c$ et minimise la fonction coût $\Lambda_i(t)$, avec l'algorithme PSO.

Afin d'avoir un comportement plus stable de la flotte à proximité du minimum, une valeur minimale de la fonction coût Λ_{iMIN} est considérée comme satisfaisante. Dans ce cas, quand $\Lambda_i(t) < \Lambda_{iMIN}$, alors la valeur de $h_i^*(t)$ est le vecteur nul.

Le choix du pas de temps τ ainsi que les intervalles de recherche des déplacements $h_i(t)$ de chaque agent sont obtenus expérimentalement et sont considérés comme des contraintes d'optimisation, tel que :

$$h_{iMIN}(t)/_\tau < h_i(t)/_\tau < h_{iMAX}(t)/_\tau \tag{3.20}$$

Le choix de $h_i(t)$ dépend donc logiquement du choix de τ.

L'algorithme 1 permet de décrire en détail les étapes pour la génération de trajectoire dans chaque agent de la flotte.

Algorithm 1 Génération de trajectoire

. Initialisation des paramètres
. Définir la configuration désirée pour la flotte avec la matrice ∂_d
repeat {à chaque pas de temps}
  . Calculer la matrice $\partial(t)$
  if $\Lambda_i(t) \leq \Lambda_{iMIN}$ **then**
    . $h[h_x, h_y] = [0, 0]$
    . Calculer la prochaine position x_{id} et y_{id} au temps $t + \tau$:
    $x_{id}(t + \tau) = x_i(t)$
    $y_{id}(t + \tau) = y_i(t)$
  else if $\big((\exists d_{ij}(t) < c) or (t\%\tau == 0)\big)$ **then**
    . Trouver tous les voisins de l'agent i dans l'ensemble $\Xi_i(t)$
    . Minimiser la fonction objective $\Lambda_i(t)$ par l'algorithme PSO
    . choisir le vecteur $h[h_x, h_y]$ qui minimise la fonction coût that $\Lambda_i(t)$
    . Calculer la prochaine position x_{id} et y_{id} au temps $t + \tau$:
    $x_{id}(t + \tau) = x_i(t) + h_x$
    $y_{id}(t + \tau) = y_i(t) + h_y$
  end if
until la fin de l'expérimentation

Afin de valider l'approche de commande de flotte proposée dans ce chapitre, des tests expérimentaux sont proposés dans la suite de ce manuscrit sur la commande d'une flotte de quadrirotors. Les résultats obtenus sont présentés dans les sections suivantes.

3.5 Résultats de simulation

Tout d'abord, l'approche de commande de flotte proposée avec l'algorithme PSO a été implémentée dans un simulateur de drones pour la commande dans une flotte d'UAVs composée de trois qudrirotors. La figure 3.4 montre une image du simulateur utilisé.

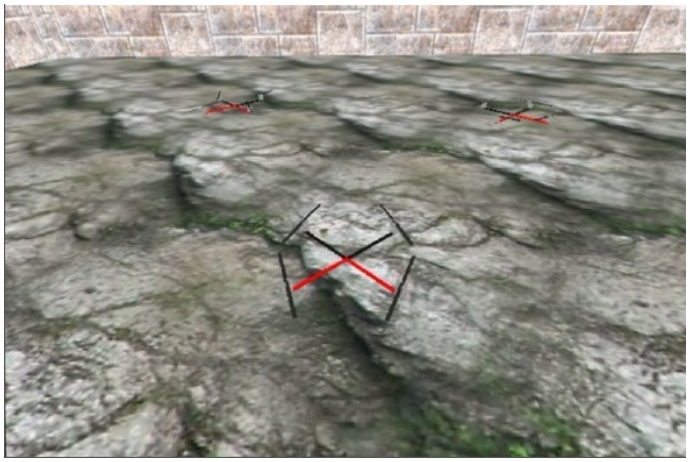

FIGURE 3.4 – Simulateur de vol en formation

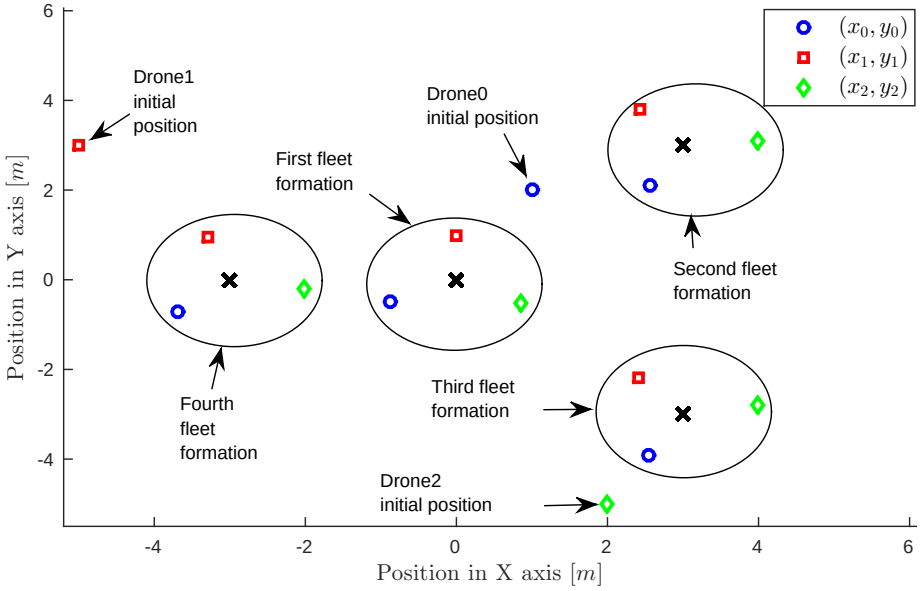

FIGURE 3.5 – Formation de la flotte autour des points références

Dans cette simulation, la position de la cible P_d et la configuration désirée des UAVs définies dans la matrice ∂_d sont fixées. Les éléments $d_{ij}(t)$ sont définis de telle sorte que la distance entre les UAVs soit homogène, ce qui conduit dans ce cas à une configuration triangulaire autour du point cible, figure 3.5.

Un ensemble de quatre points de référence sont transmis successivement à chacun des quadrirotors. Chacun des UAVs calcule indépendamment sa trajectoire pour se rendre d'un point référence à un autre en tenant compte de la position des autres agents. Les UAVs sont initialisés à des positions et vitesses aléatoires, en tenant compte des critères annoncés précédemment. La figure 3.6 illustre les trajectoires suivies par chaque UAV pour arriver à la formation désirée.

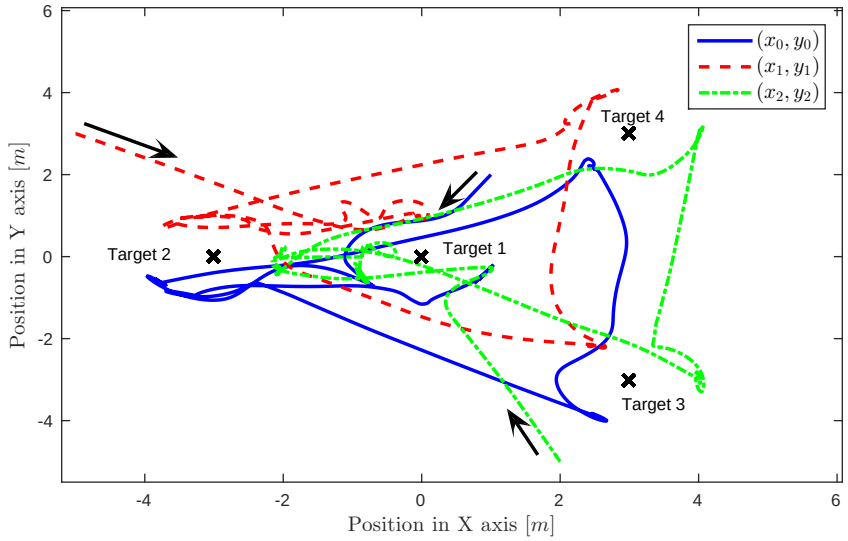

FIGURE 3.6 – Trajectoires suivies par chacun des UAVs

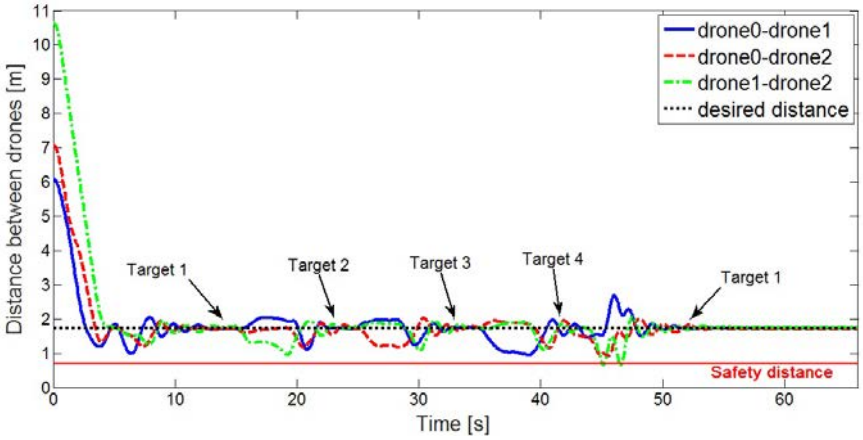

FIGURE 3.7 – Évolution et convergence des distances entre les UAVs

La configuration désirée entre les trois UAVs est une formation triangulaire uniforme avec une distance souhaitée de $1m$ vers la cible, et une distance entre agents égale à : $d_{agents} = 2(1m)\cos(\pi/6) = 1.732m$. La figure 3.7 illustre la convergence de toutes les distances entre les agents à la valeur souhaitée, assurant ainsi la formation de la flotte autour de la cible. Il convient de noter que la distance de sécurité est respectée tout au long de la simulation.

Les résultats obtenus sur le simulateur sont très satisfaisants et permettent d'obtenir une commande de flotte intéressante et cela à partir de conditions initiales différentes des agents. En vue de ces résultats de simulation et afin d'évaluer le comportement de l'algorithme dans un environnement réel, la commande a été implémentée par la suite sur des véhicules aériens réels. Les résultats obtenus sont exposés dans les sections suivantes.

3.6 Résultats expérimentaux

3.6.1 Expérimentation de l'approche sur un seul UAV

L'approche proposée a d'abord été implémentée dans les premiers tests sur la commande en temps réel d'un seul quadrirotor. L'objectif de la commande de l'UAV a été alors de générer la trajectoire la plus optimale en utilisant l'algorithme PSO, pour se rendre d'un point désiré à un autre ou pour suivre une trajectoire donnée. Pour illustrer la capacité de l'algorithme à planifier des trajectoires dans un environnement réel, des obstacles sont ajoutés sur la trajectoire du quadrirotor.

Les paramètres de simulation considérés sont illustrés dans le tableau 3.2.

ρ	σ	c	d_{obs}
5	0.4	$0.8m$	$3m$
h_{xi}, h_{xi}	τ	d_{ij}^d	d_d
$\in [-0.4m, 0.4m]$	$0.1s$	1.732	0

TABLE 3.1 – Paramètres considérés de l'algorithme PSO

Les quadrirotors utilisés dans toutes les expériences sont des Parrot ARDrone2, dont les programmes internes ont été modifiés pour pouvoir gérer notre code personnalisé. Le framework utilisé dans ce drone fait partie des plates-formes créées par l'équipe du Laboratoire Heudiasyc de l'Université de Technologie de Compiègne.

Dans cette première expérience, trois cibles/références sont transmises consécutivement au quadrirotor en utilisant une station au sol. L'algorithme d'optimisation calcule à chaque pas de temps la position souhaitée en prenant en compte les contraintes dynamiques de l'UAV.

La figure 3.8 et 3.9 illustrent la trajectoire calculée par le PSO et sa trajectoire réelle dans l'espace 2D.

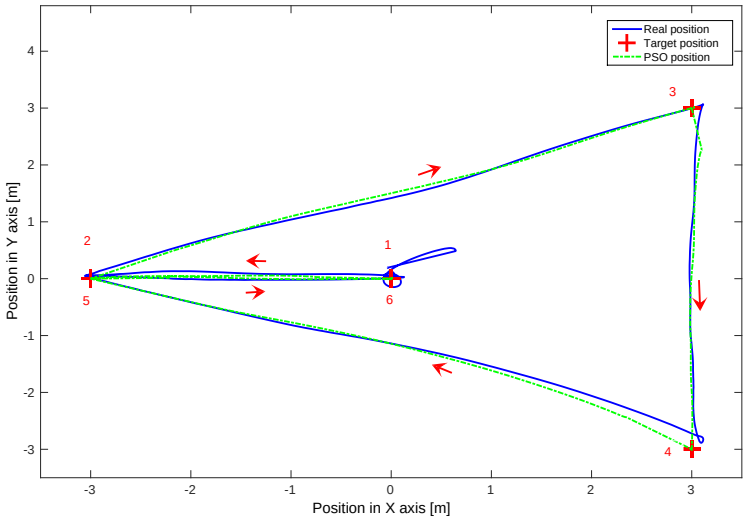

FIGURE 3.8 – Trajectoire et position du drone dans l'espace 2D pour la première expérience.

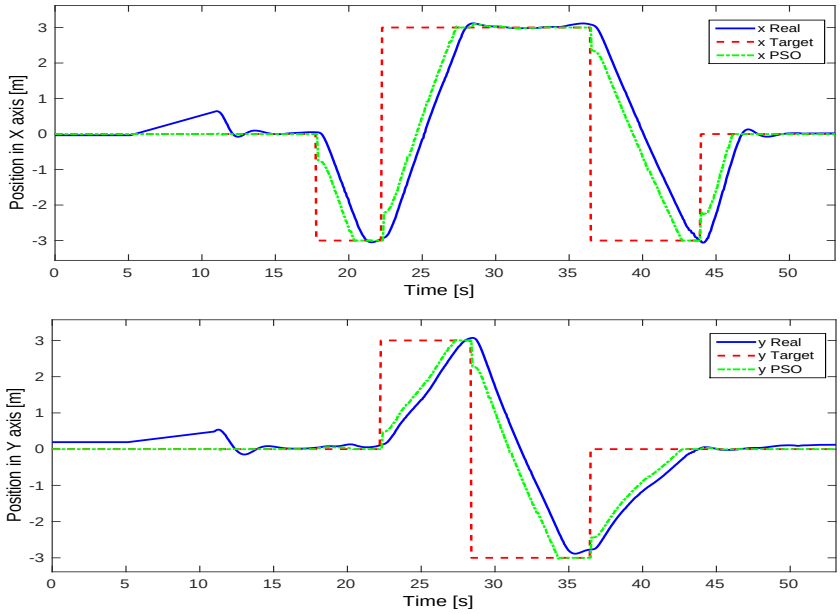

FIGURE 3.9 – Trajectoire et position du drone sur les axes x et y

La trajectoire calculée par le PSO permet effectivement de faire converger de manière stable le quadrirotor vers les points cibles désirés. Ces résultats montrent également la

capacité de notre algorithme à régir en temps réel et à optimiser la fonction objective de manière efficace. Les simulations et expérimentations peuvent être retrouvées dans le lien suivant :`https://www.youtube.com/watch?v=QLRl0Js72L0`.

Dans la deuxième expérience, afin d'illustrer la capacité de notre approche à planifier des trajectoires dans des environnements dynamiques, des obstacles ont été ajoutés aléatoirement dans l'environnement de l'UAV.

Au début, définissons une nouvelle fonction coût $\Lambda_i(t)$ permettant de tenir compte de la présence d'obstacles tel que :

$$\begin{aligned}\Lambda_i(t) = \rho\Big(\|P_d - [x_i(t) + h]\| - d_{ip}^d\Big) \\ + \sum_{j=1}^{q} a_{ij}(t)\Big(\|x_j(t) - [x_i(t) + h]\| - d_{ij}^d)\Big) + \sum_{k=1}^{m} Obs_{ik}(t)\end{aligned} \tag{3.21}$$

Avec $i \neq j$, $q = card(\Xi(t))$, $h \in R^2$, et $\rho >> 1$, d_{ip}^d représentant la distance souhaitée entre les UAV_i et la cible, avec m_i le nombre d'obstacles détectés par le i^{me} UAV.

Pour prendre en compte l'évitement des obstacles dans la stratégie de commande de la flotte, un terme supplémentaire est introduit dans la fonction coût de sorte que :

$$Obs_{ik}(t) = exp\left(\frac{c - \Big(\|x_{kobs}(t) - [x_i(t) + h]\|\Big)}{\sigma}\right) \tag{3.22}$$

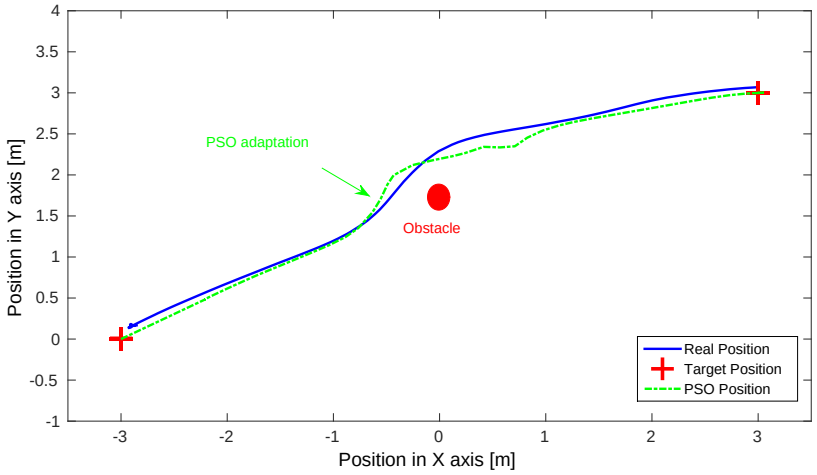

FIGURE 3.10 – Correction de la trajectoire de l'UAV pour éviter une collision avec un obstacle.

La valeur de la fonction $Obs_{ik}(t)$ dépend de la différence entre la distance entre l'agent i et l'obstacle k détecté à une position $x_{kobs}(t)$ à l'instant t, en prenant en compte une

certaine distance de sécurité c. La fonction $Obs_{ik}(t)$ converge alors vers zéro quand cette distance est beaucoup plus grande que la distance de sécurité et prend des valeurs importantes quand celle-ci se rapproche de cette distance c. Le paramètre σ est quant à lui introduit pour pondérer la fonction $Obs_{ik}(t)$. Il faut noter également que les positions des obstacles sont supposées inconnues au préalable et sont détectées à une distance égale à d_{obs}.

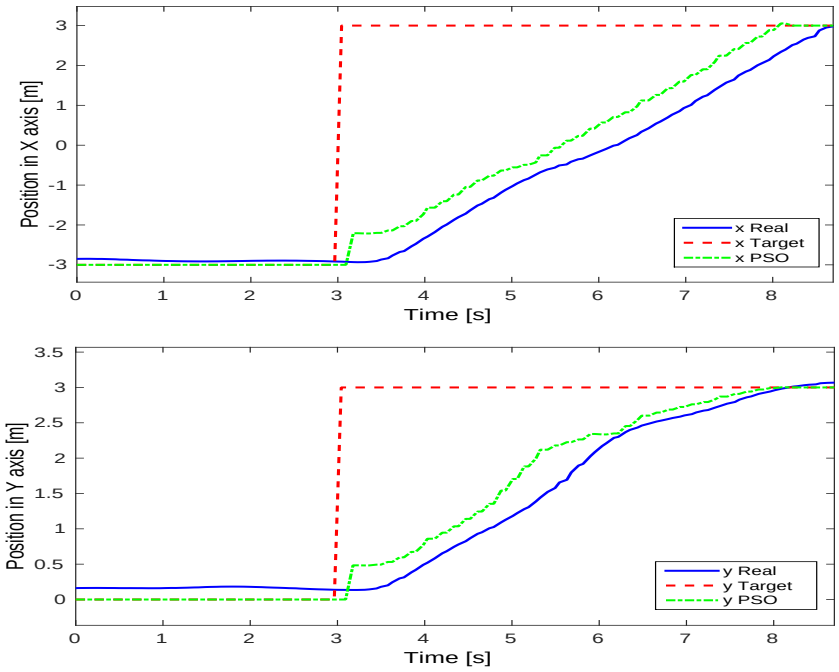

FIGURE 3.11 – Trajectoire de l'UAV sur les axes x et y dans le cas de détection d'un obstacle.

La figure 3.11 illustre la génération de la trajectoire sur les axes x et y. Lorsque l'UAV approche de l'obstacle, l'algorithme PSO adapte la trajectoire pour éviter la collision, voir la figure 3.10.

Les résultats obtenus montrent l'efficacité de l'approche proposée dans la planification de trajectoire en considérant les contraintes dynamiques du véhicule et de l'environnement. Dans la section suivante, l'approche de génération de trajectoire distribuée par PSO est implémentée sur une flotte de trois quadrirotors. Les calculs des trajectoires et des commandes sont exécutés à bord des UAVs.

3.6.2 Implémentation de l'approche sur une flotte de quadrirotors

L'approche proposée a été testée en temps réel sur une flotte de trois quadrirotors. Tous les UAVs sont commandés avec des contrôleurs similaires et indépendants les uns des autres. Le positionnement des agents est estimé à l'aide du système OptiTrack, puis transmis à l'ensemble des drones.

Chaque quadrirotor calcule sa propre trajectoire, de sorte qu'une formation uniforme se forme autour d'une cible donnée tout en évitant les collisions. Une formation triangulaire est attendue dans cette expérimentation.

Les paramètres considérés dans l'algorithme sont représentés dans le tableau 3.2.

ρ	σ	c	Λ_{iMIN}
5	0.4	$0.5m$	0.1
h_{xi}, h_{xi}	τ	d_{ij}^{d}	d_d
$\in [-0.4m, 0.4m]$	$0.1s$	1.732	3.464

TABLE 3.2 – Paramètres de PSO considérés dans le cas de commande de la flotte

Dans cette troisième expérimentation, la configuration souhaitée pour les UAVs est représentée dans la matrice ∂_d. Les éléments $d_{ij}(t)$ sont définis de telle sorte que la distance désirée entre les UAVs soit homogène, ce qui se traduit par une configuration triangulaire autour du point cible, comme illustré dans la figure 3.12.

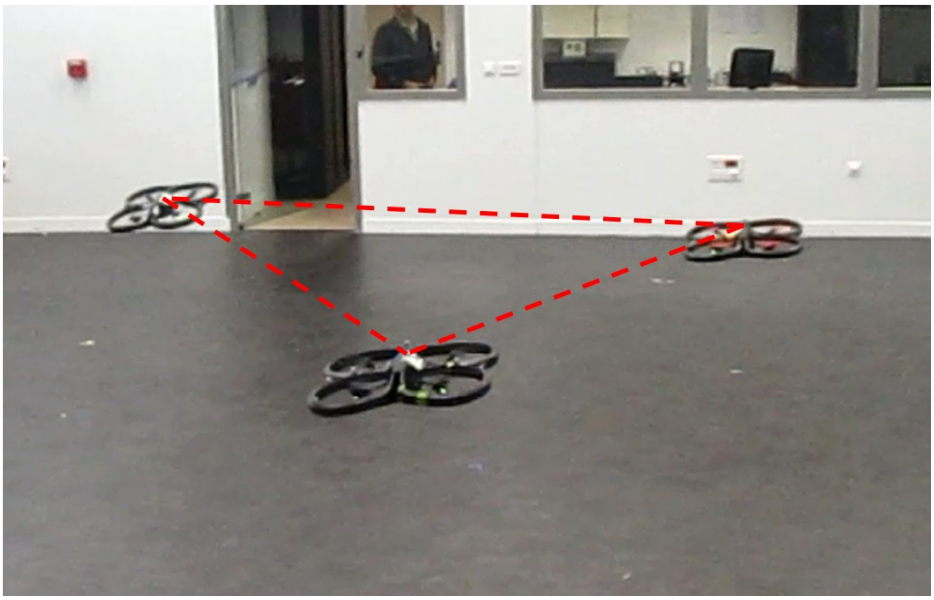

FIGURE 3.12 – Trois quadrirotors en formation

La figure 3.13 illustre le déplacement de la flotte avec les trajectoires des quadrirotors dans l'espace 2D. Quatre références (Targets) sont transmises successivement aux quadrirotors. L'algorithme d'optimisation dans chaque UAV calcule à partir de la position initiale (Start), la trajectoire de référence permettant d'atteindre les points cibles en assurant l'anti-collision des drones. Une forme triangulaire est réalisé autour des points cibles.

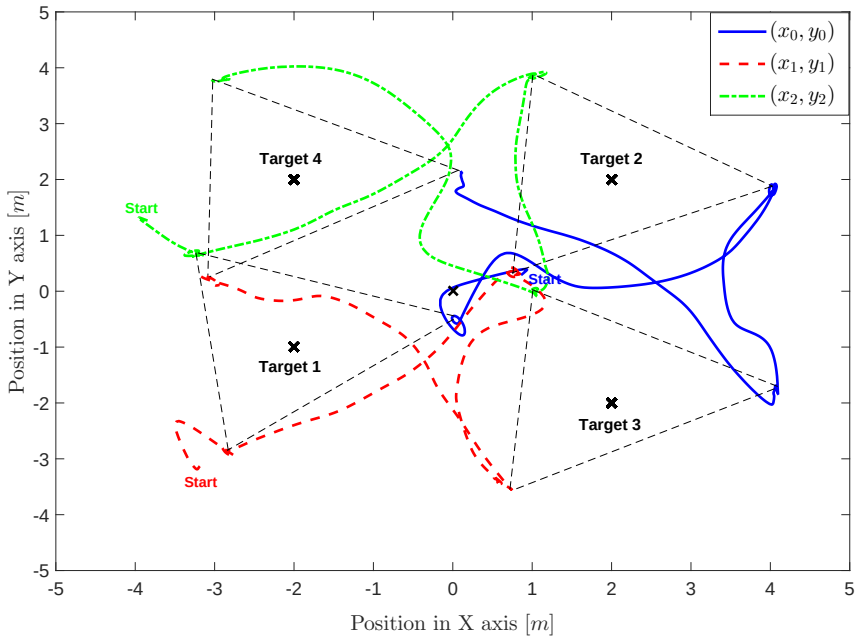

FIGURE 3.13 – Trajectoires des UAVs dans l'espace 2D dans le cas de la commande de flotte

De même que dans les tests sur le simulateur de drone, l'objectif est d'atteindre une configuration triangulaire uniforme, avec une distance désirée de $1.9m$ entre les agents et les points cibles, et une distance $d_{agents} = 2(1.9m)\cos(\pi/6) = 3.3m$ entre les UAVs.

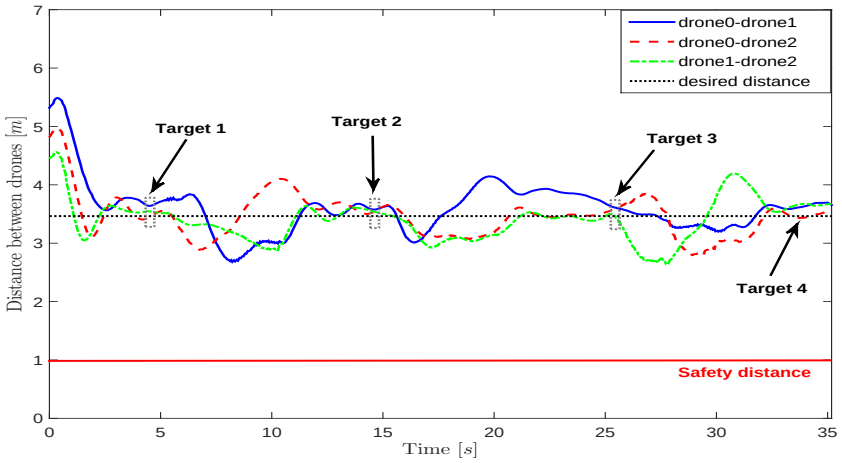

FIGURE 3.14 – Distances entre les UAVs en temps réel

Les trajectoires calculées par les algorithmes d'optimisation permettent de faire converger les distances entre les agents vers les valeurs désirées et du fait atteindre une configuration en triangle autour des points cibles comme illustré dans la figure 3.14. Ces distances

restent tout de même variables autour de la valeur souhaitée entre les targets en raison du déplacement des UAVs et des différentes perturbations présentent dans l'environnement. Nous remarquons également que la distance de sécurité est respectée tout au long de l'expérimentation, nous pouvons en conclure que l'anti-collision est donc garantie.

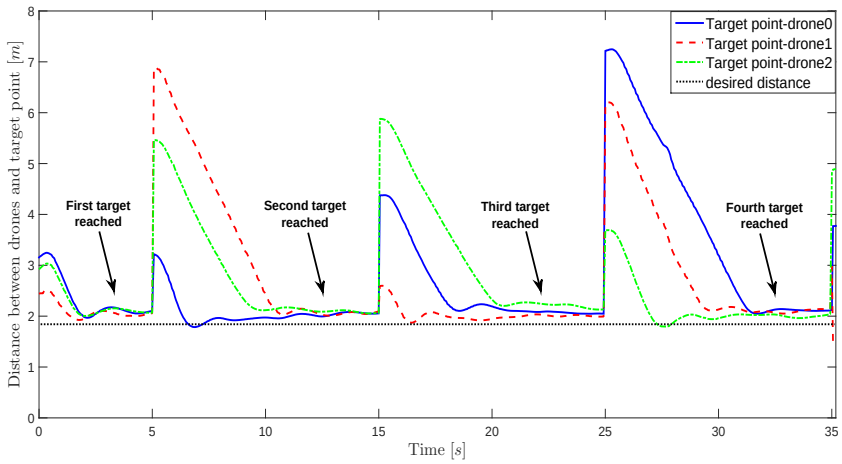

FIGURE 3.15 – Distances entre les UAVs et la référence en temps réel

La figure 3.15 représente les distances entre chaque UAV et les points targets. A chaque fois qu'un point cible est défini, les drones se déplacent vers celui-ci et restent à une distance égale à $1.9m$ préalablement définie. La flotte arrive donc à suivre les points cibles et à se positionner autour.

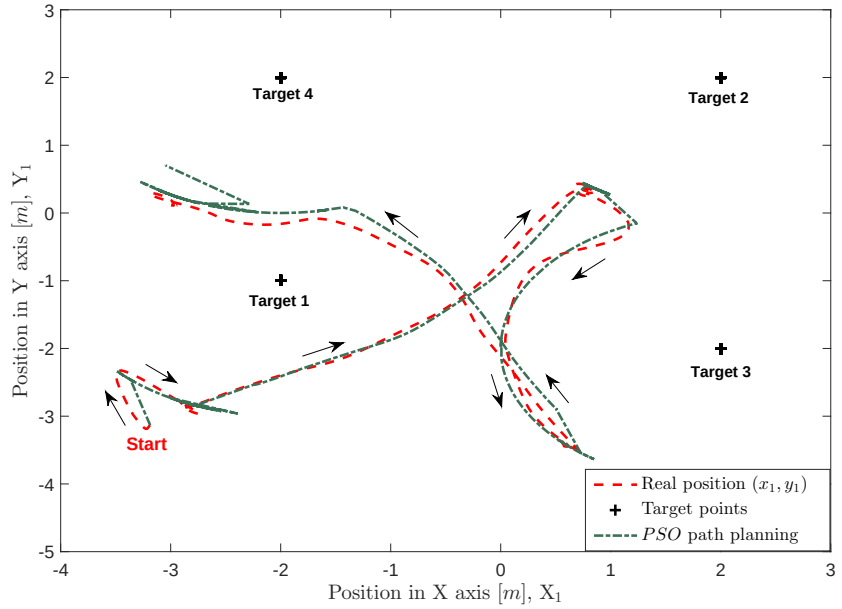

FIGURE 3.16 – Trajectoire calculée et réelle du premier quadrirotor dans l'espace 2D

Pour mieux illustrer le comportement individuel de chaque UAV, les figures 3.16 et 3.17 représentent la génération et le suivi de la trajectoire du quadrirotor 1. La trajectoire calculée par l'algorithme PSO est adéquate et permet la commande en formation de la flotte.

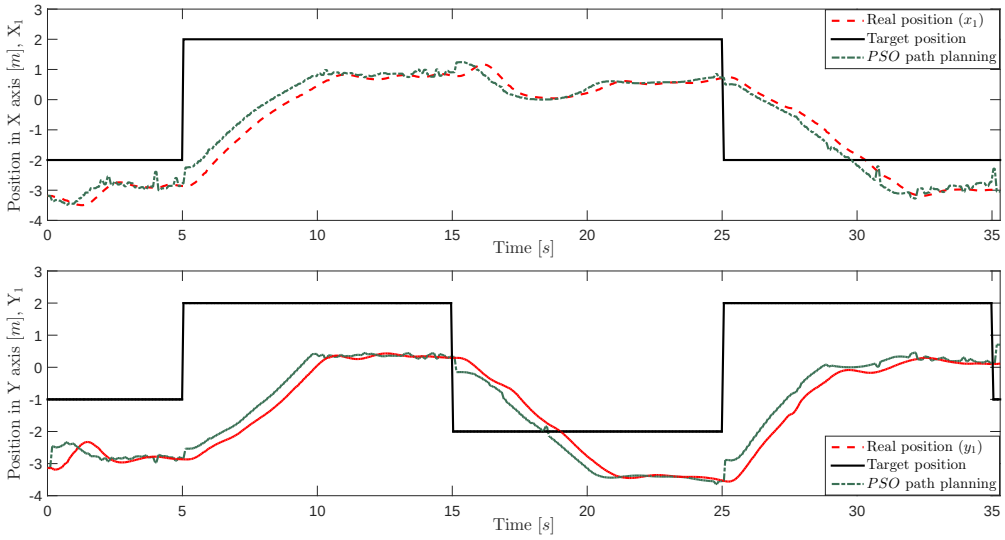

FIGURE 3.17 – Trajectoire calculée et réelle du premier quadrirotor sur les axes x et y

Les résultats expérimentaux obtenus et présentés dans ce chapitre sont très prometeurs. Les trajectoires générées par les algorithmes d'optimisation distribués sur chacun des drones, permettent d'obtenir une commande de flotte cohérente et stable. Il faut signaler également que les contrôleurs internes des quadrirotors peuvent être réglés davantage pour obtenir un meilleur suivi des trajectoires calculées.

3.7 Conclusion

Nous avons présenté dans ce troisième chapitre une nouvelle approche de commande de flotte basée sur une génération de trajectoires distribuée. Un bloc d'optimisation, implémenté sur chaque véhicule, permet en minimisant une fonction objective de calculer la trajectoire référence optimale garantissant la commande en formation de la flotte, le suivi de cible et l'évitement de collisions.

La minimisation des fonctions objectives dans chaque véhicule se fait d'une manière totalement distribuée et indépendamment les unes des autres. L'aspect temps réel nécessite également le choix d'un algorithme d'optimisation adapté avec un temps de calcul minimal. L'algorithme PSO (Particle swarm optimization) est très adapté pour ce type de problème d'optimisation dynamique. Il est également connu pour ses capacités à retrouver des solutions optimales avec un minimum de ressources. L'objectif de l'algorithme d'optimisation sera alors de trouver à chaque pas de temps le meilleur déplacement qui

minimise la fonction coût, en prenant en compte bien sûr les contraintes du système.

L'avantage aussi de notre approche est la capacité à dissocier la partie commande locale de chaque véhicule qui peut se faire d'une manière indépendante, avec la partie génération de trajectoire et commande de la flotte. Cela rend notre approche plus simple à implémenter sur des systèmes réels sans forcement changer les lois de commande internes. L'approche proposée a donc été implémentée en temps réel sur une flotte de trois quadrirotors. Les résultats obtenus sont prometteurs et montre l'efficacité de cette approche et son potentiel pour résoudre des problématiques liées à la commande de flotte.

Cette approche reste tout de même sensible aux erreurs de localisation qui peuvent engendrer des oscillations importantes sur les véhicules. Les résultats présentés dans ce chapitre ont été réalisés dans un environnement intérieur avec un système de détection de position très précis. Il serait intéressant de considérer le cas de présence d'erreur et de bruit dans la localisation des véhicules pour une implémentation dans un environnement externe.

Chapitre 4

Commande tolérante aux défauts pour la commande en formation

Sommaire

4.1 Introduction

La commande de formation s'appuie sur des techniques de commande avancées dont l'objectif est d'atteindre des performances élevées. Néanmoins, si un défaut ou une défaillance apparaît dans la flotte, ces stratégies de commande peuvent s'avérer très limitées, induisant des comportements indésirés et peuvent même conduire à l'instabilité des agents. Dans de telles situations il est nécessaire de développer des outils et des méthodes performantes tolérantes à des défauts (FTC). Par définition, la tolérance aux défauts d'un système est son aptitude à accomplir sa fonction malgré la présence ou l'occurrence de défauts, qu'il s'agisse de dégradations physiques du matériel ou de défauts logiciels. Elle apparait comme un moyen de garantir une sureté de fonctionnement.

Dans la littérature, deux approches de commande tolérantes aux défauts sont considérées, des approches passives et d'autre actives. Dans la première approche des techniques de commande robustes sont utilisées de façon à ce que le système bouclé devienne insensible à un ensemble connu de défauts. Les défauts sont alors pris en compte dans la conception initiale du système de commande. Le contrôleur est donc robuste aux défauts prédéfinis. Par contre, l'approche active réagit aux différents défauts en reconfigurant la loi de commande par l'utilisation des informations sur les défauts issus en ligne du système de diagnostic de manière à maintenir la stabilité et les performances de ce dernier. Cette approche permet notamment de traiter des défauts imprévus, mais nécessite une technique efficace de détection et d'isolation de défauts associée à une méthode de reconfiguration de la loi de commande.

Nous soulignons que la tolérance aux défauts dans les systèmes multi-agents est plus générale que sa définition pour un seul système. En réalité, le fonctionnement d'un SMA n'est pas affecté uniquement par l'apparition de défauts sur chaque agent mais également les collisions, la perte ou la dérive d'agents, la modification ou la perte du réseau de communication, peuvent être considérées comme des défauts pour un SMA. L'objectif donc de ce quatrième chapitre de thèse est d'étendre les résultats obtenus avec les deux techniques de commande en formation proposées dans les chapitres II et III aux cas de présence de défauts. Nous nous intéresserons plus particulièrement aux cas de perte d'agents et aux défauts actionneurs dans la flotte. Des simulations et des tests expérimentaux sur une flotte de qudrirotors sont réalisés afin de valider les approches proposées.

4.2 Approche de contrôle en formation tolérante aux défauts

La notion de sécurité dans les systèmes multi-agents et particulièrement dans la commande en formation, devient un champ de recherche intensif composé de la détection et de l'isolation des défauts (FDI) et de la tolérance au défauts (FTC). Un grand nombre d'études ont été menées sur ce sujet et divers résultats ont été obtenus. Les auteurs dans [Meskin2010][Meskin2009] ont développé un ensemble de filtres FDI pour détecter les défauts actionneurs en présence de graves perturbations environnementales. Ensuite, des

chaînes de Markov sont utilisées pour commander le système. D'autres travaux [Antonelli2013][Kempker2011] ont utilisé la détection de défauts en se basant sur un modèle pour générer des signaux résiduels pour les SMA. Dans [ZhWa14], le problème de consensus dans les systèmes multi-agents dans le cas de détection de défauts actionneurs est étudié. En adoptant une approche basée sur des actionneurs virtuels, il est démontré que les effets des défauts peuvent être efficacement compensés. Les auteurs dans [SaKr14] proposent une méthode tolérante aux défauts basée sur une commande Prédictive (MPC) pour la stabilisation et la navigation d'une formation hétérogène dans un espace 3D. Dans ce chapitre, nous présentons nos premiers résultats sur les commandes tolérantes aux défauts développées pour la commande en formation. Nous nous intéresserons particulièrement aux défauts actionneurs et perte d'agents dans la flotte. Nous supposons dans le reste de ce chapitre qu'un bloc de détection de défauts est préalablement implémenté en amont avec les contrôleurs de formation.

4.2.1 Commande en formation tolérante aux défauts basée consensus

Nous avons proposé dans le chapitre II une extension du modèle de flocking de Cucker-Smale. Les commandes des agents de la flotte, composées chacune de trois termes, arrivent à garantir ensemble : une convergence des vitesses des agents vers un consensus, l'évitement des collisions et le déplacement en formation tels que :

1. Toutes les différences de vitesse entre paire d'agents convergent asymptotiquement vers zéro

$$\forall i, j \in N : \lim_{t \to \infty} (v_i(t) - v_j(t)) = 0 \tag{4.1}$$

2. L'évitement de collision entre les agents est toujours assuré

$$\forall i, j \in N : d_{ij}(t) > r_0 \text{ with } d_{ij}(t) = ||q_i(t) - q_j(t)|| \tag{4.2}$$

3. La formation doit être atteinte asymptotiquement

$$\forall i, j \in N : \lim_{t \to \infty} d_{ij}(t) < R(n, r_0) \text{ avec } R(n, r_0) > r_0 \tag{4.3}$$

Le dernier point, qui consiste à assurer la cohésion de la flotte avec un déplacement en formation représente l'essentiel de notre contribution. Notre approche consiste à faire converger l'ensemble des agents à l'intérieur d'une surface (surface d'un cercle dans le cas 2D) définie par un rayon $R(n, r_0)$ et un centre représentant une moyenne pondérée des positions des agents de la flotte. L'intérêt de cette approche est qu'elle permet d'assurer un déplacement en formation sans prendre réellement en considération des inter-distances entre les agents, ce qui permet d'avoir une commande en formation plus dynamique, robuste, et plus apte à gérer les changement de dimension de la flotte dans le cas d'un ajout ou d'une perte d'agents qui pourrait être due par exemple à une défaillance de ces derniers.

La présence donc d'un défaut ou d'une défaillance sur un ou plusieurs agents de la flotte modifie le comportement de tout le groupe. Nous nous intéresserons dans cette section à trois défauts en particulier :

- a) Défaillance d'un ou plusieurs agents j :

$$v_j(t) = 0 \tag{4.4}$$

  La flotte devrait alors être en mesure de poursuivre la mission en modifiant la dimension de la surface de la formation.

- b) Présence momentanée d'un défaut sur un agent j :

$$v_i(t) = constant \tag{4.5}$$

  La flotte serait amenée à soutenir et accompagner l'agent défectueux jusqu'à ce qu'il reprenne son comportement normal.

- c) Perte de communication entre deux ou plusieurs agents (i, j) de la flotte :

$$a_{ij}(t) = 0 \tag{4.6}$$

  Les agents devraient rester en formation autour d'un ensemble de points représentant les centres des surfaces de chaque groupe d'agents.

La phase de détection, identification et isolation de défauts, étant supposée fonctionnelle avec un temps de délai négligeable. Un schéma de commande tolérante aux défauts est proposé afin de garantir un comportement adéquat de la formation en assurant les critères établis dans 4.1,4.2 et 4.3, et cela même dans le cas de présence de défauts.

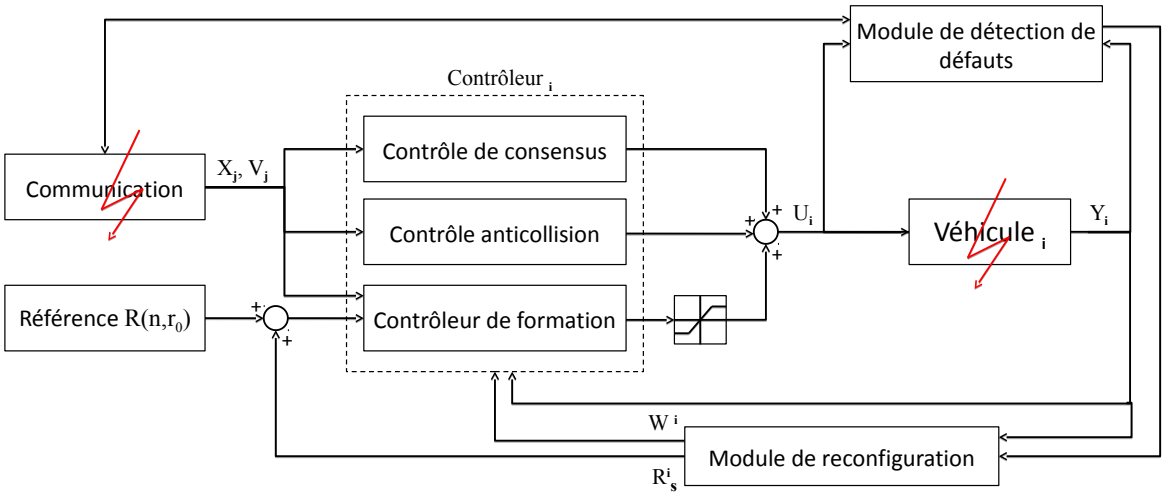

FIGURE 4.1 – Schéma de commande tolérante aux défauts pour la commande de la flotte

En introduisant le module FTC dans la commande de la flotte, l'équation de commande proposée dans 2.52 devient :

$$
\begin{aligned}
\dot{x}_i(t) &= v_i(t) \\
\dot{v}_i(t) &= \sum_{j=1}^{n} w_j a_{ij}(t)(v_j(t) - v_i(t)) \\
&\quad + \lambda_i(t) \sum_{j \neq i} f(||x_i(t) - x_j(t)||^2)(x_i(t) - x_j(t)) \\
&\quad + sat\left(\frac{H_c}{2}\left(sign(d_i^*(t) - R(n, r_0) + R_s^i) + 1\right)\frac{q^*(t) - q_i(t)}{d_i^*(t)}\right)
\end{aligned} \tag{4.7}
$$

Avec

$$
\lambda_i(t) = \left(\frac{1}{{n_i}^*}\sum_{i>j} w_j ||v_i(t) - v_j(t)||^2\right)^{1/2} \tag{4.8}
$$

$$
x_i^*(t) = \frac{1}{{n_i}^*}\sum_{j=1}^{n} w_j x_j(t) \text{ and } y_i^*(t) = \frac{1}{{n_i}^*}\sum_{j=1}^{n} w_j y_j(t) \tag{4.9}
$$

Où ${n_i}^* = \sum_{j=1}^{n} w_j$

Les modules FTC proposés permettent d'agir sur les contrôleurs des véhicules à partir des variables W^i et R_s^i et cela en fonction des défauts détectés par les modules FDI. Le vecteur $W^i = [w_1, ..., w_n]^T$ formé de n variables w_j permet en l'introduisant sur le contrôleur d'annuler (dans le cas d'une défaillance), de diminuer ou d'augmenter (dans le cas d'un défaut momentané) l'influence d'un agent j sur un autre agent i selon qu'on veuille accompagner ou non l'agent en question. Les valeurs des w_j sont alors définies tel que :

$w_j = 0$ dans le cas d'une perte de communication ou de défaillance de l'agent j.

$w_j = 1$ quand aucun défaut n'est détecté sur un agent j.

$w_j = w^*$ dans le cas de présence d'un défaut sur l'agent j

La variable R_s^i quant à elle permet de modifier le rayon du cercle d'évolution de l'agent i. Nous avons considéré (logiquement) que celui-ci devait diminuer dans le cas de perte d'un ou plusieurs agents pour favoriser la cohésion de la flotte et minimiser les distances entre les agents, et augmenter dans le cas de présence d'un défaut momentané sur un agent afin d'éviter au maximum les risques de collisions.

résultats de simulation

Afin de valider l'approche de commande tolérante aux défauts proposée dans cette section, des simulations ont été réalisées sur Matlab smulink sur un groupe de douze agents modélisés par des dynamiques en double intégrateur et initialisés à des positions et vitesses aléatoires. Deux situations différentes de présence de défauts sont traitées :

a) Perte d'agents

Dans la première simulation, trois agents sont simultanément détectés défectueux. La figure 4.2 montre l'évolution de la flotte dans l'espace 2D.

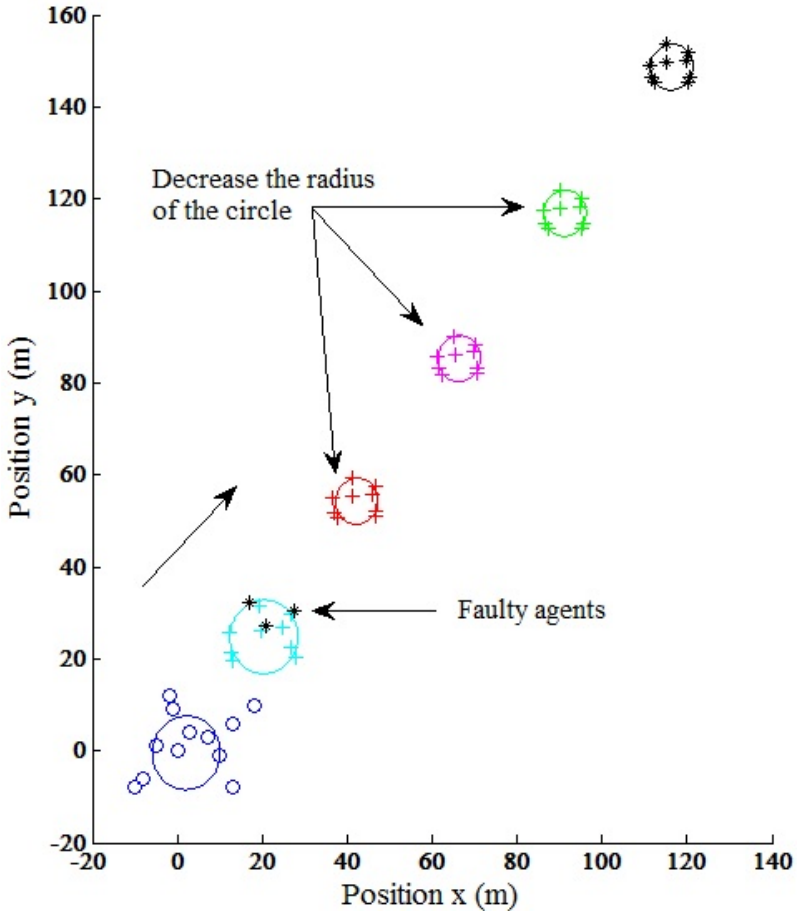

FIGURE 4.2 – Mouvement de flocking des véhicules dans le cas de perte d'agents

Les modules FTC tendent à éliminer l'influence des agents défectueux dans les lois de commande. Cela permet à la flotte de poursuivre son évolution et en même temps modifie la surface délimitant la formation en diminuant le rayon du cercle $R(n, r_0)$ avec R_s^i. Comme illustré dans la figure 4.2, tous les véhicules arrivent à converger à l'intérieur du cercle fictif de la formation.

La figure 4.3 montre l'évolution des composantes $\dot{x}_i(t)$ et $\dot{y}_i(t)$ des vitesses des véhicules dans le temps. On remarque à l'instant $t = 20s$ l'influence des agents défectueux sur le comportement de la flotte. Le consensus en vitesse est ensuite atteint à partir de $40s$ environ.

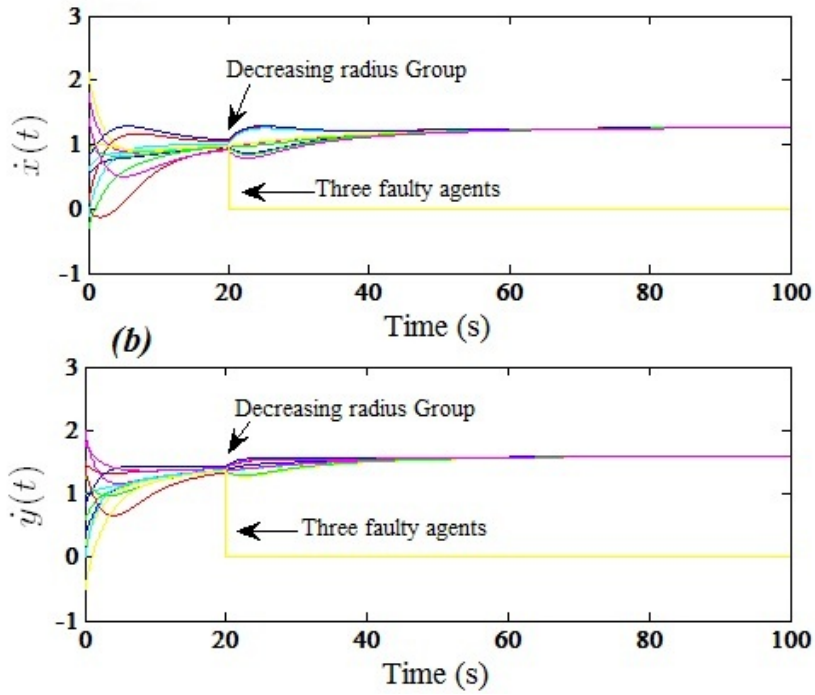

FIGURE 4.3 – Évolution des vitesse des drones au cours du temps

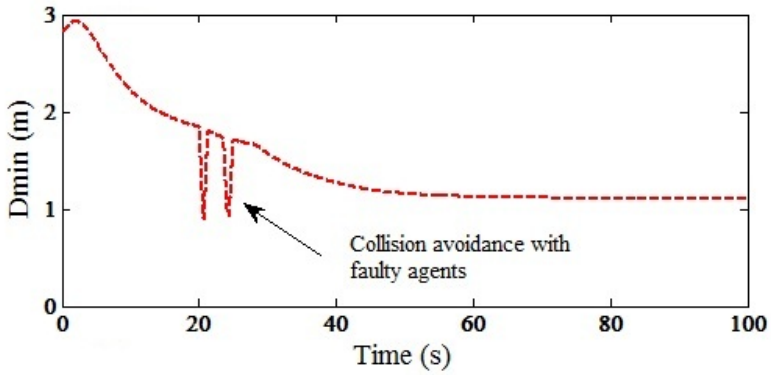

FIGURE 4.4 – Distances minimales entre les agents de la flotte dans le cas de perte d'agents

Afin de vérifier que l'anti-collision entre les véhicules est assurée à tout moment, l'évolution de la fonction D_{min} est présentée dans la figure 4.4, tel que :

$$D_{min} = \min_{i \neq j} ||x_i(t) - x_j(t)|| \tag{4.10}$$

Nous pouvons voir que les distances de sécurité entre les agents de la flotte sont toujours respectées et cela même dans le cas de détection d'agents défectueux comme illustré dans la figure 4.4 entre les instants [20s,25s].

b) Défaut actionneur sur un agent

Dans cette seconde expérimentation, un défaut actionneur est simulé sur un véhicule de la flotte. La vitesse de celui-ci est bloquée sur une valeur fixe entre l'intervalle de temps $[8s, 25s]$ comme illustré dans la figure 4.6.

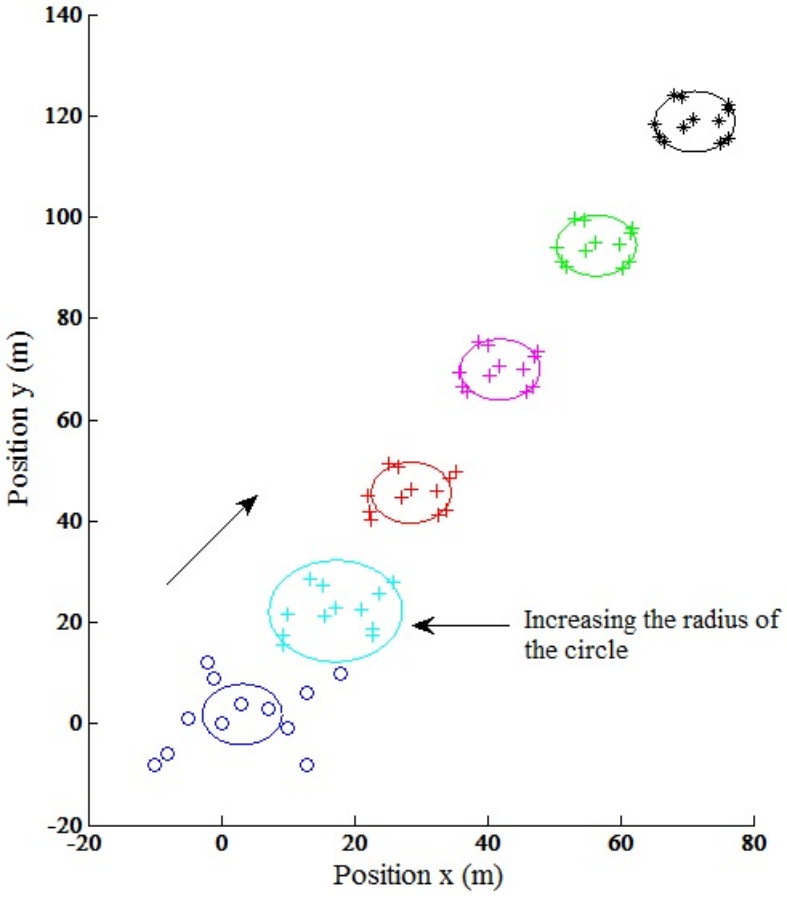

FIGURE 4.5 – Mouvement de flocking des véhicules dans le cas d'un défaut actionneur

Dans ce cas les modules FTC de chaque agent reconfigure les contrôleurs en augmentant le rayon du cercle de la formation afin d'éviter les risques de collisions avec l'agent défectueux 4.5. Les paramètres du vecteur W^i sont aussi modifiés de manière à accompagner le véhicule défectueux jusqu'à ce qu'il retrouve un comportement normal. On remarque ensuite dans la figure 4.6 que la dimension du cercle de formation reprend sa taille initiale après que l'agent défectueux ait repris son fonctionnement normal, figure 4.5.

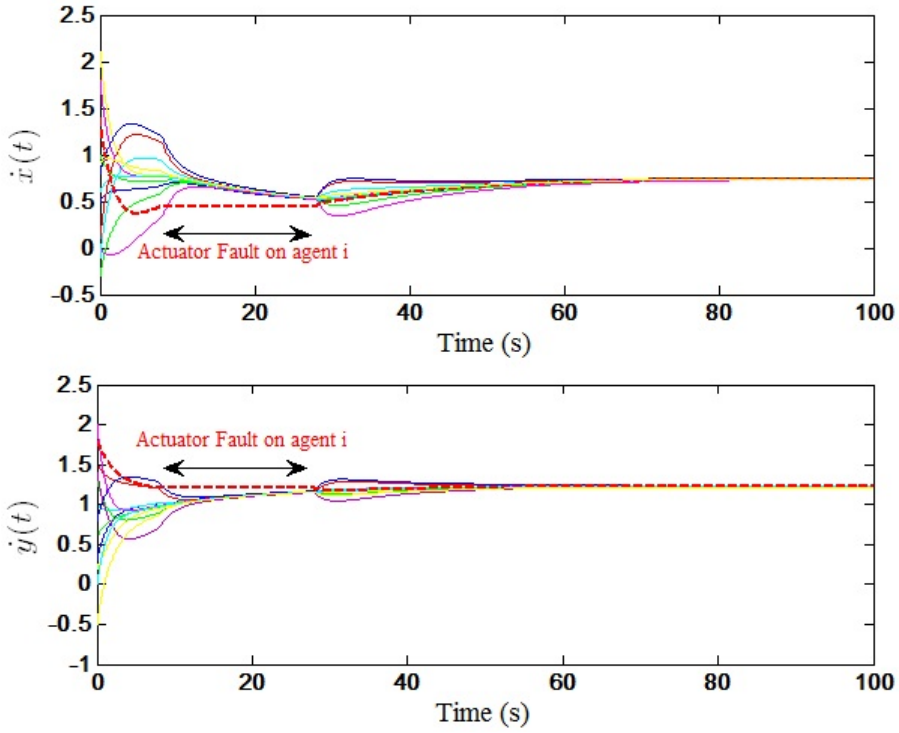

FIGURE 4.6 – Évolution des vitesses des drones au cours du temps. Cas d'un défaut actionneur

L'alignement en vitesse est également garanti dans cette situation, les agents de la flotte convergent tous vers la même vitesse comme illustré dans la figure 4.5. La figure 4.7 montre également que les distances entre les véhicules dans la flotte sont supérieures à la distance de sécurité r_0 tout au long du déplacement.

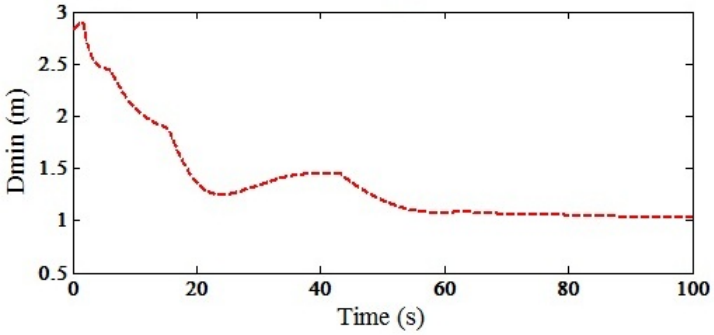

FIGURE 4.7 – Distances minimales entre les agents de la flotte dans le cas de perte d'agents

Les résultats présentés dans ces simulations montrent l'efficacité de l'approche proposée pour palier aux défauts actionneurs et aux cas de pertes d'agents dans la flotte. Le

consensus en vitesse reste toujours assuré avec une distance de sécurité respectée entre les agents. Des tests ont été réalisés avec des conditions initiales différentes. Les résultats restent concluant avec des temps de convergence dépendants des vitesses et positions de départ. La taille et la variation du rayon du cercle de formation dépend logiquement du nombre d'agents dans la flotte. La valeur de celui-ci a été fixée expérimentalement dans les simulations présentées. Nous pouvons considérer également une taille de cercle variable en fonction des vitesses des agents. Cette valeur serait d'autant plus grande que les vitesses des véhicules augmentent.

4.2.2 Commande en formation tolérante aux défauts basée sur une optimisation par PSO

Dans le chapitre III une commande en formation basée sur une planification de trajectoire distribuée est proposée pour la commande d'une flotte d'UAVs. L'avantage de cette approche est qu'elle permet de dissocier le contrôleur local de chaque véhicule avec le module de commande de formation, ce qui facilite son intégration sur différents types et dynamiques de véhicules.

L'objectif de la commande est présenté sous la forme d'un problème d'optimisation distribuée. Des modules d'optimisation basés sur des algorithmes PSO, sont implémentés dans chaque UAV_i et arrivent à définir à chaque pas de temps leurs meilleurs déplacements $h_i^*(t)$ qui minimisent les fonctions objectives $\Lambda_i(t)$, afin de produire un déplacement en formation garantissant :

1. La convergence de la flotte d'une configuration géométrique initiale $\partial(t_0)$ vers une configuration métrique désirée ∂_d

$$\lim_{t\to+\infty} \partial(t) = \partial_d \tag{4.11}$$

2. Le déplacement de la flotte vers des points cibles P_d

$$\forall i \in N, lim_{t\to+\infty} d_{ip}(t) = d_{ip}^d \tag{4.12}$$

3. L'évitement de collisions entre les agents

$$\forall i, j \in N, i \neq j : \|x_i(t) - x_j(t)\| > c \tag{4.13}$$

Dans cette section, nous nous intéresserons dans un premier temps à l'étude de la robustesse de l'approche proposée par rapport à différents types de perturbations. Des tests expérimentaux sont réalisés sur un seul puis sur une flotte de trois quadrirotors. Nous aborderons à la fin de cette partie des solutions aux cas de pertes d'agents dans la flotte.

Résultats expérimentaux

Les paramètres de commande utilisés dans la suite de cette section sont les mêmes que ceux utilisés dans le chapitre III.

a)Un agent avec présence de perturbation

Nous considérons dans cette première expérimentation le cas d'évolution d'un seul agent sans la présence d'aucun voisin. Le scénario est le même que celui dans 3.6.1, sauf qu'ici le quadrirotor est soumis à des perturbations externes comme illustré dans les figures 4.8 et 4.11.

FIGURE 4.8 – Un opérateur pousse un UAV et génère une perturbation.

La fonction d'optimisation $\Lambda_i(t)$ implémentée dans le quadrirotor est basée uniquement sur sa propre position et la position de la cible P_d. La fonction objective devient alors :

$$\Lambda(t) = \Big(\|P_d - [x(t) + h(t)]\| - d_p^d \Big) \tag{4.14}$$

Avec $h(t) \in R^2$, et $d_p^d = 0$ la distance désirée entre l'agent et le point cible.

L'objectif se résume alors à trouver le meilleur vecteur $h^*(t)$ minimisant la fonction de coût $\Lambda(t)$ telle que :

$$\forall i \in N : \lim_{t\to\infty} \Lambda(t) = 0 \Rightarrow \lim_{t\to\infty} P_d - x(t) = 0 \tag{4.15}$$

Où $h_i^*(t)$ représente le déplacement désiré du drone entre deux instants t et $t + \tau$. La trajectoire de référence à l'instant $t + \tau$ devient :

$$h_i^*(t) = Y_{id}(t + \tau) - Y_i(t) \tag{4.16}$$

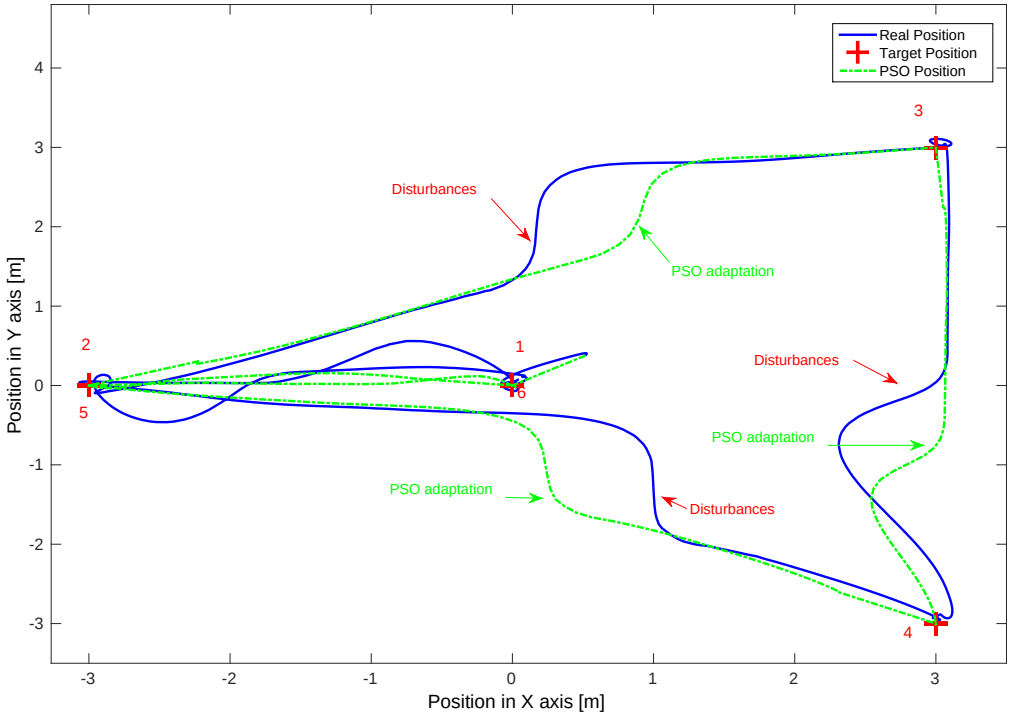

FIGURE 4.9 – Représentation 2D de la trajectoire du drone.

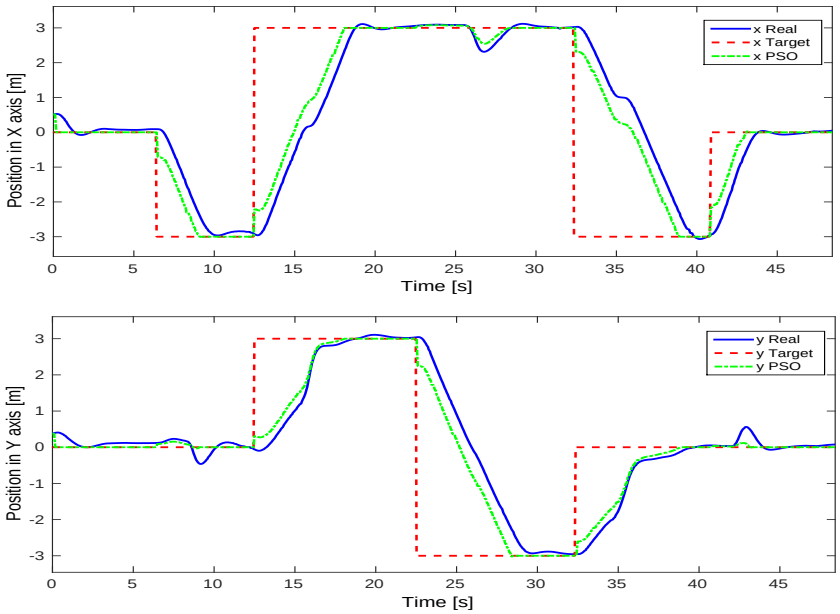

FIGURE 4.10 – Trajectoires références et réelles du drone

Les figures 4.9 et 4.10 permettent de visualiser la trajectoire réelle de l'UAV ainsi que celle générée par le module d'optimisation en fonction des points cibles $1, ..., 5$.

Au lieu de ne compenser que l'erreur engendrée par les perturbations externes, l'algorithme PSO recalcule la trajectoire adéquate pour éliminer au maximum l'action des perturbations, ce qui permet au véhicule d'avoir un comportement plus stable. Nous constatons également que le quadrirotor arrive à chaque fois à atteindre les points cibles et à se stabiliser sans difficulté.

FIGURE 4.11 – Un opérateur modifie la position d'un UAV afin de générer de larges perturbations.

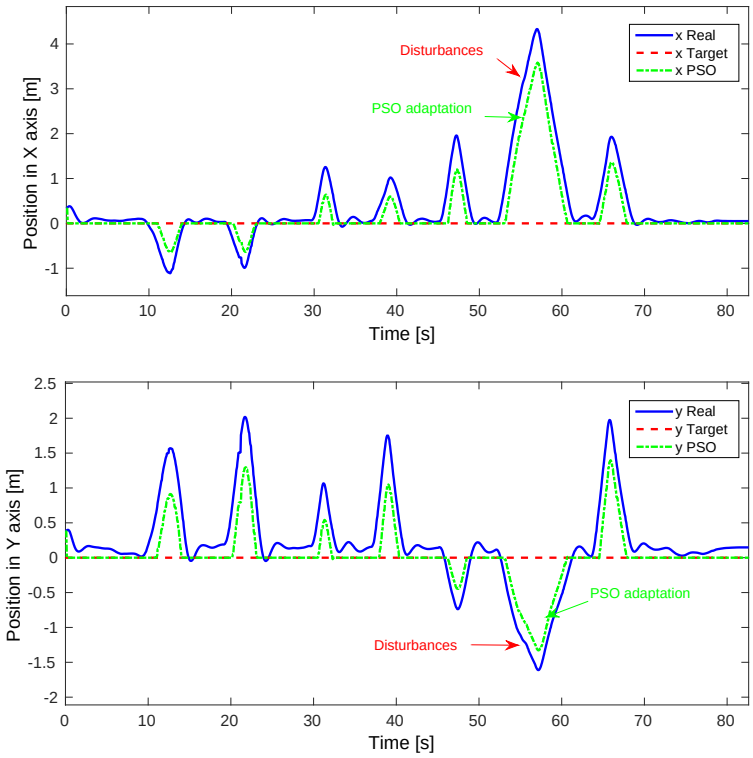

FIGURE 4.12 – Performances du PSO dans la génération de trajectoires.

Dans le même contexte, le quadrirotor est soumis cette fois-ci à de fortes perturbations alors qu'il reste en mode vol stationnaire stabilisé sur un des points cibles. La figure 4.11 montre l'action d'un opérateur qui écarte le drone de sa position d'une seule main.

Comme observé également sur la figure 4.12, l'algorithme PSO recalcule une nouvelle trajectoire pour revenir à la position de la cible, et comme celle-ci reste proche de la position de l'UAV, l'action de la perturbation est minimisée ce qui permet d'avoir un comportement plus lisse du quadrirotor.

Les résultats obtenus montrent un aspect très intéressant de la génération de trajectoire par l'approche PSO. Nous remarquons également la robustesse de cette méthode vis-à-vis des perturbations.

b)Flotte de trois quadrirotors

Dans la deuxième expérimentation, l'approche de commande proposée est implémentée sur une flotte de trois quadrirotors. Chaque UAV calcule sa trajectoire en tenant compte : de la position de ses voisins, des points références et de la configuration désirée des UAVs définie dans la matrice ∂_d. Les éléments $d_{ij}(t)$ sont définis de telle sorte que les distances entre les UAVs soient homogènes, conduisant à une formation triangulaire autour des points cibles. Le schéma général de commande considéré dans le chapitre III est repris dans la figure 4.13

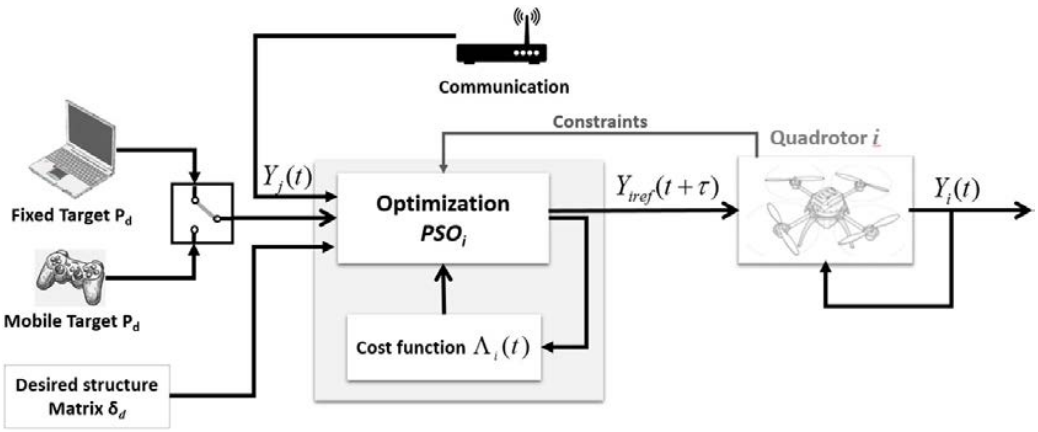

FIGURE 4.13 – Schéma général de commande

Deux types d'objectif de déplacement de la flotte sont considérés dans cette expérience, figure 4.13. Le premier se résume en une succession de points cibles à atteindre par les véhicules figure 4.16. Le deuxième type d'objectif est le suivi d'une trajectoire continue qui peut être la position d'un autre agent de la flotte (un leader) ou comme dans notre cas la position d'un agent virtuel (une cible mobile) définie par un opérateur extérieur à l'aide d'une manette figure 4.17.

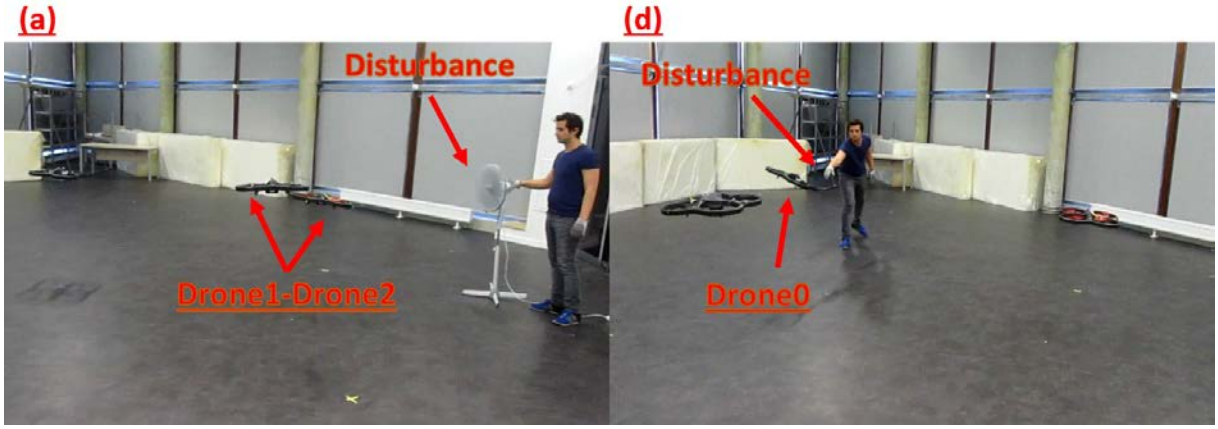

FIGURE 4.14 – Les perturbations introduites lors des expérimentations.

Les quadrirotors sont soumis au cours de leurs déplacements à différents types de perturbations comme illustré dans les figures 4.14. et 4.15. l'objectif, comme dans le cas précédent (déplacement d'un seul agent), est de voir le comportement de la flotte dans le cas de présence de perturbations afin d'évaluer la robustesse de l'approche de commande en formation proposée.

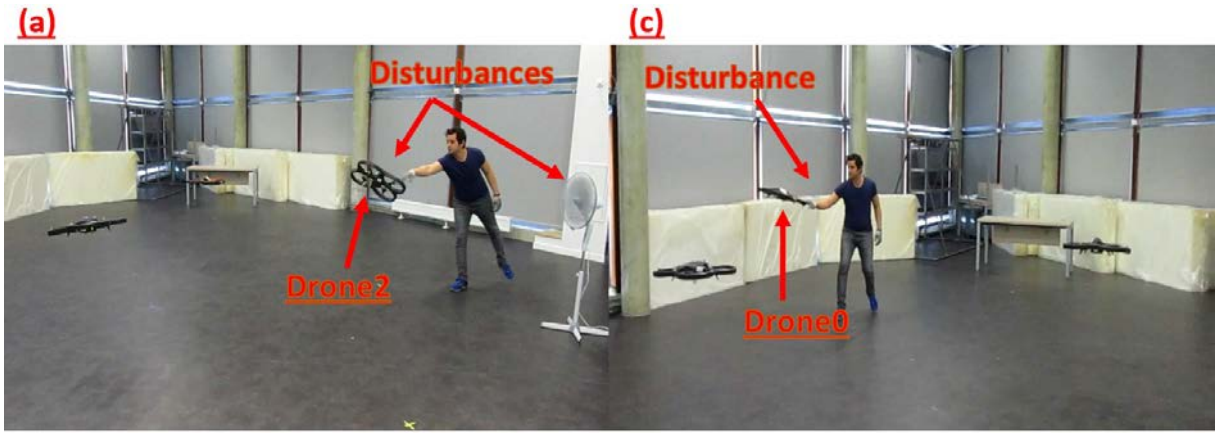

FIGURE 4.15 – Un membre de l'équipe introduit des perturbations sur un véhicule de la flotte.

Les figures 4.16 et 4.17 permettent de décrire l'évolution des trajectoires références et réelles des trois UAVs au cours du temps pour les deux types d'objectifs (**cibles fixes et mobiles**). Nous constatons que les trajectoires calculées par les algorithmes PSO sont stables dans les deux modes de commande de formation et que les perturbations sont clairement atténuées. Les qudrirotors arrivent à suivre la position de la cible en se déplaçant en formation.

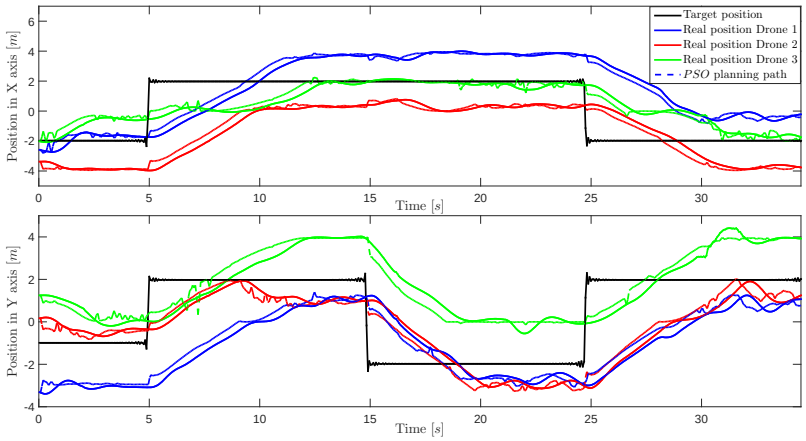

FIGURE 4.16 – Trajectoires réelles et calculées de tous les UAVs. cible fixe.

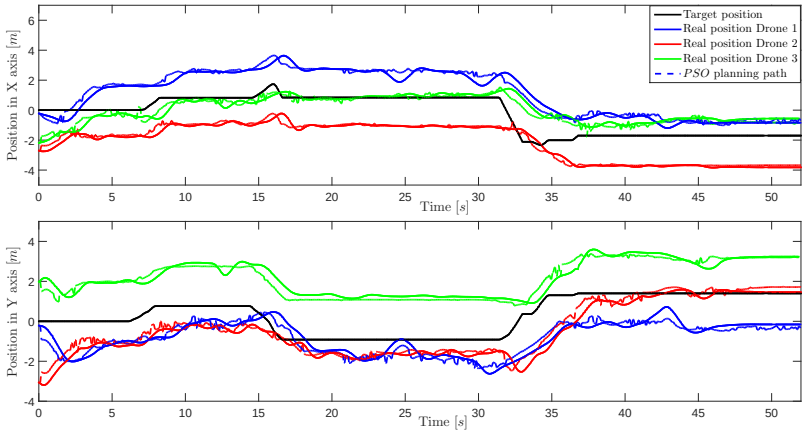

FIGURE 4.17 – Trajectoires réelles et calculées de tous les UAVs. cible mobile.

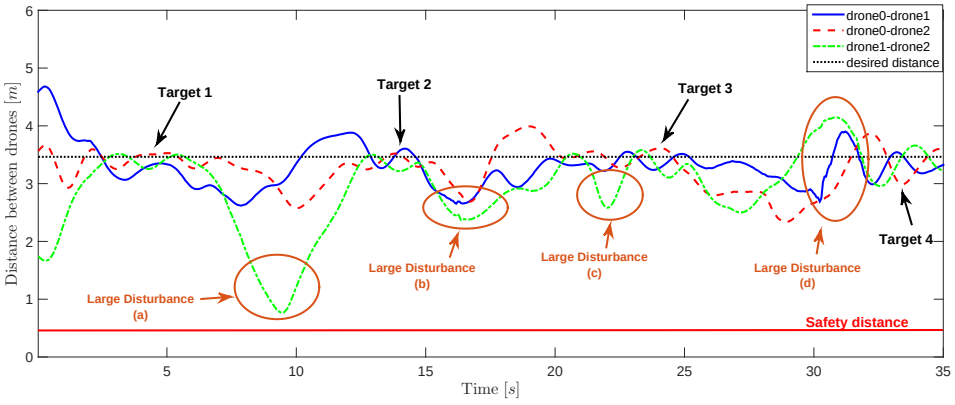

FIGURE 4.18 – Distances entre agents en temps réel dans le cas de présence d'importantes perturbations

Les trajectoires calculées par les différents modules d'optimisation permettent aux UAVs de garder un comportement stable sans dépassement même dans le cas de présence d'erreurs importantes et arrivent à chaque fois à converger vers les consignes désirées, comme constaté sur les figures 4.18 , 4.19 et 4.20 .

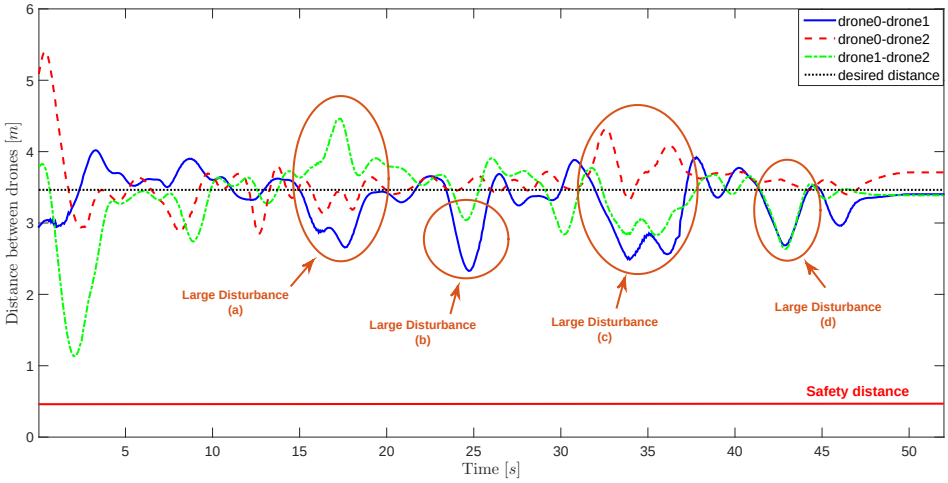

FIGURE 4.19 – Distances entre agents en temps réel en présence d'importantes perturbations. cible mobile.

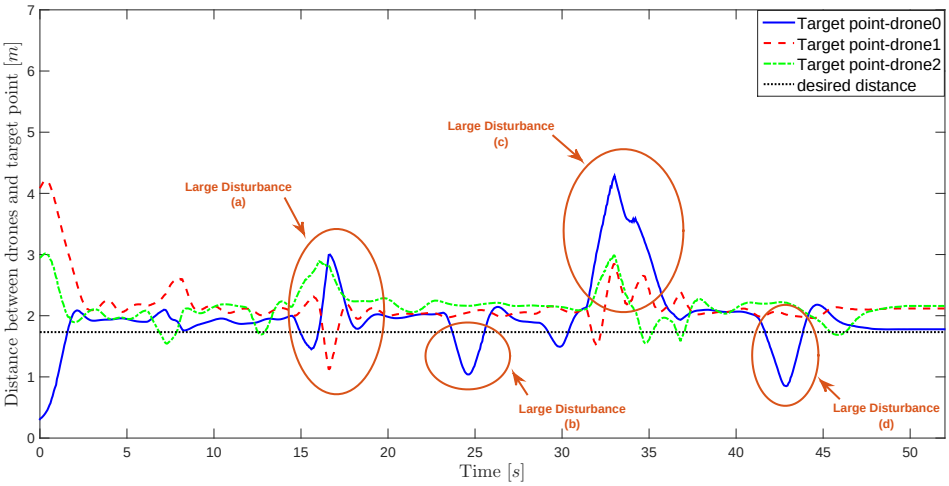

FIGURE 4.20 – Distances entre agents et la trajectoire référence en temps réel en présence de perturbations

c)Flotte de trois quadrirotors et présence de défauts

Nous proposons dans cette dernière partie l'implémentation d'un module de re-configuration des fonctions objectives présentes dans les modules d'optimisation afin de gérer un certain

nombre de défauts qui peuvent être détectés dans la flotte 4.21.

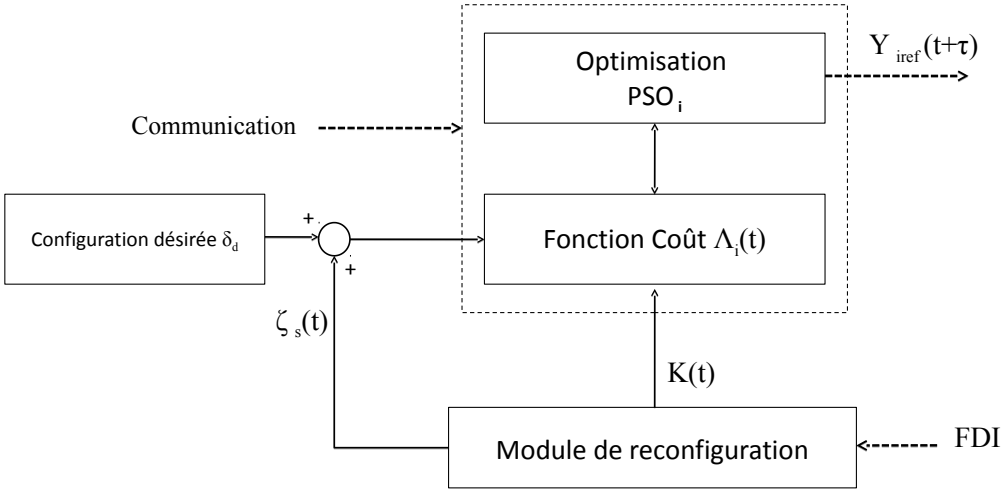

FIGURE 4.21 – Schéma de re-configuration des modules d'optimisation

Les modules de re-configuration à l'aide de modules FDI supposés fonctionnels, permettent d'agir sur les fonctions objectives de chacun des algorithmes PSO afin de modifier, en fonction des défauts détectés, les trajectoires calculées par les modules d'optimisation.

La situation considérée dans ce qui suit est le cas de perte d'un agent dans la flotte. Soit $K(t)$ une matrice diagonale $N \times N$ d'éléments $k_i(t)$, où :

1. $k_i(t) = 1$ Dans le cas de non présence de défaut sur l'agent i
2. $k_i(t) = 0$ Dans le cas où un agent i est défaillant

En introduisant les modifications sur la fonction coût $\Lambda_i(t)$ on trouve :

$$\begin{aligned} \Lambda_i(t) &= \rho\Big(\|P_d - [x_i(t) + h(t)]\| - d_{ip}^d\Big) \\ &+ \sum_{j=1}^{q} k_j(t) \times a_{ij}(t)\Big(\|x_j(t) - [x_i(t) + h(t)]\| - d_{ij}^d)\Big) \end{aligned} \tag{4.17}$$

Une valeur zéro d'un paramètre $k_j(t)$ permettrait alors d'annuler l'influence d'un agent j sur la génération de trajectoire des autres agents de la flotte. La perte d'un agent serait alors prise en compte au niveau des fonctions objectives des modules d'optimisation ce qui facilite grandement le contrôle de formation.

La figure 4.22 montre le cas d'une expérimentation où un agent de la flotte fait un atterrissage d'urgence suite à une défaillance de batterie. Les modules FDI des deux autres agents détectent que celui-ci est défaillant et transmet cette information au module de

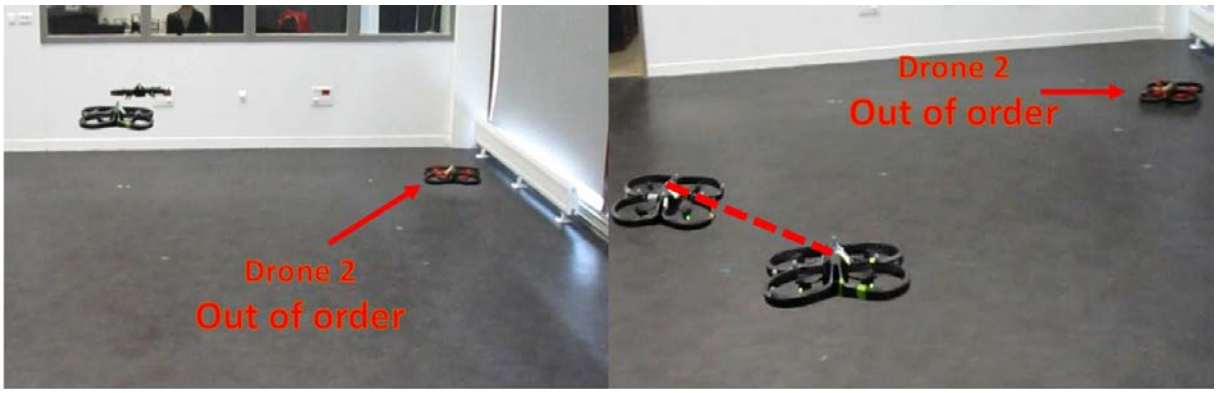

FIGURE 4.22 – Détection d'une défaillance sur un agent de la flotte

re-configuration. Ce module va agir sur la fonction objective en éliminant l'influence de l'agent défectueux à travers la matrice $K(t)$ et modifier la configuration désirée de la flotte avec la matrice $\xi_s(t)$.

Les figures 4.23 , 4.24 et 4.25 représentent les trajectoires enregistrées sur les trois quadrirotors ainsi que l'évolution des distances entre eux et avec la trajectoire référence.

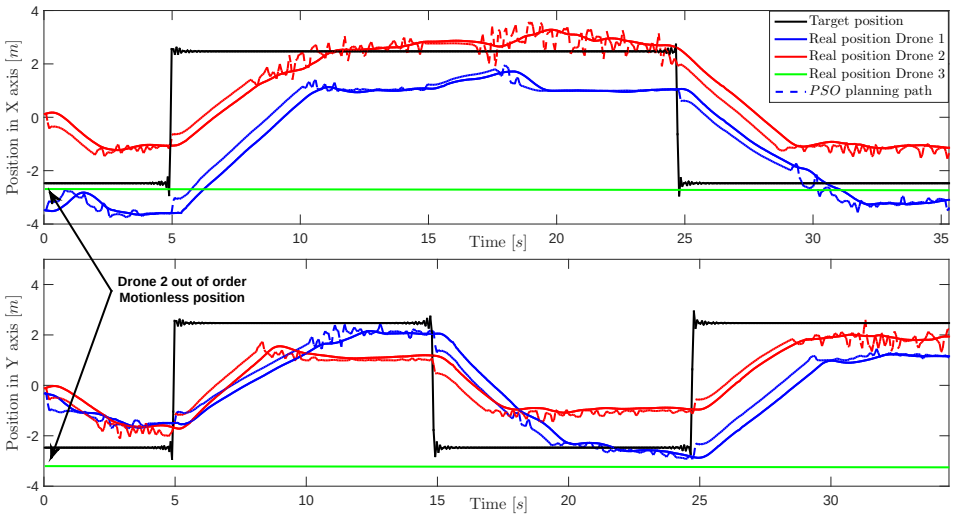

FIGURE 4.23 – Trajectoires réelles et calculées du quadrirotor 1

A partir des figures, nous pouvons constater que le drone 3 est détecté défectueux au départ de la simulation et reste à la même position. Les deux autres drones continuent la mission et se déplace en formation vers les points cibles définis. Nous constatons également que la distance de sécurité est respectée tout au long de la simulation et que la configuration désirée est bien atteinte à chaque point cible.

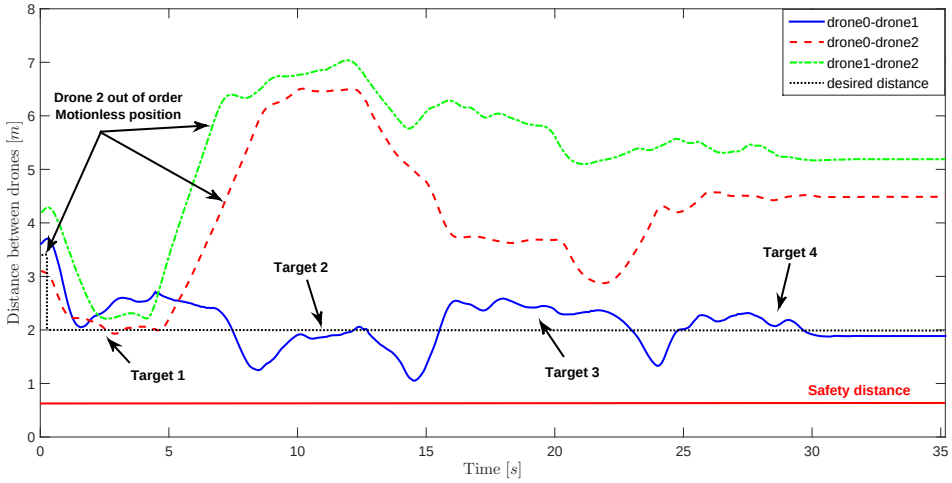

FIGURE 4.24 – Distances entre les UAVs dans le cas d'une défaillance d'un agent.

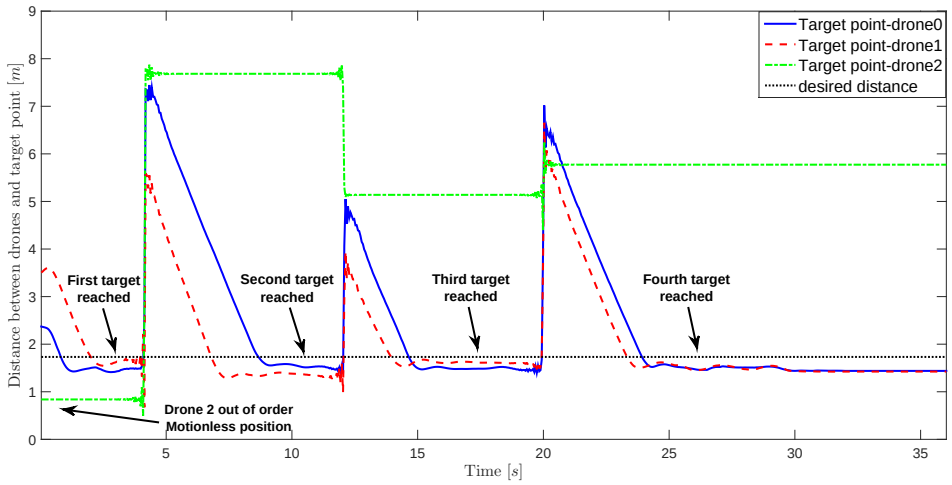

FIGURE 4.25 – Distances entre les UAVs et la trajectoire référence dans le cas d'une défaillance d'un agent.

Toutes les simulations peuvent être retrouvées sur ce lien :

`https://youtu.be/VD3vbGZNhqM`

Nous avons montré à travers les résultats expérimentaux présentés dans cette deuxième partie du chapitre, les performances de l'approche proposée du point de vue de la génération de trajectoire, de la commande de flotte et également de la robustesse vis-àvis des perturbations. L'aspect tolérance aux défauts n'a pas été étudié pour cette approche dans ce chapitre. Néanmoins un premier résultat sur la reconfiguration de la fonction objective dans le cas de perte d'un agent à été présenté.

4.3 Conclusion

Dans ce quatrième chapitre de thèse, des premiers résultats sur la conception de commandes tolérantes aux défauts dans le contexte de la commande de flotte sont présentés. Nous nous sommes intéressés particulièrement sur les cas de perte d'agents dans la flotte et de défauts actionneurs (blocage de la vitesse sur une valeur donnée).

Dans le cas de la commande basée consensus, dès qu'un véhicule est détecté défaillant, il est immédiatement considéré comme un obstacle et sa position n'est plus prise en compte dans la commande de consensus. Dans ce cas, afin de consolider la formation, le rayon de la flotte est diminué par la commande FTC. Dans le cas de détection d'un défaut actionneur, les gains concernant l'agent défectueux sont modifiés et le rayon de la flotte augmenté pour diminuer le risque de collisions. Dans la commande basée sur la génération de trajectoire par PSO, le même procédé est possible, mais le changement s'effectuera au niveau de la fonction objective.

Les extensions proposées dans ce dernier chapitre restent simplistes pour le moment et ne représentent que des résultats préliminaires. Des travaux plus poussés sont à envisager et représentent des perspectives d'amélioration très intéressantes.

Chapitre 5

Conclusion et perspectives

Sommaire

5.1 Conclusion

Cette thèse avait pour objectif principal d'explorer les techniques et les approches de commande de flotte de véhicules autonomes dans le but de concevoir et de réaliser des commandes de vol en formation d'UAVs en temps réel. Nous avions également pour but de prendre en considération le cas de présence de défauts et de défaillances dans la flotte en proposant des commandes tolérantes aux défauts dans le contexte multi-agents.

Nous avons commencé au départ par une recherche bibliographique sur les travaux existants. Tenant compte de la richesse du domaine abordé dans ce sujet, nous avons choisi des directions de recherche spécifiques en essayant de proposer des contributions pertinentes. Nous avons commencé par étudier les approches de commandes de flotte basées sur les algorithmes de consensus. Un grand nombre d'algorithmes sont proposés dans la littéraire et peuvent se différencier soit selon les objectifs de la commande : entre Rendez-vous (consensus en position), flocking (consensus en vitesse), commande de formation (Consensus sur la configuration et distance entre agents), déploiement, soit selon les dynamiques des systèmes commandés : simple intégrateur, double intégrateur, système linéaire général, modèle mono-cycle ou système non-linéaire. D'autres contraintes peuvent également être prises en compte telles que l'évitement de collisions, l'évitement d'obstacles, les contraintes de communication, de perception, de calcul, d'autonomie, de ressources ...etc. En fonction des contraintes considérées et des objectifs de commande, différentes architectures de commande sont développées, centralisée, décentralisée ou distribuée. Nous nous sommes intéressés particulièrement dans cette thèse sur les approches de commande de flotte distribuée. Une extension a été proposée sur un modèle de flocking proposé par Cucker-Smale prenant en compte l'évitement des collisions entre les agents de la flotte. Notre contribution a consisté à implémenter une force de rappel en direction d'un point virtuel pour assurer le déplacement en formation de la flotte. La méthode a été développée sur la commande d'une flotte de véhicules modélisés par des dynamiques en double intégrateurs. Nous avons proposé également une extension pour la commande d'une flotte de quadrirotors en considérant des dynamiques non-linéaires.

Dans un second temps, nous nous sommes intéressés à des commandes de flotte basées sur des techniques d'optimisation. Une approche basée sur une génération de trajectoire distribuée est proposée. Dans celle-ci, un bloc d'optimisation, implémenté sur chaque véhicule, permet en minimisant une fonction objective de calculer la trajectoire référence optimale garantissant : la commande en formation de la flotte, le suivi d'une référence et l'évitement de collisions. La minimisation des fonctions objectives dans chaque véhicule se fait d'une manière totalement distribuée et indépendamment les unes des autres. L'aspect temps réel nécessite également le choix d'un algorithme d'optimisation adapté avec un temps de calcul minimal. L'algorithme PSO (Particle swarm optimization), utilisé dans notre approche, est très adapté pour ce type de problème d'optimisation dynamique. Notre méthode a d'abord été testée en simulation sur Matlab puis sur un simulateur de drones. Des expérimentations ont également été effectuées sur le vol en formation d'une flotte de quadrirotors. L'avantage principal de notre approche est la capacité à dissocier la partie commande locale de chaque véhicule qui peut se faire d'une manière indépendante, avec

la partie génération de trajectoire pour la commande de flotte. Dans la dernière partie de cette thèse nous avons proposé des premiers résultats sur la conception de lois de commande tolérantes aux défauts pour les deux approches de commande de flotte proposées. Nous nous sommes intéressés à des problématiques liées à la perte d'agents dans la flotte, et aux défauts actionneurs. Nous avons pris pour hypothèse dans ces développements qu'un module FDI de détection et d'isolation des défauts quasi-parfait est implémenté en amont à notre commande.

Comme évoqué dans le paragraphes précédent, au cours de ce travail, nous avons testé et validé nos approches de commande de flotte proposées en simulation d'abord sur Matlab puis dans les plateformes de simulation et expérimentales du laboratoire Heudiasyc. Pour les expérimentations, nous avons implémenté nos lois de commande sur des quadrirotors de type ArDrone2, qui évoluent dans un environnement Indoor avec un système Optitrack de localisation de position. Les résultats obtenus ont été concluants et les performances des approches proposées nous semblent intéressantes et des perceptives d'améliorations sont envisagées.

5.2 Perspectives

Dans ce qui suit, nous pressentons quelques perspectives pour la suite des travaux réalisés dans cette thèse :

Dans l'algorithme de flocking proposé, nous avons considéré le cas d'un graphe complet, où chaque agent reçoit les informations sur les positions et vitesses de tous les véhicules de la flotte. La prochaine étape serait de considérer une communication plus restreinte, où chaque agent communique uniquement avec ses voisins les plus proches. Dans ce cas, si le graphe reste connecté, la convergence vers un consensus en vitesse et l'évitement de collisions seront toujours assurés. Pour la commande de formation, le centre virtuel ne sera pas unique mais sera formé par un ensemble de nœuds virtuels. Il serait intéressant d'étudier le comportement de la flotte dans cette configuration.

Dans cette thèse nous avons travaillé sur des véhicules de type Holonome seulement, nous souhaitons développer nos algorithmes pour des modèles non-holonomes en considérant leurs contraintes de déplacement. Cette étude est déjà mise en œuvre en collaboration avec le laboratoire Georgia Institute of Technology. Dans ce projet, une plateforme de robots mobiles non-holonomes "Le Robotarium" est développée sur Matlab [PiGo17]. Notre objectif est d'implémenter nos algorithmes sur cette plateforme pour qu'ensuite ils soient implémentés sur les robots réels. Nous avons commencé également à travailler sur les problèmes d'évitement d'obstacles et nous envisageons d'approfondir nos connaissances sur ces problématiques.

Les problèmes liées aux pertes de communications, pertes de données, retards et erreurs de localisation n'ont pas été considérés dans cette thèse. Il serait intéressant de dévelop-

per les approches proposées dans ce sens pour pouvoir effectuer des vols en formation dans un environnement extérieur. Des premiers résultats sur la conception de commandes tolérantes aux défauts dans le contexte de la commande de flotte sont présentés. Nous nous sommes intéressés uniquement aux cas de perte d'agents et aux défauts actionneurs. L'aspect détection et tolérance aux défauts devront être étudier plus en détails.

Bibliographie

[AlMo11] ALFI, Alireza et MODARES, Hamidreza. System identification and control using adaptive particle swarm optimization. Applied Mathematical Modelling, 2011, vol. 35, no 3, p. 1210-1221.

[AhCh12] AHN, Shin Mi, CHOI, Heesun, HA, Seung-Yeal, et al. On collision-avoiding initial configurations to Cucker-Smale type flocking models. Commun. Math. Sci, 2012, vol. 10, no 2, p. 625-643.

[AlSh08] M. Aldeena, and R. Sharma "Estimation of states, faults and unknown disturbances in non-linear systems", International Journal of Control, Vol. 81, No. 8, pp. 1195-1201, 2008.

[AcIs12] CAI, Kai et ISHII, Hideaki. Average consensus on arbitrary strongly connected digraphs with dynamic topologies. In : American Control Conference (ACC), 2012. IEEE, 2012. p. 14-19.

[AbZh13] Abdolhosseini, M., Zhang, Y.M., Rabbath, C.A. : An efficient model predictive control scheme for an unmanned quadrotor helicopter. J. Intell. Robot. Syst. 70(1–4), 27–38 (2013)

[Ahn12] AHN, Shin Mi, CHOI, Heesun, HA, Seung-Yeal, et al. On collisionavoiding initial configurations to Cucker-Smale type flocking models. Commun. Math. Sci, 2012, vol. 10, no 2, p. 625-643.

[AsWi13] K. J. Astrom, and B. Wittenmark, "Computer-controlled systems : theory and design," Courier Corporation, 2013

[Alekseev13] ALEKSEEV, Vladimir Mikhalovich. Optimal control. Springer Science Business Media, 2013.

[BeTi02] 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.

[Bakul08] Bakule, L. (2008). Decentralized control : An overview. Annual Reviews in Control, 32(1) :87 – 98.

[BlKi03] M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki. "Diagnosis and fault-tolerant control", Springer-Verlag, 2003.

[BoMe03] J.D. Boskovic, and R.K. Mehra "Failure detection, identification and reconfiguration system for a redundant actuator assembly", Proceedings of the 5th IFAC

Symposium on Fault Detection, Supervision and Safety for Technical Processes SAFEPROCESS'03, pp. 429-434, 2003.

[BlFe08] BLIMAN, Pierre-Alexandre et FERRARI-TRECATE, Giancarlo. Average consensus problems in networks of agents with delayed communications. Automatica, 2008, vol. 44, no 8, p. 1985-1995.

[Bai10] BAI, Qinghai. Analysis of particle swarm optimization algorithm. Computer and information science, 2010, vol. 3, no 1, p. 180.

[BuCo09] F. Bullo, J. Cortés, and S. Martinez. Distributed Control of Robotic Networks, Princeton University Press, 2009.

[ChKu07] CHAIMOWICZ, Luiz and KUMAR, Vijay. Aerial shepherds : Coordination among uavs and swarms of robots. In : Distributed Autonomous Robotic Systems 6. Springer Japan, 2007. p. 243-252

[CoMa06] CORTÉS, Jorge, MARTÍNEZ, Sonia, et BULLO, Francesco. Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions. IEEE Transactions on Automatic Control, 2006, vol. 51, no 8, p. 1289-1298.

[CoMo06] CONSOLINI, Luca, MORBIDI, Fabio, PRATTICHIZZO, Domenico, et al. On the control of a leader-follower formation of nonholonomic mobile robots. In : Decision and Control, 2006 45th IEEE Conference on. IEEE, 2006. p. 5992-5997.

[CoMo08] CONSOLINI, Luca, MORBIDI, Fabio, PRATTICHIZZO, Domenico, et al. Leader–follower formation control of nonholonomic mobile robots with input constraints. Automatica, 2008, vol. 44, no 5, p. 1343-1349.

[CaBo13] Eduardo F Camacho and Carlos Bordons. Model predictive control. Springer, 2013.

[ChCa02] CHAIMOWICZ, Luiz, CAMPOS, Mario FM, et KUMAR, Vijay. Dynamic role assignment for cooperative robots. In : Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on. IEEE, 2002. p. 293-298.

[CoMa04] J. Cortés, S. Martinez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks, IEEE Transactions on Robotics and Automation, Vol. 20, No. 2, pp. 243-255, 2004.

[Cao11] CAO, Li, ZHENG, Yufan, et ZHOU, Qing. A necessary and sufficient condition for consensus of continuous-time agents over undirected time-varying networks. Automatic Control, IEEE Transactions on, 2011, vol. 56, no 8, p. 1915-1920.

[CaIs14] CAI, Kai et ISHII, Hideaki. Average consensus on arbitrary strongly connected digraphs with time-varying topologies. IEEE Transactions on Automatic Control, 2014, vol. 59, no 4, p. 1066-1071.

[CuSm07] CUCKER, Felipe et SMALE, Steve. On the mathematics of emergence. Japanese Journal of Mathematics, 2007, vol. 2, no 1, p. 197-227.

[CuDo10] CUCKER, Felipe et DONG, Jiu-Gang. Avoiding collisions in flocks. Automatic Control, IEEE Transactions on, 2010, vol. 55, no 5, p. 1238-1243.

[CaAl13] CAMACHO, Eduardo F. et ALBA, Carlos Bordons. Model predictive control. Springer Science Business Media, 2013.

[CaCh04] CAI, Tao, PAN, Feng, et CHEN, Jie. Adaptive particle swarm optimization algorithm. In : Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on. IEEE, 2004. p. 2245-2247.

[CLERC02] CLERC, Maurice. L'optimisation par essaim particulaire. Technique et Science Informatiques, 2002, vol. 21, no 7, p. 941-964.

[DiKy07] DIMAROGONAS, Dimos V. et KYRIAKOPOULOS, Kostas J. On the rendezvous problem for multiple nonholonomic agents. IEEE Transactions on automatic control, 2007, vol. 52, no 5, p. 916-922.

[DePr02] DEB, Kalyanmoy, PRATAP, Amrit, AGARWAL, Sameer, et al. A fast and elitist multi objective genetic algorithm : NSGA-II. Evolutionary Computation, IEEE Transactions on, 2002, vol. 6, no 2, p. 182-197.

[DoBi08] DORIGO, Marco, BIRATTARI, Mauro, BLUM, Christian, et al. (ed.). Ant Colony Optimization and Swarm Intelligence : 6th International Conference, ANTS 2008, Brussels, Belgium, September 22-24, 2008, Proceedings. Springer, 2008.

[DaMo11] DALMAO, Federico et MORDECKI, Ernesto. Cucker-smale flocking under hierarchical leadership and random interactions. SIAM Journal on Applied Mathematics, 2011, vol. 71, no 4, p. 1307-1316.

[DeIs01] DE PERSIS, Claudio et ISIDORI, Alberto. A geometric approach to nonlinear fault detection and isolation. IEEE transactions on automatic control, 2001, vol. 46, no 6, p. 853-865.

[DuSw16] DU, Ke-Lin et SWAMY, M. N. S. Particle swarm optimization. In : Search and Optimization by Metaheuristics. Springer International Publishing, 2016. p. 153-173.

[Endelbrecht14] ENGELBRECHT, Andries. Particle swarm optimization. In : Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation. ACM, 2014. p. 381-406.

[EbKe95] EBERHART, Russell et KENNEDY, James. A new optimizer using particle swarm theory. In : Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on. IEEE, 1995. p. 39-43.

[FrGr97] FRANKLIN, Stan et GRAESSER, Art. Is it an Agent, or just a Program ? : A Taxonomy for Autonomous Agents. In : International Workshop on Agent Theories, Architectures, and Languages. Springer Berlin Heidelberg, 1996. p. 21-35.

[Ferber99] FERBER, Jacques. Multi-agent systems : an introduction to distributed artificial intelligence. Reading : Addison-Wesley, 1999.

[FaMu02] FAX, J. Alexander et MURRAY, Richard M. Graph laplacians and stabilization of vehicle formations. IFAC Proceedings Volumes, 2002, vol. 35, no 1, p. 55-60.

[FaMu04] FAX, J. Alexander et MURRAY, Richard M. Information flow and cooperative control of vehicle formations. IEEE transactions on automatic control, 2004, vol. 49, no 9, p. 1465-1476.

[Ferber97] FERBER, Jacques. Les systèmes multi-agents : un aperçu général. Techniques et sciences informatiques, 1997, vol. 16, no 8.

[GiMi16] GILLIÉRON, Pierre-Yves et MILOT, Alexandre Manuel. Intégration des véhicules intelligents dans un contexte multimodal. In : 28e Colloque Franco-Suisse ; Route et trafic. 2016.

[HeCo14] HERNANDEZ, Andres, COPOT, Cosmin, CERQUERA, Juan, and al. Formation Control of UGVs using an UAV as Remote Vision Sensor. 2014.

[Hackman11] HACKMAN, J. Richard. Collaborative intelligence : Using teams to solve hard problems. Berrett-Koehler Publishers, 2011.

[HoFa15] 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.

[Henry99] David Henry "Diagnostic et contrôle de cohérence des systèmes multivariables incertains", thèse doctorat, Université de Bordeaux 1, 1999.

[HaLi09] HA, Seung-Yeal, LIU, Jian-Guo, et al. A simple proof of the Cucker-Smale flocking dynamics and mean-field limit. Communications in Mathematical Sciences, 2009, vol. 7, no 2, p. 297-325.

[HeNa16] HENRION, Didier, NALDI, Simone, et EL DIN, Mohab Safey. Exact algorithms for linear matrix inequalities. SIAM Journal on Optimization, 2016, vol. 26, no 4, p. 2512-2539.

[JiWa11] JIA, Yongnan et WANG, Long. Experimental implementation of distributed flocking algorithm for multiple robotic fish. Control Engineering Practice, 2014, vol. 30, p. 1-11.

[JiEg07] M. Ji and M. Egerstedt. Distributed Coordination Control of Multi-Agent Systems While Preserving Connectedness. IEEE Transactions on Robotics, Vol. 23, No. 4, pp. 693-703, Aug. 2007.

[JiLi14] JIANG, Yulian, LIU, Jianchang, TAN, Shubin, et al. Robust consensus algorithm for multi-agent systems with exogenous disturbances under convergence conditions. International Journal of Systems Science, 2014, vol. 45, no 9, p. 1869-1879.

[JiEg07] M. Ji and M. Egerstedt. Distributed Coordination Control of Multi-Agent Systems While Preserving Connectedness. IEEE Transactions on Robotics, Vol. 23, No. 4, pp. 693-703, Aug. 2007.

[JaLI03] A. Jadbabaie, J. Lin, and A.S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules, IEEE Transactions on automatic control, Vol. 48, No. 6, pp. 988-1001, 2003.

[Kha96] KHALIL, Hassan K. Noninear Systems. Prentice-Hall, New Jersey, 1996.

[KiSu07] KIM, Tae-Hyoung et SUGIE, Toshiharu. Cooperative control for target-capturing task based on a cyclic pursuit strategy. Automatica, 2007, vol. 43, no 8, p. 1426-1431.

[KwMa10] KWOK, Andrew et MARTÌNEZ, Sonia. Unicycle coverage control via hybrid modeling. IEEE Transactions on Automatic Control, 2010, vol. 55, no 2, p. 528-532.

[KuMe13] Kushleyev, A., Mellinger, D., Powers, C., and Kumar, V. (2013). Towards a swarm of agile micro quadrotors. Autonomous Robots, 35 :287– 300.

[KaKh08] SEMSAR-KAZEROONI, Elham et KHORASANI, Khashayar. Optimal consensus algorithms for cooperative team of agents subject to partial information. Automatica, 2008, vol. 44, no 11, p. 2766-2777.

[Kodirschek92] Daniel E Koditschek. Task encoding : Toward a scientific paradigm for robot planning and control. Robotics and autonomous systems, 9(1) :539, 1992.

[KaKh09] SEMSAR-KAZEROONI, Elham et KHORASANI, Khashayar. Multiagent team cooperation : A game theory approach. Automatica, 2009, vol. 45, no 10, p. 2205-2213.

[Kennedy10] KENNEDY, James. Particle swarm optimization. In : Encyclopedia of Machine Learning. Springer US, 2010. p. 760-766.

[KaVe02] S. Kanev, and M. Verhaegen "Controller reconfiguration for non-linear systems", Control Engineering Practice, Vol. 8, pp. 1223-1235, 2002.

[Khalil02] H. K. Khalil. Nonlinear Systems. Prentice-Hall, New Jersey, 3rd edition, 2002.

[FiAr88] FILIPPOV, Aleksei Fedorovich et ARSCOTT, Felix Medland (ed.). Differential Equations with Discontinuous Righthand Sides : Control Systems. Springer, 1988.

[KiSh11] KIM, Hongkeun, SHIM, Hyungbo, et SEO, Jin Heon. Output consensus of heterogeneous uncertain linear multi-agent systems. IEEE Transactions on Automatic Control, 2011, vol. 56, no 1, p. 200-206.

[KeKa09] B. Kedjar, and A. H. Kamal, "DSP-based implementation of an LQR with integral action for a three-phase three-wire shunt active power filter," IEEE Transactions on Industrial Electronics, vol. 56, no. 8, pp. 2821-2828, 2009.

[KlGo14] KLEMENT, Nathalie, GOURGAND, Michel, et GRANGEON, Nathalie. Planification de tâches et affectation de ressources : résolution par métaheuristiques inspirées du bin packing et de la PSO. In : ROADEF-15ème congrès annuel de la Société française de recherche opérationnelle et d'aide à la décision. 2014.

[Kennedy11] KENNEDY, James. Particle swarm optimization. In : Encyclopedia of machine learning. Springer US, 2011. p. 760-766.

[LiBr04] LIN, Zhiyun, BROUCKE, Mireille, et FRANCIS, Bruce. Local control strategies for groups of mobile autonomous agents. IEEE Transactions on automatic control, 2004, vol. 49, no 4, p. 622-629.

[LiMa16] LIANG, Kuo-Yun, MÅRTENSSON, Jonas, et JOHANSSON, Karl H. Experiments on platoon formation of heavy trucks in traffic. In : Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on. IEEE, 2016. p. 1813-1819.

[LeDi15] S. Lee, Y. Diaz-Mercado, and M. Egerstedt. Multi-Robot Control Using Time-Varying Density Functions. IEEE Transactions on Robotics, Vol. 31, No. 2, pp. 489-493, Apr. 2015.

[JaLi03] JADBABAIE, Ali, LIN, Jie, et MORSE, A. Stephen. Coordination of groups of mobile autonomous agents using nearest neighbor rules. Automatic Control, IEEE Transactions on, 2003, vol. 48, no 6, p. 988-1001.

[LiYu15] Z. X. Liu, C. Yuan, Y. M. Zhang, and J. Luo, "A learning-based fault tolerant tracking control of an unmanned quadrotor helicopter," Journal of Intelligent Robotic Systems, pp. 1-18, 2015.

[Leip13] LEIPHOLZ, Horst. Stability theory : An introduction to the stability of dynamic systems and rigid bodies. Springer-Verlag, 2013.

[LeDi15] LEE, Sung G., DIAZ-MERCADO, Yancy, et EGERSTEDT, Magnus. Multirobot control using time-varying density functions. IEEE Transactions on Robotics, 2015, vol. 31, no 2, p. 489-493.

[LuCi08] LUO, Ci-yong et CHEN, Min-you. Adaptive particle swarm optimization algorithm [J]. Control and Decision, 2008, vol. 10, p. 010.

[LaYa06] Lai, L.C., Yang, C.C., Wu, C.J. : Time-optimal control of a hovering quad-rotor helicopter. J. Intell. Robot. Syst. 45(2), 115–135 (2006)

[MaNu06] MARQUES, Lino, NUNES, Urbano, et DE ALMEIDA, Anibal T. Particle swarm-based olfactory guided search. Autonomous Robots, 2006, vol. 20, no 3, p. 277-287.

[MoCa09] MOORE, Brandon J. et CANUDAS-DE-WIT, Carlos. Formation control via distributed optimization of alignment error. In : Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on. IEEE, 2009. p. 3075-3080.

[MaBr04] MARSHALL, Joshua A., BROUCKE, Mireille E., et FRANCIS, Bruce A. Formations of vehicles in cyclic pursuit. IEEE Transactions on automatic control, 2004, vol. 49, no 11, p. 1963-1974.

[Murray09] R.M. Murray. Optimization-based control. Technical Report, California Institute of Technology, CA, 2009.

[MiZa08] MICHAEL, Nathan, ZAVLANOS, Michael M., KUMAR, Vijay, et al. Distributed multi-robot task assignment and formation control. In : Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on. IEEE, 2008. p. 128-133.

[MeEg10] MESBAHI, Mehran et EGERSTEDT, Magnus. Graph theoretic methods in multiagent networks. Princeton University Press, 2010.

[Moreau04] MOREAU, Luc. Stability of continuous-time distributed consensus algorithms. In : Decision and Control, 2004. CDC. 43rd IEEE Conference on. IEEE, 2004. p. 3998-4003.

[MoBa17] MOBAYEN, Saleh et BALEANU, Dumitru. Linear matrix inequalities design approach for robust stabilization of uncertain nonlinear systems with perturbation based on optimally-tuned global sliding mode control. Journal of Vibration and Control, 2017, vol. 23, no 8, p. 1285-1295.

[MeKu11] D. Mellinger and V. Kumar, Minimum snap trajectory generation and control for quadrotors, IEEE International Conference on Robotics and Automation, 2011.

[NaVa08] S. Narasimhan, P. Vachhani, and R. Rengaswamy "New nonlinear residual feedback observer for fault diagnosis in nonlinear systems", Automatica, Vol. 44, No. 9, pp. 2222-2229, September 2008.

[NiSt03] H. Niemann, and J. Stoudtrup "Passive fault tolerant control of double inverted pendulum a case study example", Proceedings of the 5th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes SAFEPROCESS'03, Washington, D.C., USA, 2003.

[Saber06] OLFATI-SABER, Reza. Flocking for multi-agent dynamic systems : Algorithms and theory. IEEE Transactions on automatic control, 2006, vol. 51, no 3, p. 401-420.

[OlDu09] S. Olaru, D. Dumur, and S. Dobre. On the geometry of predictive control with nonlinear constraints. Informatics in Control, Automation and Robotics, pages 301–314, 2009.

[OlGr15] Sorin Olaru, Alexandra Grancharova, and Fernando Lobo Pereira. Developments in Model-Based Optimization and Control : Distributed Control and Industrial Applications, volume 464. Springer, 2015.

[OlMu07] R. Olfati-Saber, J.A. Fax, and R.M. Murray. Consensus and cooperation in networked multi-agent systems, Proceedings of the IEEE, Vol. 95, No. 1, pp. 215-233, 2007.

[PiKu08] PIMENTA, Luciano CA, KUMAR, Vijay, MESQUITA, Renato C., et al. Sensing and coverage for a network of heterogeneous robots. In : Decision and Control, 2008. CDC 2008. 47th IEEE Conference on. IEEE, 2008. p. 3947-3952.

[Prodan12] Ionela Prodan. Constrained control of dynamical Multi-Agent systems. PhD thesis, Supelec, 2012.

[Patton94] PATTON, Ron J. Robust model-based fault diagnosis : the state of the art. IFAC Proceedings Volumes, 1994, vol. 27, no 5, p. 1-24.

[Park10] PARK, Jaemann, KIM, H. Jin, et HA, Seung-Yeal. Cucker-Smale flocking with inter-particle bonding forces. Automatic Control, IEEE Transactions on, 2010, vol. 55, no 11, p. 2617-2623.

[PiGo17] D. Pickem, P. Glotfelter, L. Wang, M. Mote, A. Ames, E. Feron, and M. Egerstedt. The Robotarium : A Remotely Accessible Swarm Robotics Research Testbed, IEEE International Conference on Robotics and Automation, 2017.

[RaKu16] RADMANESH, Mohammadreza, KUMAR, Manish, NEMATI, Alireza, et al. Dynamic optimal UAV trajectory planning in the National Airspace System via mixed integer linear programming. Proceedings of the Institution of Mechanical Engineers, Part G : Journal of Aerospace Engineering, 2016, vol. 230, no 9, p. 1668-1682.

[ReMo16] REJEB, Jihene Ben, MORARESCU, Irinel-Constantin, et DAAFOUZ, Jamal. Stratégies événementielles de réinitialisation pour un consensus dans les systèmes multi-agents. Journal Européen des Systèmes Automatisés (JESA), 2016, vol. 49, no 1, p. 93-113.

[REN08] REN, Wei. On consensus algorithms for double-integrator dynamics. IEEE Transactions on Automatic Control, 2008, vol. 53, no 6, p. 1503-1509.

[RiHo02] 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.

[Roth95] HAYES-ROTH, Barbara. An architecture for adaptive intelligent systems. Artificial Intelligence, 1995, vol. 72, no 1-2, p. 329-365.

[Reynolds87] Reynolds, C. W. (1987). Flocks, herds, and schools : A distributed behavioral model. In Computer Graphics, pages 25–34.

[ReBe05] REN, Wei, BEARD, Randal W., et al. Consensus seeking in multiagent systems under dynamically changing interaction topologies. IEEE Transactions on automatic control, 2005, vol. 50, no 5, p. 655-661

[Ripoll99] Patrick Ripoll "Conception d'un système de diagnostic flou appliqué au moteur automobile", thèse doctorat, Université de Savoie, 1999.

[ReBe05] REN, Wei et BEARD, Randal W. Consensus seeking in multiagent systems under dynamically changing interaction topologies. IEEE Transactions on automatic control, 2005, vol. 50, no 5, p. 655-661.

[RaZi16] RAISSOULI, Mustapha, ZINE, Rabie, et EL YASSINI, Khalid. Outils d'aide à la décision pour la planification des réseaux de distribution de l'énergie électrique. REVUE AFRICAINE DE LA RECHERCHE EN INFORMATIQUE ET MATHÉMATIQUES APPLIQUÉES, 2016, vol. 13.

[RoTa13] ROBERGE, Vincent, TARBOUCHI, Mohammed, et LABONTÉ, Gilles. Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning. IEEE Transactions on Industrial Informatics, 2013, vol. 9, no 1, p. 132-141.

[RoTa13] ROBERGE, Vincent, TARBOUCHI, Mohammed, et LABONTÉ, Gilles. Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning. IEEE Transactions on Industrial Informatics, 2013, vol. 9, no 1, p. 132-141.

[ReBe08] ReBe08 REN, Wei et BEARD, Randal W. Distributed consensus in multi-vehicle cooperative control. London : springer, 2008

[ScMc06] SCHWAGER, Mac, MCLURKIN, James, et RUS, Daniela. Distributed Coverage Control with Sensory Feedback for Networked Robots. In : robotics : science and systems. 2006.

[SeDi09] SEURET, Alexandre, DIMAROGONAS, Dimos V., et JOHANSSON, Karl H. Consensus of double integrator multi-agents under communication delay. IFAC Proceedings Volumes, 2009, vol. 42, no 14, p. 376-381.

[SePa08] SEPULCHRE, Rodolphe, PALEY, Derek A., et LEONARD, Naomi Ehrich. Stabilization of planar collective motion with limited communication. IEEE Transactions on Automatic Control, 2008, vol. 53, no 3, p. 706-719.

[Saif15] SAIF, Osamah, FANTONI, Isabelle, et ZAVALA-RÍO, Arturo. Real-time flocking of multiple-quadrotor system of systems. In : System of Systems Engineering Conference (SoSE), 2015 10th. IEEE, 2015. p. 286-291.

[ShWa06] 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.

[SoAu10] 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.

[SaMu04] OLFATI-SABER, Reza et MURRAY, Richard M. Consensus problems in networks of agents with switching topology and time-delays. Automatic Control, IEEE Transactions on, 2004, vol. 49, no 9, p. 1520-1533.

[SaVo16] Saska, M., Vonásek, V., Chudoba, J., Thomas, J., Loianno, G., Kumar, V. (2016). Swarm Distribution and Deployment for Cooperative Surveillance by Micro-Aerial Vehicles. Journal of Intelligent and Robotic Systems, 84(1-4), 469-492.

[SuMo14] Z. Sun, S. Mou, B.D.O. Anderson, and A.S. Morse. Formation movements in minimally rigid formation control with mismatched mutual distances, IEEE Conference on Decision and Control, pp. 6161-6166, 2014.

[StGe00] M. Staroswiecki, and A. Gehin "From control to supervision", Proceedings of the 4rd IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes SAFEPROCESS'2000, Budapest, Hungary, pp. 312-323, 2000.

[SeDi08] SEURET, Alexandre, DIMAROGONAS, Dimos V., et JOHANSSON, Karl H. Consensus under communication delays. In : Decision and Control, 2008. CDC 2008. 47th IEEE Conference on. IEEE, 2008. p. 4922-4927.

[SeDi08] SEURET, Alexandre, DIMAROGONAS, Dimos V., et JOHANSSON, Karl H. Consensus under communication delays. In : Decision and Control, 2008. CDC 2008. 47th IEEE Conference on. IEEE, 2008. p. 4922-4927.

[SaKr14] SASKA, Martin, KRAJNIK, Tomas, VONÁSEK, Vojtch, et al. Fault-tolerant formation driving mechanism designed for heterogeneous MAVs-UGVs groups. Journal of Intelligent Robotic Systems, 2014, vol. 73, no 1-4, p. 603.

[SuMo14] Z. Sun, S. Mou, B.D.O. Anderson, and A.S. Morse. Formation movements in minimally rigid formation control with mismatched mutual distances, IEEE Conference on Decision and Control, pp. 6161-6166, 2014.

[ShSi07] SHELOKAR, P. S., SIARRY, Patrick, JAYARAMAN, Valadi K., et al. Particle swarm and ant colony algorithms hybridized for improved continuous optimization. Applied mathematics and computation, 2007, vol. 188, no 1, p. 129-142.

[TaJa07] Tanner, H., Jadbabaie, A., and Pappas, G. (2007). Flocking in fixed and switching networks. IEEE Transactions on Automatic Control, 52(5) :863–868.

[TaGo14] TABIA, Nourredine, GONDRAN, Alexandre, BAALA, Oumaya, et al. Recherche Tabou Robuste pour l'allocation de fréquences. In : ROADEF 2014, 15ème congrès annuel de la Société française de recherche opérationnelle et d'aide à la décision. 2014.

[Trelea03] TRELEA, Ioan Cristian. The particle swarm optimization algorithm : convergence analysis and parameter selection. Information processing letters, 2003, vol. 85, no 6, p. 317-325.

[TaJa07] H.G. Tanner, A. Jadbabaie, and G.J. Pappas. Flocking in Fixed and Switching Networks, IEEE Transactions on Automatic Control, Vol. 52, No. 5, 2007.

[ViMa11] VIAN, John Lyle, MANSOURI, Ali Reza, et SAAD, Emad William. System and method for inspection of structures and objects by swarm of remote unmanned vehicles. U.S. Patent No 8,060,270, 15 nov. 2011.

[VaVi14] 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.

[Vanaret15] VANARET, Charlie. Hybridation d'algorithmes évolutionnaires et de méthodes d'intervalles pour l'optimisation de problèmes difficiles. 2015. Thèse de doctorat. École Doctorale Mathématiques, Informatique et Télécommunications (Toulouse) ; 142547247.

[WoJe95] WOOLDRIDGE, Michael et JENNINGS, Nicholas R. Intelligent agents : Theory and practice. The knowledge engineering review, 1995, vol. 10, no 02, p. 115-152.

[XiWa07] XIE, Guangming et WANG, Long. Consensus control for a class of networks of dynamic agents. International Journal of Robust and Nonlinear Control, 2007, vol. 17, no 1011, p. 941-959.

[YaKu09] YANG, Kai-Hsiang, KUO, Tai-Liang, LEE, Hahn-Ming, et al. A reviewer recommendation system based on collaborative intelligence. In : Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 01. IEEE Computer Society, 2009. p. 564-567.

[YuWa16] YU, Junzhi, WANG, Chen, et XIE, Guangming. Coordination of multiple robotic fish with applications to underwater robot competition. IEEE Transactions on Industrial Electronics, 2016, vol. 63, no 2, p. 1280-1288.

[ZhLa13] ZHANG, Qin, LAPIERRE, Lionel, et XIANG, Xianbo. Distributed control of coordinated path tracking for networked nonholonomic mobile vehicles. IEEE Transactions on Industrial Informatics, 2013, vol. 9, no 1, p. 472-484.

[Zhou00] K. Zhou "A new controller architecture for high performance, robust and fault tolerant control", Proceedings of the 39th IEEE Conference on Decision and Control, 2000.

[ZhCh13] Y. M. Zhang, A, Chamseddine, C. A. Rabbath, B. W. Gordon, C. Y. Su, S. Rakheja, C. Fulford, J. Apkarian, and P. Gosselin, "Development of advanced FDD and FTC techniques with application to an unmanned quadrotor helicopter testbed," Journal of Franklin Institute, vol. 350, no. 9, pp. 2396-2422, 2013.

[ZhJi01] Y. M. Zhang, and J. Jiang, "Integrated design of reconfigurable fault-tolerant control systems," Journal of Guidance, Control, and Dynamics, vol. 24, no. 1, pp. 133-136, 2001.

[ZhHu13] ZHU, Qingbao, HU, Jun, et HENSCHEN, Larry. A new moving target interception algorithm for mobile robots based on sub-goal forecasting and an improved scout ant algorithm. Applied Soft Computing, 2013, vol. 13, no 1, p. 539-549.

[ZhWa14] ZHOU, Bo, WANG, Wei, et YE, Hao. Cooperative control for consensus of multi-agent systems with actuator faults. Computers Electrical Engineering, 2014, vol. 40, no 7, p. 2154-2166.

[ZhSc14] D. Zhou and M. Schwager. Vector field following for quadrotors using differential flatness, IEEE International Conference on Robotics and Automation, 2014.

Résumé

Ces dernière années, l'émergence des nouvelles technologiques tels que la miniaturisation des composants, les dispositifs de communication sans fils, l'augmentation de la taille de stockage et les capacités de calcul, ont permis la conception de systèmes multi-agents coopératifs de plus en plus complexes. L'un des plus grands axes de recherche dans cette thématique concerne la commande en formation de flottes de véhicules autonomes. Un grand nombre d'applications et de missions, civiles et militaires, telles que l'exploration, la surveillance, et la maintenance, ont alors été développées et réalisées dans des milieux variés (terre, air, eau). Durant l'exécution de ces tâches, les véhicules doivent interagir avec leur environnement et entre eux pour se coordonner. Les outils de communication disponibles disposent souvent d'une portée limitée. La préservation de la connexion au sein du groupe devient alors un des objectifs à satisfaire pour que la tâche puisse être accomplie avec succès. Une des possibilités pour garantir cette contrainte est le déplacement en formation permettant de préserver les distances et la structure géométrique du groupe. Il est toutefois nécessaire de disposer d'outils et de méthodes d'analyse et de commande de ces types de systèmes afin d'exploiter au maximum leurs potentiels. Cette thèse s'inscrit dans cette direction de recherche en présentant une synthèse et une analyse des systèmes dynamiques multi-agents et plus particulièrement la commande en formation de véhicules autonomes. Les lois de commande développée dans la littérature pour la commande en formation permettent d'accomplir un grand nombre de missions avec un niveau de performance élevé. Toutefois, si un défaut/défaillant apparaît dans la formation, ces lois de commandes peuvent s'avérer très limitées, engendrant un comportement instable du système. Le développement de commandes tolérantes aux défauts devient alors primordial pour maintenir les performances de commande en présence de défauts. cette problématique sera traitée dans ce mémoire de thèse et concernera le développement et la conception de commandes en formation tolérantes au défaut dévolu à une flotte de véhicules autonomes suivant différente configuration/structuration.

Mots-clés: Commande de flotte, Flocking, Algorithme de consensus, PSO, Optimisation, Méta-heuristique, Quadrirotors, Commande distribuée.

Abstract

In recent years, the emergence of new technologies such as miniaturization of components, wireless communication devices, increased storage size and computing capabilities have allowed the design of increasingly complex cooperative multiagent systems. One of the main research axes in this topic concerns the formation control of fleets of autonomous vehicles. Many applications and missions, civilian and military, such as exploration, surveillance, and maintenance, were developed and carried out in various environments. During the execution of these tasks, the vehicles must interact with their environment and among themselves to coordinate. The available communication tools are often limited in scope. The preservation of the connection within the group then becomes one of the objectives to be satisfied in order to carry out the task successfully. One of the possibilities to guarantee this constraint is the training displacement, which makes it possible to preserve the distances and the geometrical structure of the group. However, it is necessary to have tools and methods for analyzing and controlling these types of systems in order to make the most of their potential. This thesis is part of this research direction by presenting a synthesis and analysis of multiagent dynamical systems and more particularly the formation control of autonomous vehicles. The control laws developed in the literature for formation control allow to carry out a large number of missions with a high level of performance. However, if a fault/failure occurs in the training, these control laws can be very limited, resulting in unstable system behavior. The development of faulttolerant controls becomes paramount to maintaining control performance in the presence of faults. This problem will be dealt with in more detail in this thesis and will concern the development and design of Faulttolerant controls devolved to a fleet of autonomous vehicles according to different configuration/structuring.

Keywords: Fleet control , Flocking, consensus algorithm, PSO, Optimization, Métaheuristics, Quadrotors, distributed control.