Distance and orientation-based formation control of UAVs and coordination with UGVs

Stevedan Ogochukwu Omodolor

LUND
UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6181
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Printed in Sweden by Tryckeriet i E-huset
Lund 2022

Abstract

Nowadays, research in autonomous drones has increased thanks to the advancement of drone technology. Nevertheless, there are still limitations when performing specific missions due to flight duration, computational load and mission complexity. This thesis investigates ways to solve this problem by taking advantage of multiple UAVs and UGVs. This thesis aims to implement and evaluate strategies for formation and coordination of multiple UAVs and UGVs. Firstly, we present a brief review of state of the art on formation and flocking control, further specifying the advantages and limitations of each approach.

Secondly, we use a behaviour-based approach to obtain multi-UAV formation control. We adapt the algorithm to apply it to a single integrator system model to control the UGVs' formation. We then propose an extension to the original algorithm to consider orientation during formation and a leader-follower strategy to coordinate the interaction between the units using a cluster-based approach.

Finally, we tested our proposed control laws in simulation and in experiments. The simulations were done in Matlab, while the real-implementation experiments were performed using the Crazyflie quadcopters and three-wheeled omniwheel robots.

Acknowledgements

I want to appreciate those who have contributed to this thesis and supported me in one way or the other, without them this project would not have been possible. I would like to thank my supervisors, Anders Robertsson, Björn Olofsson, and Ph.D. student Zheng Jia for their advice and contributions during this thesis. I also want to thank other members of the Department of Automatic Control in Lund for their help, specifically Alexander Pisarevskiy, for his help setting up the hardware used to perform experiments. I would also like to thank the contributors to the open-source libraries used in this project, specifically ROS, Crazyswarm, and Dynamixel (Automatic Control department in Lund). I would also like to thank my family, my three best friends, and my loving partner for their motivation and support. Finally, I thank God, my father, for taking me through this journey and letting me find the opportunity to work with amazing people.

List of Abbreviations

UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
PID	Proportional Integral Derivative
INDI	Incremental Nonlinear Dynamic Inversion
FCM	Flight duration, Computation load and Mission complexity

List of Figures

2.1 Leader mode strategy 22
2.2 Virtual structure formation approach [Rashid and Issa, 2019] 22
2.3 Behaviour-Based formation approach [Rashid and Issa, 2019] 23

3.1 (a) represents an undirected graph where information between every node is bidirectional and (b) represents a directed graph where information sharing is limited [Fionda and Palopoli, 2011] 26
3.2 Interaction range r between agent i and its neighbours 27
3.3 $\beta - agents$, **(a)** wall obstacles **(b)** spherical obstacles [Olfati-Saber, 2006] 31
3.4 Concept of potential minimization using spring tension 32

3.5 The black square is the original formation and orientation, the red square is the desired orientation. The orientation agent is coloured Dandelion. 35
3.6 In the figure, c_i is the cluster, E_{m_k,c_i} is the energy it takes UGV_k to maintain cluster c_i, E_{c_k,c_i} is the energy it takes UGV_k to change to cluster c_i. 36

4.1 Left figure shows the relations between the frames of reference. Right figure shows the axes of the body frames and the rotors. [Greiff, 2017] 39
4.2 Global coordinate, body coordinate, wheel rotation, and wheel linear velocity. Figure on the right is the body rotated ψ in the global frame . 42
4.3 UAV simulation model 43
4.4 X-axis: comparison of input, experimental data, and output from identified model in Equation 4.28 44
4.5 Y-axis: comparison of input, experimental data, and output from identified model in Equation 4.29 45
4.6 UGV simulation model 45

5.1 Crazyflie 47
5.2 Lighthouse positioning system [Taffanel et al., 2021]. 47
5.3 Different loops of cascaded PID controllers [Bitcraze, 1999]. 48
5.4 Three-wheeled omni-wheeled robot 49
5.5 Velocity controller for the three-wheeled omni-wheels robot 50
5.6 Communication architecture 50
5.7 Distributed architecture 51

6.1 Path of the four UAVs for the free-space square formation in (6.1.1.1) 54
6.2 Inter-agent distance of the four UAVs for the free-space square formation in (6.1.1.1) 55
6.3 Absolute error (m) between the navigation goal (position, velocity) and the current centroid in (6.1.1.1) 55
6.4 Path of the four UAVs for the free-space square formation with noise standard deviation $\sigma = 0.04$ in (6.1.1.2) 56
6.5 Inter-agent distance of the four UAVs for the free-space square formation with noise standard deviation $\sigma = 0.04$ in (6.1.1.2) 57
6.6 Path of the four UAVs for the free-space square formation with 0°orientation with respect to the global frame of reference in (6.1.2) . 58
6.7 Inter-agent distance of the four UAVs for the free-space square formation with orientation in (6.1.2) 59
6.8 Absolute error (m) between the navigation goal (position, velocity) and the current centroid in (6.1.2) 59
6.9 Evolution of the orientation in (6.1.2) 60

6.10 Path of the four UAVs for the free-space square formation and flocking in (6.1.3) 61
6.11 Inter-agent distance of the four UAVs for the free-space square formation and flocking in (6.1.3) 62
6.12 Absolute error (m) between the navigation goal (position, velocity) and the current centroid in (6.1.3) 62
6.13 Path of the four UAVs for the free-space square formation and flocking with obstacle avoidance in (6.1.4.1) 64
6.14 Inter-agent distance of the four UAVs for the free-space square formation and flocking with obstacle avoidance in (6.1.4.1) 65
6.15 Absolute error (m) between the navigation goal (position, velocity) and the current centroid in (6.1.4.1) 65
6.16 Path of the four UAVs for the free-space square formation and flocking with obstacle avoidance in (6.1.4.2) 66
6.17 Inter-agent distance of the four UAVs for the free-space square formation and flocking with obstacle avoidance in (6.1.4.2) 67
6.18 Absolute error (m) between the navigation goal (position, velocity) and the current centroid in (6.1.4.2) 67
6.19 Path of the three UAVs, two UGVs and clusters for the free-space UAV and UGV formation and flocking in (6.1.5) 69
6.20 Inter-agent distance of the four UAVs and two UGVs for the Free-Space square formation, flocking and coordination of the multi-agent system, CXDY refers to distance from cluster$_x$ to UGV$_y$ in (6.1.5) 70
6.21 Cluster solutions : sub-index **1A** represents cluster 1 solution A in (6.1.5) 70
6.22 Real experiment: path of the three UAVs for free-space formation in (6.2.1.1) 72
6.23 Real experiment: inter-agent distance of the three UAVs for free-space formation in (6.2.1.1) 73
6.24 Real experiment: absolute error (m) between the navigation goal (position, velocity) and the current centroid of the three UAVs for free-space formation in (6.2.1.1) 73
6.25 Real experiment: path of the three UAVs for free-space formation with external disturbance in (6.2.1.2) 74
6.26 Real experiment result: inter-agent distance of the three UAVs for free-space formation with external disturbance in (6.2.1.2) 75
6.27 Real experiment: Absolute error(m) between the navigation goal(position, velocity) and the current centroid for free-Space square formation with external disturbance in (6.2.1.2) 75
6.28 Real experiment: path of the three UAVs for free-space formation with 0°orientation in (6.2.2) 76
6.29 Real experiment result: inter-agent distance of the three UAVs for free-space formation with orientation control in (6.2.2) 77

6.30 Real experiment: absolute error (m) between the navigation goal (position, velocity) and the current centroid for free-space formation with orientation control in (6.2.2) 77
6.31 Evolution of the orientation in (6.2.2) 78
6.32 Real experiment: path of the three UAVs for free-space formation and flocking in (6.2.3) 79
6.34 Real experiment: absolute error (m) between the navigation goal (position, velocity) and the current centroid for free-space formation and flocking in (6.2.3) 80
6.33 Real experiment result: inter-agent distance of the three UAVs for free-space formation and flocking in (6.2.3) 80
6.35 Real experiment: path of the three UAVs for free-space formation and flocking with obstacle avoidance in (6.2.4.1) 81
6.36 Inter-agent distance of the three UAVs for formation and flocking with static avoidance in (6.2.4.1) 82
6.37 Absolute error (m) between the navigation goal (position, velocity) and the current centroid for formation and flocking with static obstacle avoidance in (6.2.4.1) 82
6.38 Distance of the three UAVs to the current obstacle for formation and flocking with static obstacle avoidance in (6.2.4.1) 83
6.39 Real experiment: Path of the three UAVs for free-Space formation and flocking with dynamic obstacle avoidance second scenario in (6.2.4.2). The discontinuous green line is the position of the obstacle while the continuous one is the position of UAV_3 84
6.40 Inter-agent distance of the three UAVs for formation and flocking with dynamic obstacle avoidance in (6.2.4.2) 85
6.41 Distance of the three UAVs to the current obstacle for formation and flocking with dynamic obstacle avoidance in (6.2.4.2) 85
6.42 Real experiment: path of the three UAVs, cluster and one UGV for formation and flocking in (6.2.5.1) 87
6.43 Inter-agent distance of the three UAVs, cluster and one UGV for formation and flocking in (6.2.5.1) 88
6.44 Real experiment: path of the three UAVs, cluster and two UGVs for formation and flocking in (6.2.5.2) 89
6.45 Inter-agent distance between UAV cluster and two UGVs for formation and flocking in (6.2.5.2) 90

B.1 Path of the three UAVs for the free-space formation 100
B.3 Absolute error (m) between the navigation goal (position, velocity) and the current centroid for the three UAVs 101
B.2 Inter-agent distance of the three UAVs for the free-space formation formation 101
B.4 Path of the three UAVs for the free-space formation with 90°orientation 102

B.5 Inter-agent distance of the three UAVs for the free-space formation . . 102
B.6 Absolute error (m) between the navigation goal (position, velocity) and the current centroid for the three UAVs 103
B.7 Evolution of the orientation for the UAVs 103
B.8 Path of the five UAVs for the free-space formation 104
B.10 Absolute error (m) between the navigation goal (position, velocity) and the current centroid for the five UAVs 105
B.9 Inter-agent distance of the five UAVs for the free-space formation . . 105
B.11 Path of the five UAVs for the free-space formation with 90°orientation in (B.2.2) 106
B.12 Inter-agent distance of the five UAVs for the free-space formation . . 107
B.13 Absolute error (m) between the navigation goal (position, velocity) and the current centroid for the five UAVs 107
B.14 Evolution of the orientation for the five UAVs 108

List of Tables

5.1 Bill of materials 52

6.1 Parameters used to perform the experiments in Section 6.1.1.1 54
6.2 Parameters used to perform the experiments in Section 6.1.4.1 63
6.3 Parameters used to perform the experiments in Section 6.2.1.1 71
6.4 Parameters (UGV) used to perform the experiments in Section 6.2.5 . 86

List of Algorithms

1 Coordination algorithm between UAVs and UGVs 37

List of Symbols

q	position
p	velocity
a	acceleration
$G(q)$	graph
A	adjacency matrix
r	cutoff range
G	global frame
B	body frame
α	agent in formation
β	obstacle agent
γ	navigation agent
$\mathbb{C}_q$	cluster configuration
μ	model noise mean
σ	model noise variance

Contents

List of Abbreviations **7**

List of Figures **7**

List of Tables **11**

List of Symbols **15**

1. Introduction **19**
1.1 Thesis Outline 20

2. State of the Art **21**
2.1 Formation Control 21

3. Theory **26**
3.1 Preliminaries 26
3.2 Flocking Algorithm 28
3.3 Coordination between UAV and UGV 34

4. Modelling **38**
4.1 UAV 38
4.2 UGV 42

5. Implementation **46**
5.1 Simulation 46
5.2 Real-time Experiment 46
5.3 Bill of Materials 52

6. Results **53**
6.1 Simulation 53
6.2 Real-time Experiments 71

7. Conclusion **91**
7.1 Future Work 92

Bibliography **94**

A. Calculation of Position and Velocity of β-agent **98**

B. Simulation Results **99**
B.1 Three UAVs 100
B.2 Five UAVs 104

1

Introduction

Research in unmanned aerial vehicles has gained popularity in recent years thanks to the advancement both from a technology and software standpoint [Nex et al., 2022]. This interest has resulted in cheaper alternatives equipped with advanced capabilities like localization and autonomous navigation. Nevertheless, there are still limitations when performing specific missions. These limitations are generally because of the following problems: flight duration, computation load and mission complexity. For abbreviation, we will refer to this problems as the **FCM** problem. One way to counter this is the use of multiple drones. As a result, we can perform complex missions while satisfying energy and time constraints.

This has motivated researchers to investigate techniques to coordinate and control multiple UAVs, hence the name formation control. In literature, multiple approaches exist to solve the formation problem, but for this project, we impose the **FCM** problem as a constraint in the design process. This constraint will considerably influence the design choices we will use. For example, we opt for a more distributed computation strategy to ensure minimal computational load on a centralized unit and distribute computation on each robot agent.

There are specific long-range missions requiring quick recharging or ground and air surveillance. UGVs can prove beneficial in these scenarios because they can carry bigger weights and have longer battery life than UAVs. For example, the use of UGVs as a mobile charging station is one such. This solution can reduce the number of static rechargeable stations needed in-between missions. There has been considerable research on the formation and coordination of multiple robots, most centered on homogeneous robots. Robots within a formation can either be homogeneous or heterogeneous. Homogeneous robots are those that share the same controller and physical shape while heterogeneous robots are those that that have different shapes and different control architectures and capabilities [Gigliotta, 2018].

Considering the current state of multiple-agent coordination applied to the use case for long-range missions requiring quick recharging or ground and air surveillance, the objective of this thesis is to implement in a robotic system a

distributed control strategy for the formation and coordination of multiple UAVs and UGVs. This raised the following research questions:

- Can a fleet of robots flock while maintaining a predefined formation from its initial position towards a predefined way-point that is part of a path defined by a higher level task-planner, whose design is out of the scope of this project?
- Based on the outcome of the previous research question, given the same conditions, in between which there is a set of static obstacles, whose shapes and locations we know in advance, is it possible to design a control strategy to avoid the obstacles?

1.1 Thesis Outline

We organized the thesis as follows:

In Chapter 2, a brief introduction to the current state of the art on formation control is detailed. First, we introduce the different approaches used in formation control and then explain the different formation architectures from literature.

In Chapter 3, we introduce the theoretical formulation used in the thesis. A brief introduction to graph theory is given, followed by the formation algorithm. We explain the initial control strategy that we use from the literature. In the following sections, we propose an extension of that strategy to include orientation control and a strategy to coordinate UGVs and UAVs.

In Chapter 4, we explain the mathematical equations describing the models of UAVs and a three-wheeled omni-directional UGV. After that, we state the simplified model used during the simulation. In Chapter 5, we explain how we implemented the system in simulation using Matlab and in experiments using ROS as the communication layer. In Chapter 6, we discuss the results obtained after applying the algorithm to different scenarios. Finally, in chapter 7, we state the conclusions and discuss future works.

2

State of the Art

This chapter will review different approaches to formation control. We will explain the different approaches used to achieve formation control from a communication and control point of view.

2.1 Formation Control

Formation control is a division of robotics that studies the coordination and control of multiple robots to achieve a predefined formation while satisfying constraints like obstacle avoidance and mission objectives[Chen and Wang, 2005]. We generally consider two aspects when tackling a formation problem: the control approach and the network architecture.

2.1.1 Formation Control Approaches

Currently, there are three approaches when it comes to formation control: Leader-follower, Behaviour-based and virtual structure[Saif, 2016].

2.1.1.1 Leader-Follower Approach. In a leader-follower approach, we assign one agent the leader role. The leader tracks a predefined path generated from a mission planner, while the other robots try to follow the leader with the desired formation (Figure 2.1) [Rashid and Issa, 2019]. They are two type of leader-follower strategy: the leader mode strategy and the front mode strategy. In the leader mode approach, the follower agent try to maintain a formation directly with the leader. Representation of the communication between the robot agents in a leader mode strategy where each follower agent try to maintain a certain distance from the leader agent and only communicates with the leader can be in Figure 2.1. On the other hand, for the front mode strategy each agent follows the next agent until reaching the leader agent.

Some of the advantages of the leader-follower approach are that it is easy to understand and implement and the stability analysis is straightforward. However, the control strategies have some drawbacks, one of which is the dependency on one

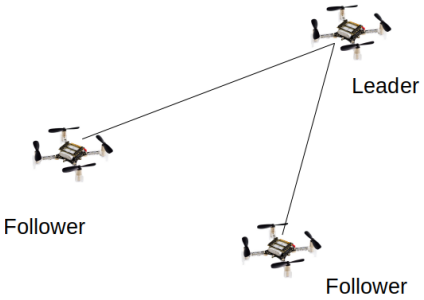

Figure 2.1 Leader mode strategy

robot to maintain the formation. Another drawback is its centralized nature, which makes it challenging to scale.

In current literature, there have been solutions to counter the drawbacks of the leader-follower approach. For example, [Shi et al., 2005] proposed the use of a virtual agent, eliminating the reliance on one agent. [Hou and Fantoni, 2015] proposed the use of a distributed leader-follower formation to counter the centralization problem.

2.1.1.2 Virtual Structure. With a virtual structure, each agent follows the desired trajectory computed from a centralized system to maintain the formation. It forces the agent to act as a rigid formation [Chen and Wang, 2005]. This approach does not consider the interaction between the agents [Saif, 2016].

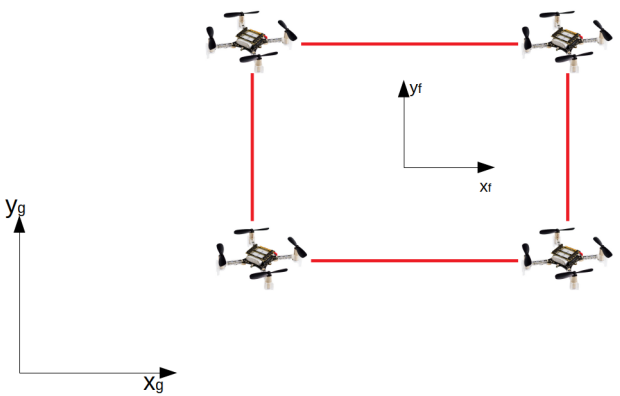

Figure 2.2 Virtual structure formation approach [Rashid and Issa, 2019]

One drawback of this approach, similar to the leader-follower approach, is its centralized nature which could result in high computational and communication

costs. In addition, the lack of interaction between the agents can result in unwanted collisions.

2.1.1.3 Behaviour-based Approaches. In the Behaviour-based approach, each agent has to adhere to some rules (Behaviours) to achieve a formation. The inspiration for these rules comes from the collective Behaviour of the motion of animals. One of the most pioneering works in the distributed Behavioural model is the one done by Reynolds, [Reynolds, 1987]. Although it was in the context of computer graphics, it has inspired many researchers in robotics to come up with rules and control laws to apply this phenomenon in the robotic ecosystem. In [Reynolds, 1987], the author introduced three heuristic rules that each agent has to follow in order to maintain the desired formation:

- Flock centering/formation cohesion: The goal of each agent is to stay as close as possible to nearby agents while also converging to a desired global objective. They achieved this Behaviour by using an attractive/repulsive force to maintain the desired global formation.
- Velocity matching/velocity consensus/velocity alignment: This rule ensures that all agents match the velocity of their nearby neighbours.
- Collision avoidance: This rule ensures collision-free Behaviour with a predefined safety distance with nearby agents. The agent achieves this with adequate repulsive forces when other agents are within the safety distance.

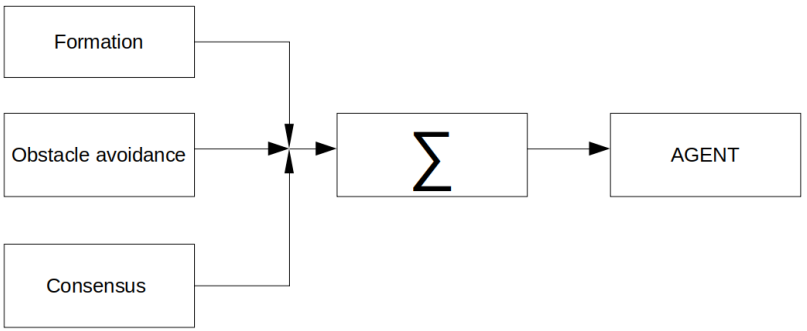

Figure 2.3 Behaviour-Based formation approach [Rashid and Issa, 2019]

One notable work done using approach is by [Olfati-Saber, 2006], where he introduced three flocking algorithms that not only embedded the rules of Reynolds but also included an analytical proof of their stability. The author did this with the use of the Lyapunov stability approach.

The main advantage of this kind of approach is the agents' self-organization nature, scalability and distributed control. The self-organization nature stems from the fact that each agent only has to follow the sets of rules in order to maintain the desired formation. Also, it uses a distributed control approach, which vastly reduces the computation because each agent only has to interact with other agents in its neighbourhood. It makes it easy to scale the formation.

Nevertheless, there are some drawbacks. This approach only ensures the convergence of lattice-type formation with fixed inter-agent distances but does not ensure patterns like V-shapes or rectangular shapes. Additionally, it is challenging to perform a stability analysis of this approach.

2.1.2 Other Approaches

Concerning the sensing capability of each agent, two commonly used formation approaches are displacement-based or distance-based [Sun, 2016].

- **Displacement-based approach**: We achieve formation by controlling the displacement of the neighbouring agents with respect to a global reference system. It requires each agent to have information on the position and displacement of other agents. In this approach, we require the orientation of the global coordinate.
- **Distance-based approach**: The formation is achieved based on the relative inter-agent distances. This approach does not consider the robot's orientation in a global coordinate system, nor does it require a global coordinate system. Compared to displacement-based approach, it is inherently distributed. However, it does require more hardware to achieve inter-agent communication.

Another approach that is gaining popularity is the use of model predictive control combined with a local principle of potential field models in the objective function in order to achieve the formation [Soria et al., 2021].

2.1.3 Formation Control Architectures

There are different control architectures for formation control: centralized, decentralized and distributed [Saif, 2016]. The following paragraphs gives an overview of the advantages and limitations of each architecture.

2.1.3.1 Centralized Architecture. We control all the agents from one centralized controller in this control architecture. The controller receives the states of each agent in the formation as input. Then it computes the necessary signal for each agent and sends it back to them.

One of the advantages of this approach is that global information of all the agents is known. This makes it possible to find optimal path planning and self-organization solutions. Moreover, the dependency on a central computer raises

scalability issues due to the high computational cost and communication power needed. In addition, a fault in the centralized computer will affect all the agents in the formation.

2.1.3.2 Distributed Architecture. In the distributed control architecture, each agent has a controller onboard that computes the control signal based on its current state and the states of the agents in its neighbourhood. Most formation-based controllers use this approach. We can ensure information-sharing through cameras, LIDARs, or peer-to-peer communication. Some works that use this approach are [Olfati-Saber, 2006], [Saif et al., 2019].

One advantage of this architecture is that it is easy to scale because the interaction between agents is limited to its neighbours. In addition, since the system does not depend on a centralized computer, it is reliable. Moreover, due to the lack of global information of all the agents, the solution resulting from the formation controller is in general not optimal.

2.1.3.3 Decentralized Control Architecture. In a decentralized control architecture, the idea is to divide the whole control system into independent subsystems [Bakule, 2008]. Each subsystem includes its controller. The communication between the subsystems is mainly done by a mechanical connection, for example with a physical springs. The controller of each subsystem only has information on the states of the agents in its own subsystem.

Given that the controller of each subsystem is not aware of the agents in other subsystems, we cannot apply this approach to formation control. As stated in [Saif, 2016], most definitions of decentralized formation control conflict with the distributed architecture because the controllers of each agent can access information from neighbour agents.

3

Theory

In this chapter, we state the theoretical formulation used during this thesis. First, we define some preliminaries on graph theory. Then we introduce the initial control strategy used in literature to achieve formation control proposed by [Olfati-Saber, 2006]. Then we explain the extension made by [Saif et al., 2019] to consider effects when implementing this approach in real systems. Finally, we explain the extension, we propose in order to include orientation control and coordination between the UAVs and the UGVs.

3.1 Preliminaries

3.1.1 Graph Theory

In order to represent the interaction within a multi-agent system, we use an undirected graph $\mathbf{G}(\mathscr{V}, \mathscr{E})$. $\mathscr{V}$ represents a set of vertices $\mathscr{V} = \{1, 2, ..., n\}$ and $\mathscr{E}$ a set of edges $\mathscr{E} \subseteq \{(i, j) : i, j \in \mathscr{V}, j \neq i\}$. Each node represents an agent (robot) and the edges are its interaction with other agents. An undirected graph (Figure 3.1) is different from a directed graph in the way the information is exchanged between two nodes. For an undirected graph the information-sharing is bidirectional. A more detailed explanation of graph theory can be found in [Diestel, 2010].

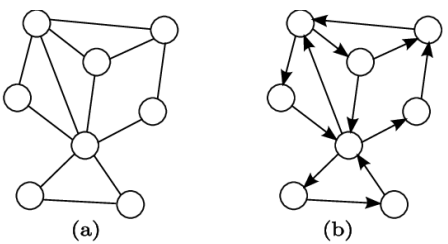

Figure 3.1 (a) represents an undirected graph where information between every node is bidirectional and (b) represents a directed graph where information sharing is limited [Fionda and Palopoli, 2011]

We define a weighted adjacency matrix $\mathscr{A}$ as an M x M matrix with elements $[a_{ij}]$ containing the inter-agent position information. $[a_{ij}]$ is d if $(i,j) \in \mathscr{E}$ and 0 otherwise, where d represents the distance between two nodes (robots).

We define q_i as the position of node i in the Euclidean space. The configuration space of all the nodes can be defined as $q = col(q_1, q_2, ..., q_n)$. We assume r_i to be the maximum range of communication of all agent (Figure 3.2). We used the same r_i for all agent, r. A spatial neighbour [Olfati-Saber, 2006] of agent i can then be defined using the Euclidean distance $\|.\|$ and communication range r as

$$N_i(q) = \{j \in \mathscr{E} : \|q_i - q_j\| < r\} \tag{3.1}$$

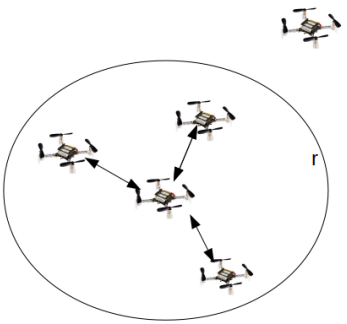

Figure 3.2 Interaction range r between agent i and its neighbours

Given an interaction range, we define a proximity net $\mathbf{G}(\mathscr{V}, \mathscr{E}(q))$ as a structure that consists of vertices $\mathscr{V}$ and a set of edges as

$$\mathscr{E}(q) = \{(i,j) \in \mathscr{V} \times \mathscr{V} : \|q_j - q_i\| < r, i \neq j\} \tag{3.2}$$

With the definition in Equation (3.1), an agent that is part of the formation has to adhere to the following equality:

$$\|q_j - q_i\| = d, \forall j \in N_i(q) \tag{3.3}$$

The proximity net $\mathbf{G}(\mathscr{V}, \mathscr{E}(q))$ that satisfies the condition in Equation (3.3) is defined as "α-*lattice*". The configuration q close to the "α-*lattice*" due to edge-length uncertainty δ is defined as "*quasi* α-*lattice*" and it satisfies the following inequality:

$$-\delta \leq \|q_j - q_i\| - d \leq \delta, \forall j \in N_i(q) \tag{3.4}$$

The degree to which the proximity net deviates from the α-lattice can be defined using the following deviation energy function:

$$E(q) = \frac{1}{(|\mathscr{E}(q)| + 1)} \sum_{i=1}^{n} \sum_{j \in N_i} \psi(\|q_j - q_i\| - d) \tag{3.5}$$

where ψ is the pairwise potential function whose global minimum coincides with the α-lattice. Similarly, in the case of the quasi-α-lattice that includes uncertainty, the deviation energy function can be defined as

$$E(q) \leq \frac{|\varepsilon(q)|}{(|\mathscr{E}(q)|+1)}\delta^2 \leq \delta^2 = \varepsilon^2 d^2 \quad \varepsilon \ll 1 \tag{3.6}$$

3.1.2 Single and Double Integrator Dynamics

There are multiple ways of modeling an agent when performing formation control. In this subsection, we will explain the single and double integrator dynamics.

The single integrator only considers the agent's position and takes as input the velocity:

$$\begin{cases} \dot{q} = p \\ p = \mu \end{cases} \tag{3.7}$$

where μ is the input of the system, q is the position and p is velocity.

On the other hand, the double integrator considers the position and velocity of each agent and takes acceleration as input:

$$\begin{cases} \dot{q} = p \\ \dot{p} = a \\ a = \mu \end{cases} \tag{3.8}$$

where a is the acceleration.

Depending on the dynamics of the agents, it might be preferable to choose either a single or double integrator model. The decision boils down to the dynamic nature of the system and the controller we use to control it. For instance, a single integrator system would work in a system with a slow dynamic that does not require rapid changes in velocity set-point during operation. The UGV used in this project is an example of this system.

3.2 Flocking Algorithm

This section will first discuss the flocking algorithm used in this thesis proposed by [Olfati-Saber, 2006] and extended by [Saif et al., 2019]. It uses a behaviour-based approach. The reason for this choice was that it used a distributed architecture with a control law similar to a PID controller that we could implement in a constrained hardware like the Crazyflie. We discuss this control strategy which was intended to be applied to second-order dynamic systems. Then we adapt that to a first-order dynamic system.

Finally, we propose an extension to the control strategy to include orientation control and a strategy to coordinate between the UAVs and the UGVs.

To better understand formation control, [Olfati-Saber, 2006] defines three different agents, α-agent (the agents in the formation), β-agent (virtual agents that represent the obstacles) and the γ-agent that considers the objective of the entire group.

The concept of σ-norm [Olfati-Saber, 2006] and the smooth adjacency elements are used to define the collective pairwise potential function that was mentioned in Section 3.1.1. The σ-norm maps a vector in $\mathbb{R}^m$ to a non-negative real number $\mathbb{R}^m \rightarrow R_{\geq 0}$ and is defined as

$$\|z\|_\sigma = \frac{1}{\varepsilon}\left[\sqrt{1+\varepsilon\|z\|^2}-1\right] \tag{3.9}$$

where the parameter $\varepsilon > 0$. The gradient $\sigma_\varepsilon(z) = \nabla\|z\|_\sigma$ of the σ-norm that we defined previously is

$$\sigma_\varepsilon(z) = \frac{z}{\sqrt{1+\varepsilon\|z\|^2}} = \frac{z}{1+\varepsilon\|z\|_\sigma} \tag{3.10}$$

The benefit of the σ-norm is that it is differentiable everywhere, including at $z = 0$. This is not the case for the Euclidean norm $\|z\|$, which is not differentiable at $z = 0$. This norm is helpful when defining the potential function for the formation control.

Let the element a_{ij} of the smooth adjacency matrix A be defined as:

$$a_{ij}(q) = ph(\|q_j - q_i\|_\sigma / r_\alpha) \in [0,1] \quad j \neq i \tag{3.11}$$

where $ph(z)$ (bumper function) is a C^1-smooth function that maps a real number to the interval $[0,1]$, "$R^+ \rightarrow [0,1]$" with a finite cut-off at $r_\alpha = \|r\|_\sigma$.

$$ph(z) = \begin{cases} 1, & z \in [0,h) \\ \frac{1}{2} + \cos\left(\pi \frac{(z-h)}{(1-h)}\right), & z \in [h,1] \\ 0, & otherwise \end{cases} \tag{3.12}$$

The use of the bumper function introduces into the proximity net a position-dependent a_{ij} that varies between the interval $[0,1]$ for values within the cutoff range r_α.

The smooth collective potential function used in order to design the formation algorithm can be defined as

$$V(q) = \frac{1}{2}\sum_i \sum_{j \neq i} \psi_\alpha(\|q_j - q_i\|_\sigma) \tag{3.13}$$

where ψ is a smooth pairwise attractive/repulsive potential with a finite cut-off at r_α and a global minimum at $d_\alpha = \|d\|_\sigma$. This potential is differentiable even at singular configurations $q_j = q_i$.

The potential ψ_α can be defined as follows:

$$\psi_\alpha(z) = \int_{d_\alpha}^{z} \phi_\alpha ds \tag{3.14}$$

where ϕ_α is an action function that is zero for $z \geq r_\alpha$

$$\phi_\alpha = ph(z/r_\alpha)\phi(z-d_\alpha) \tag{3.15}$$

$$\phi(z) = \frac{1}{2}[(a+b)\sigma_1(z+c)+(a-c)] \tag{3.16}$$

$$\sigma_1(z) = \frac{z}{\sqrt{1+z^2}} \tag{3.17}$$

where $\phi(z)$ is an uneven sigmoidal function that satisfies the following:

$$0 < a \leq b \tag{3.18}$$

$$\phi(0) = 0 \tag{3.19}$$

$$c = \frac{|a-b|}{\sqrt{4ab}} \tag{3.20}$$

In order to include an obstacle avoidance in the formation, [Olfati-Saber, 2006] proposed the use of an agent-based representation of neighbouring obstacles by introducing the term β-agent. A β-agent, also known as a virtual agent, appears at the obstacle's frontier when an α-agent is close to it. The position $q_{i,k}$ and velocity $p_{i,k}$ of a β-agent is produced by projecting the α-agent to the obstacle, as which can be seen in Figure 3.3. The calculation of the position and velocity of the β-agent is in Appendix A. The repulsive potential function that considers the interaction between an α and β-agent is:

$$\psi_\beta(z) = \int_{d_\beta}^{z} \phi_\beta(s)ds \geq 0 \tag{3.21}$$

where the repulsive action function is,

$$\psi_\beta(z) = ph(z/d_\beta)(\sigma_1(z-d_\beta)-1) \tag{3.22}$$

where $d_\beta = \|d'\|_\sigma$,

The adjacency matrix element of the interaction between these two agents is

$$b_{i,k} = ph(\|q_{i,k}-q_i\|_\sigma/d_\beta) \tag{3.23}$$

The constraint on the agent-to-obstacle for an α-agent is specified as follows:

$$\|\hat{q}_{i,k}-q_i\| = d^{'}, \forall k \in N_i^\beta \tag{3.24}$$

$$N_i^\beta = \{k \in \mathscr{V}_\beta : \|\hat{q}_{i,k}-q_i\| < r^{'}\} \tag{3.25}$$

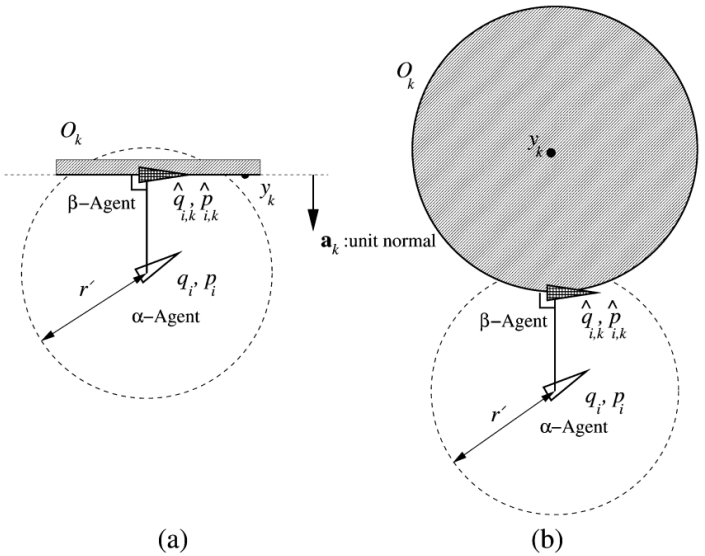

Figure 3.3 $\beta - agents$, **(a)** wall obstacles **(b)** spherical obstacles [Olfati-Saber, 2006]

where r' is the interaction range between the obstacle and the robot.

The collective potential function that includes the three agents, stability of which has been proven in [Olfati-Saber, 2006], is

$$V(q) = c_1^{\alpha} V_{\alpha}(q) + c_1^{\beta} V_{\beta}(q) + c_1^{\gamma} V_{\gamma}(q) \tag{3.26}$$

where

$$V_{\alpha}(q) = \sum_{i \in \mathcal{V}_{\alpha}} \sum_{j \in \mathcal{V}_{\alpha} \setminus \{i\}} \psi_{\alpha}(\|q_j - q_i\|_{\sigma}) \tag{3.27}$$

$$V_{\beta}(q) = \sum_{i \in \mathcal{V}_{\alpha}} \sum_{k \in N_i^{\beta}} \psi_{\beta}(\|\hat{q}_{i,k} - q_i\|_{\sigma}) \tag{3.28}$$

$$V_{\gamma}(q) = \sum_{i \in \mathcal{V}_{\alpha}} (\sqrt{1 + \|q_i - q_r\|^2} - 1) \tag{3.29}$$

Taking into consideration the collective potential functions of the three agents discussed previously, a distributed algorithm for flocking is proposed by [Olfati-Saber, 2006], that includes the formation term u_i^{α}, the obstacle avoidance

term u_i^β, and the navigation term u_i^γ:

$$u_i = u_i^\alpha + u_i^\beta + u_i^\gamma \tag{3.30}$$

$$u_i^\alpha = c_1^\alpha \sum_{j \in N_i^\alpha} \phi_\alpha(\|q_j - q_i\|_\sigma) n_{ij} + c_2^\alpha \sum_{j \in N_i^\alpha} a_{ij}(q)(p_j - p_i) \tag{3.31}$$

$$u_i^\beta = c_1^\beta \sum_{j \in N_i^\beta} \phi_\beta(\|\hat{q}_{i,k} - q_i\|_\sigma) \hat{n}_{i,k} + c_2^\beta \sum_{j \in N_i^\beta} b_{i,k}(q)(\hat{p}_{,k} - p_i) \tag{3.32}$$

$$u_i^\gamma = -c_1^\gamma (q_i - q_r) - c_2^\gamma (p_i - p_r) \tag{3.33}$$

where c_1, c_2 for (α, β, γ) are positive gains, q_r and p_r are the navigation goal, which would come from a higher-level planning system.

The formation and obstacle terms includes two terms

$$u_i = f_i^g + f_i^d \tag{3.34}$$

where on the one hand, $f_i^g = -\nabla V(q)$ is a gradient-based term which drives the formation and obstacle potentials mentioned in Equations 3.27 and 3.28 to their minimum value. This corresponds to desired inter-agent and safety distances. A better way of understanding the concept of potential minimization is to assume a spring is connected to each agent in a formation, see Figure 3.4. Reaching a minimal potential is similar to reaching a point where there is no tension or pressure applied to the springs.

Figure 3.4 Concept of potential minimization using spring tension

On the other hand, the consensus term, f_i^d matches the velocity of agent i with agents in its interaction range and it acts like a damping force. The vectors n_{ij} and $\hat{n}_{i,k}$ are

$$n_{ij} = \frac{q_j - q_i}{\sqrt{1+\varepsilon\|q_j - q_i\|^2}} \qquad \hat{n}_{i,k} = \frac{\hat{q}_{i,k} - q_i}{\sqrt{1+\varepsilon\|\hat{q}_{i,k} - q_i\|^2}} \tag{3.35}$$

One issue with the control law (3.30) that arises during implementation in a real robotic system is perturbations caused by unmodelled dynamics. These perturbations produce a steady-state error in the system that is difficult to eliminate. The reader can find more information on this problem in [Saif et al., 2019]. Based on the proposal by [Saif et al., 2019], we extend the formation control law by including an integrator term to handle the steady-state error:

$$u_i^\alpha = c_1^\alpha \sum_{j\in N_i^\alpha} \phi_\alpha(\|q_j - q_i\|_\sigma) n_{ij} + c_2^\alpha \sum_{j\in N_i^\alpha} a_{ij}(q)(p_j - p_i) + c_3^\alpha \int_0^t e_i^r dt \tag{3.36}$$

where

$$e_i^r = \sum_{j\in N_i^\alpha} \phi_\alpha(\|q_j - q_i\|_\sigma) n_{ij} \tag{3.37}$$

3.2.1 Adaptation to Single Integrator Dynamics

In order to apply this control law in a first-order integrator dynamic system, we adapted the control law stated in Equation (3.30), the result of which is as follows:

$$u_i = u_i^\alpha + u_i^\beta + u_i^\gamma \tag{3.38}$$

$$u_i^\alpha = c_1^\alpha \sum_{j\in N_i^\alpha} \phi_\alpha(\|q_j - q_i\|_\sigma) n_{ij} + c_3^\alpha \int_0^t \sum_{j\in N_i^\alpha} \phi_\alpha(\|q_j - q_i\|_\sigma) n_{ij} dt \tag{3.39}$$

$$u_i^\beta = c_1^\beta \sum_{j\in N_i^\beta} \phi_\beta(\|q_{i,k} - q_i\|_\sigma) \hat{n}_{i,k} \tag{3.40}$$

$$u_i^\gamma = -c_1^\gamma (q_i - q_r) \tag{3.41}$$

One vital observation about the control law presented in Equation (3.38), if we only consider the formation and obstacle avoidance term, is that we do not require information about the global frame, but rather the relative distances between the agents.

3.2.2 Formation Control with Orientation

The majority of the distance-based controllers in the literature are mostly for formation stabilization. [Sun, 2016] proposed the concepts of orientation agents. [Sun, 2016] defined orientation agents as those that have information about the global coordinates and that are used to orientate the whole formation. By defining the relative position between these agents, an additional input can be added to the controller to achieve the desired orientation.

We also define a new concept of orientation agent. The difference compared to [Sun, 2016] is that we only require one of the agents to have information on the orientation of the global coordinate. Furthermore, we propose an orientation-based controller using the concept of rotational potential (rotational spring), see Figure 3.5. The following equation is intended for rotation around the z-axis, but it is extendable to the three axes of rotation:

$$V(\Delta\delta) = \frac{1}{2}(\Delta\delta)^2 \tag{3.42}$$

where $\Delta\delta = \delta_{ref} - \delta_c$, δ is the angle of the vector from the centroid to the orientation agent and $\{.\}_{ref}$ refers to the desired angle while $\{.\}_c$ refers to the current one. A visual illustration of this concept can be seen in Figure 3.5.

Applying the gradient to Equation (3.42), we get the following additional component in the control law:

$$u_i^{\delta} = \nabla V(\Delta\delta) = c_1^{\delta}(\delta_{ref} - \delta_{oa})\hat{r}_z \tag{3.43}$$

where the $\hat{r}_{z,i}$ indicates the direction of the acceleration (green arrow in Figure 3.5) needed for each agent in order for it to rotate with respect to the centroid. One way to understand this concept is to imagine we are applying a centrifugal force to each agent based on the direction and magnitude obtained from the gradient of the orientation potential. We can obtain this vector by performing the following cross multiplication:

$$\hat{r}_z = \overline{f} \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.44}$$

where $\overline{f}$ is the vector from the formation centroid q_c and the orientation agent.

3.3 Coordination between UAV and UGV

In order to develop a coordination strategy, we make the following assumption concerning the use case:

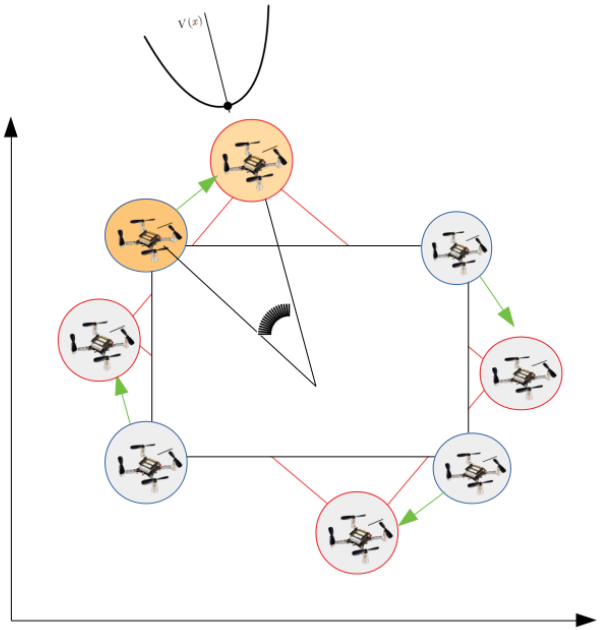

Figure 3.5 The black square is the original formation and orientation, the red square is the desired orientation. The orientation agent is coloured Dandelion.

Assumption 1-$\mathscr{F}$. *This strategy is intended in the case whereby the UGVs act as mobile charging stations or emergency landing stations and try to be as close as possible to as many UAVs that are available.*

With Assumption 1-$\mathscr{F}$, we define the concept of cluster configuration $\mathbb{C}_q$. A cluster configuration consists of N_c number of clusters of UAVs and N_u number of UGVs. We impose the condition that $N_c = N_u$, Figure 3.6.

We initialize $\mathbb{C}_q$ with K-means++ center initialization proposed by [Arthur and Vassilvitskii, 2007], using these as the initial center locations of the UGVs. We decided to use this clustering method because it is not computationally expensive. In addition, due to the initialization strategy, we can find clusters as close as possible to each of the UGVs while preventing intersections between the $UGV_i - c_i$ cluster configuration. To decide when to reconfigure a cluster configuration, we introduce the concept of total energy to maintain or change a configuration. We define the concept of energy as the Euclidean distances. The energy to maintain a cluster

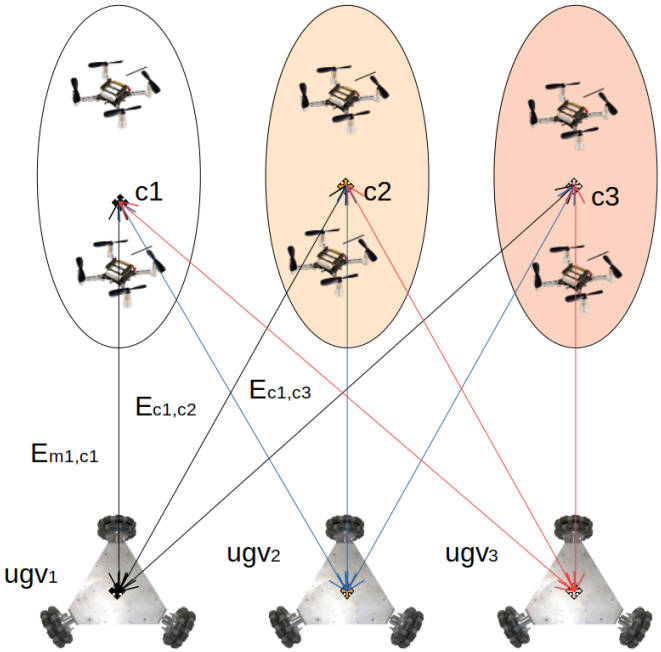

Figure 3.6 In the figure, c_i is the cluster, E_{m_k,c_i} is the energy it takes UGV_k to maintain cluster c_i, E_{c_k,c_i} is the energy it takes UGV_k to change to cluster c_i.

configuration can be computed as:

$$E_{m,\mathbb{C}_q} = \sum_{n \in N_u} E_{m,n,k} \tag{3.45}$$

$$E_{m,n,k} = \|c_k - q_{UGV,n}\| \tag{3.46}$$

where $E_{m,i,k}$ is the energy of the current configuration at the current time instant. It is considered to be the energy it takes UGV_i to maintain its current cluster c_k. On other hand, the energy to change a configuration is defined as:

$$E_{c,\mathbb{C}_q} = \sum_{n \in N_u} E_{m,n,k} \tag{3.47}$$

$$E_{c,n,p} = \min(\|c_p - q_{UGV,n}\|), \qquad p \in 1...N_c, k \neq p \tag{3.48}$$

The energy to change is the energy of a cluster configuration c_k close to the current one with the minimal overall energy compared to other possible clusters configurations. We modify the $\mathbb{C}_q$ when the total energy to change the configuration

is less than the energy it takes to maintain the cluster. The following Algorithm 1 is a pseudo-code of the algorithm.

Algorithm 1 Coordination algorithm between UAVs and UGVs

0: $\mathbb{C}_{q,c} \leftarrow computeCluster(UGV, UAV)$ {Current configuration}
0: $\mathbb{C}_{q,a}$ {Alternative configuration}
0: **while** { }**do** {Main control loop}
0: $\mathbb{C}_{q,a} \leftarrow computeCluster(UGV, UAV)$
0: $E_m \leftarrow computeEm(UGV, C_{q,a}, C_{q,c})$
0: $E_c \leftarrow computeEc(UGV, C_{q,a}, C_{q,c})$
0: **if** $E_c \leq E_m$ **then**
0: $\mathbb{C}_{q,c} \leftarrow \mathbb{C}_{q,a}$
0: **end if**
0: **end while**=0

After obtaining the new $\mathbb{C}_q$, we then apply a two robot formation between each cluster and its corresponding UGVs.

4

Modelling

This section details the mathematical models used to describe the Crazyflie UAV and the three-wheeled omni-wheels robot. After that, we explain the simplified model used to perform simulation and testing of the proposed formation and control algorithms.

4.1 UAV

4.1.1 Rigid-body Dynamics

The models developed in this section are based on [Greiff, 2017] and [Luukkonen, 2011]. In order to derive the systems equations, we will use the Euler-Lagrange equations based on the Tait-Bryan rotation convention ZYX [Greiff, 2017]. In addition, for comparison, the Newton-Euler approach is used. We define three frames of reference; the global frame **G**, inertial frame **I**, and the body frame **B**. The notation, for example, $_{GB}$, describes the rotation of the body frame to the global frame. The notation $\hat{}$ represents the basis vectors in a given frame. Figure 4.1 gives a visual representation of the relationship between these frames of reference. To maintain a consistent reference, we denote P [m] the position of the center of mass of the UAV in a global frame. We denote η [rad] as the Tait-Bryan angle body rotation in the global coordinate and ω [rad/s] denotes the angular velocity of the body frame and V_B is the linear velocities in the body frame. Finally, we denote with Ω_i [rad/s] the angular speed of rotor i. The s vector contains the linear and angular positions of the quadcopter. The vector representation is defined as follows:

$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \eta = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}, \omega = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}, V_B = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}, \Omega = \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \\ \Omega_4 \end{bmatrix} S = \begin{bmatrix} P \\ \eta \end{bmatrix} \tag{4.1}$$

It is worth mentioning that rotational angles in the Euler convention are roll [ϕ], pitch [θ] and yaw [ψ]. The inertial frame is at the position P at the body center of

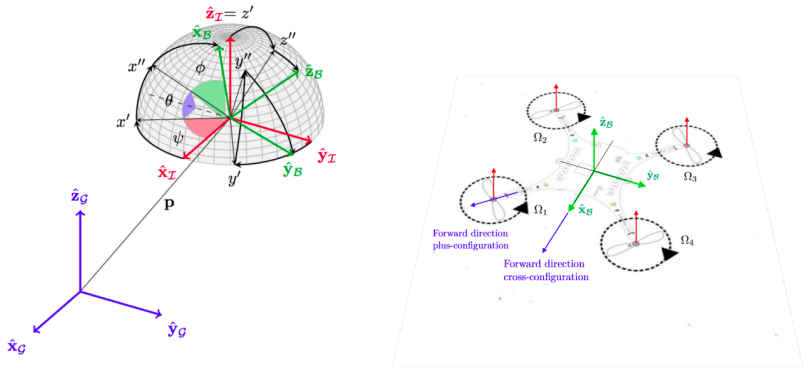

Figure 4.1 Left figure shows the relations between the frames of reference. Right figure shows the axes of the body frames and the rotors. [Greiff, 2017]

mass, while the body frame is positioned at the center of mass but rotated in $SO(3)$ to the inertial frame. We obtain the rotation of the body to the global frame R_{GB} by combining the rotation of the body frame to the inertial frame, which according to the ZYX convention, is given as follows:

$$\mathbf{R}(\psi) = \begin{bmatrix} c_\psi & s_\psi & 0 \\ -s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{R}(\phi) = \begin{bmatrix} c_\theta & 0 & -s_\theta \\ 0 & 1 & 0 \\ s_\theta & 0 & c_\theta \end{bmatrix}, \mathbf{R}(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & s_\phi \\ 0 & -s_\phi & c_\phi \end{bmatrix} \tag{4.2}$$

c_i and s_i denotes $\cos(i)$ and $\sin(i)$. Combining the 3 previous matrixes we stated previously, we get final rotation matrix:

$$R_{GB} = \mathbf{R}(\phi)\mathbf{R}(\theta)\mathbf{R}(\psi) = \begin{bmatrix} c_\phi c_\phi & s_\psi c_\phi & -s_\theta \\ c_\psi s_\theta s_\phi - s_\psi c_\phi & s_\psi s_\theta s_\phi - c_\phi c_\psi & c_\theta s_\phi \\ c_\psi s_\theta c_\phi + s_\phi s_\psi & s_\psi s_\theta c_\phi - c_\psi s_\phi & c_\phi c_\theta \end{bmatrix} \tag{4.3}$$

Based on the Euler's rotational theorem, as stated in [Greiff, 2017], the following equality holds

$$R_{GB}^{-1} = R_{GB}^T = R_{BG} \tag{4.4}$$

The angular rate vector of the quadcopter in the body frame as stated in [Greiff, 2017] is defined as

$$\omega_B = W_\eta \dot{\eta} = \begin{bmatrix} 1 & 0 & -s_\theta \\ 0 & c_\phi & c_\theta s_\phi \\ 0 & -s_\phi & c_\theta s_\phi \end{bmatrix} \dot{\eta} \tag{4.5}$$

where W_η is invertible if $\theta \neq (2k-1)\phi/2, (k \in \mathbb{Z})$. Given the symmetric nature of the quadcopter, we define the inertia matrix as

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{4.6}$$

The force generated by rotor i with rotor speed Ω_i [rad/s] and in the direction of the rotor axis can be obtained as follows [Luukkonen, 2011],

$$f_i = k_i \Omega_i^2 \tag{4.7}$$

The torque around the rotor axis is related to the angular velocity and acceleration of the rotor [Luukkonen, 2011] as

$$\tau_i = b_i \Omega_i^2 + I_M \dot{\Omega}_i \tag{4.8}$$

For the Equation 4.8, k is the lift constant, b is the drag constant and I_M is the inertial moment of the rotor. The total forces generated by each rotor create a torque T in the z-direction of the quadcopter, while the total body torque τ_B is the combination of the torque generated in each of the axes of the corresponding body frame

$$\tau = \sum_{i=1}^{4} f_i = k \sum_{i=1}^{4} \omega_i^2, \tau_B = \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} \tag{4.9}$$

$$\tau_B = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} lk(-\omega_2 + \omega_4) \\ lk(-\omega_2 + \omega_4) \\ \sum_{i=1}^{4} \tau_{M_i} \end{bmatrix} \tag{4.10}$$

where l is the distance from the rotor to the center of mass of the quadcopter.

4.1.1.1 Newton-Euler Equations. We can also describe the dynamics of the quadcopter using the Newton-Euler equations [Luukkonen, 2011]. In order to do so, the quadcopter is assumed to be a rigid body:

$$m\dot{V}_B + \omega \times (mV_B) = R_{GB}^T G + T \tag{4.11}$$

where $R_{GB}^T G$ is the gravitational term and T is the total thrust. In the inertial frame, the centrifugal force is null [Luukkonen, 2011], simplifying the previous equation to

$$m\ddot{P} = G + R_{GB} T \tag{4.12}$$

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = -g \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \frac{T}{m} \begin{bmatrix} c_\psi s_\theta c_\phi + s_\psi c_\phi \\ s_\psi s_\theta c_\phi + c_\psi s_\phi \\ c_\theta c_\phi \end{bmatrix} \tag{4.13}$$

The angular velocity can be obtained through the relation between the angular acceleration of the inertia $I\dot{\omega}$, the centripetal forces $\omega \times (I\omega)$, the gyroscopic forces Γ and the external torque, all expressed in the body frame [Luukkonen, 2011]

$$I\dot{\omega} + \omega \times (I\omega) + \Gamma = \tau \begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} = \begin{bmatrix} (I_{xx} - I_{zz})\omega_y\omega_x/I_{xx} \\ (I_{xx} - I_{zz})\omega_x\omega_z/I_{yy} \\ (I_{xx} - I_{zz})\omega_x\omega_z/I_{zz} \end{bmatrix} - I_r \begin{bmatrix} \omega_y/I_{xx} \\ -\omega_x/I_{yy} \\ 0 \end{bmatrix} \Omega_\gamma + \begin{bmatrix} \tau_\phi/I_{xx} \\ \tau_\theta/I_{yy} \\ \tau_\psi/I_{zz} \end{bmatrix} \tag{4.14}$$

where $\Omega_\gamma = \Omega_1 - \Omega_2 + \Omega_3 - \Omega_4$. The angular acceleration can be obtained by time differentiating the transformation matrix W_η^{-1} in the inertial frame:

$$\ddot{\eta} = \frac{d}{dt}(W_\eta^{-1}\omega) = \frac{d}{dt}(W_\eta^{-1})\omega + W_\eta^{-1}\dot{\omega} = \begin{bmatrix} 0 & \dot{\phi}C_\phi T_\phi + \dot{\theta}s_\phi/c_\theta^2 & -\dot{\phi}s_\phi c_\theta + \dot{\theta}c_\phi/c_\theta^2 \\ 0 & -\dot{\phi}s_\phi & -\dot{\phi}c_\phi \\ 0 & \dot{\phi}/c_\theta + \dot{\phi}s_\phi T_\theta/c_\theta & -\dot{\phi}s_\phi c_\theta + \dot{\theta}c_\phi T_\theta/c_\theta \end{bmatrix} \omega + W_\eta^{-1}\dot{\omega} \tag{4.15}$$

4.1.1.2 Euler-Lagrange Equations. To obtain the quadcopter dynamic equations, the Euler-Langrage approach uses a conservation of energy approach, which can be written in terms of translational, rotational and potential energy [Greiff, 2017]

$$\mathscr{L}(s,\dot{s}) = E_{trans} + E_{rot} - E_{pot} = \frac{1}{2}m\dot{P}^T\dot{P} + \frac{1}{2}\omega^T I\omega - mgz \tag{4.16}$$

This results in the following Euler-Lagrangian equation:

$$\begin{bmatrix} f \\ \tau \end{bmatrix} = \frac{d}{dt}\left(\frac{\partial \mathscr{L}}{\partial \dot{s}}\right) - \frac{\partial \mathscr{L}}{\partial s} \tag{4.17}$$

The angular and linear velocities are independent and can be treated separately. This results in the following linear external force, which is similar to the Euler-Newton relations in Equation (4.12):

$$f = R_{GB}T = m\ddot{P} + mg \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{4.18}$$

The rotational energy can be rewritten in the inertial frame as

$$E_{rot} = \frac{1}{2}\omega^T I\omega = \frac{1}{2}\dot{\eta}J\dot{\eta} \tag{4.19}$$

where $J(\eta) = W_\eta^T I W_\eta$ is a Jacobian matrix that converts the angular rotation in the body frame to the angular rotation in inertial frame. This result in the following external torque:

$$\tau = \tau_B = J\ddot{\eta} + \frac{d}{dt}(J)\dot{\eta} - \frac{1}{2}\frac{\partial}{\partial \eta}(\dot{\eta}^T J\dot{\eta}) = J\ddot{\eta} + C(\eta,\dot{\eta})\dot{\eta} \tag{4.20}$$

where the $C(\eta, \dot{\eta})$ matrix is the Coriolis term. After isolating the angular acceleration in the inertial frame, we get a result that is similar to the one obtained in Equations (4.14) and (4.15):

$$\ddot{\eta} = J^{-1}(\tau_B - C(\eta, \dot{\eta})\dot{\eta}) \tag{4.21}$$

4.2 UGV

In this section, we describe a kinematic model of a three-wheeled omni-directional robot by only taking into consideration the geometry of the system dynamics. The mathematical equation is based on [Galgamuwa et al., 2015].

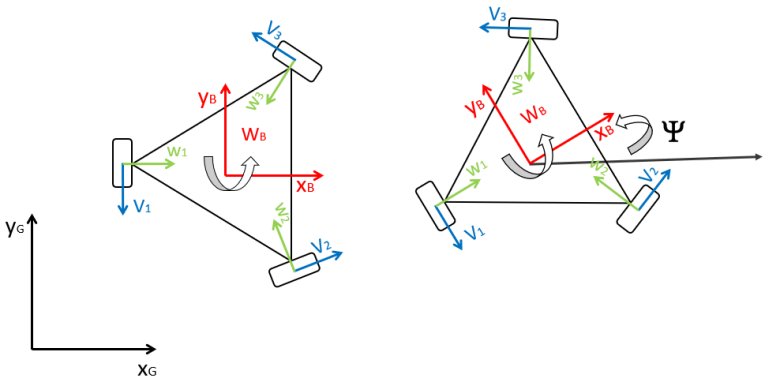

Figure 4.2 Global coordinate, body coordinate, wheel rotation, and wheel linear velocity. Figure on the right is the body rotated ψ in the global frame

The axis naming convention of the global and local frame is similar to ones used in Section 4.1.1. We denote with ψ the orientation of the robot in the global frame, R is the radius of the robot body, r is the radius of the robot wheel, V_B is the velocity in the body frame, ω [rad/s] is the angular velocity of each of the wheels, v_i [m/s] is the velocity at wheel i, and V_G is the velocity in the global frame. The vector representation is defined as follows:

$$\omega = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}, V_B = \begin{bmatrix} v_f \\ v_n \\ \omega \end{bmatrix}, V_G = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix}, \tag{4.22}$$

We can relate the local body velocity and the global velocity with the following equations:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_f \\ v_n \\ \omega \end{bmatrix} \tag{4.23}$$

Using geometry and the axis representation in Figure 4.2, we can relate the velocity of the wheels to the velocity of the body frame. The result of this is:

$$v_1 = -v_n + R\omega \tag{4.24}$$

$$v_2 = v_f \cos\frac{\pi}{6} + v_n \cos\frac{\pi}{3} + R\omega \tag{4.25}$$

$$v_3 = -v_f \cos\frac{\pi}{6} + v_n \cos\frac{\pi}{3} + R\omega \tag{4.26}$$

where $v_i = r\omega_i$ for $i = 1, 2, 3$. Simplifying Equation (4.24), we can get the following equations that relate the local body velocity to the wheels' angular rotation:

$$\begin{bmatrix} v_f \\ v_n \\ \omega \end{bmatrix} = \frac{r}{3} \begin{bmatrix} 0 & \sqrt{3} & -\sqrt{3} \\ -2 & 1 & 1 \\ 1/R & 1/R & 1/R \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \tag{4.27}$$

By combining Equations (4.23) and (4.27), we can relate the global velocity to the local wheel rotational angular rates.

4.2.1 Simplified Simulation Model

In order to test the formation control law before performing tests on real hardware, we developed simulation models for both the UAV and the UGV using transfer functions. These models represent both the robots' dynamics and their internal controller. Although these simulation models do not consider the whole dynamics of the robots, they were adequate to analyze the controller. We assumed that the x and y-axes were decoupled for the simulation models.

Figure 4.3 shows the simulation model of the UAV. It consists of a series of integrators that output the position with an acceleration input. We modeled the velocity controller with a second-order transfer function using experimental data obtained from the Crazyflie UAV. It receives as input $\dot{q}_r$ and outputs the current velocity $\dot{q}$ of the robot. We performed the identification of the transfer function using the identification toolbox that Matlab offers [Matlab, 2022].

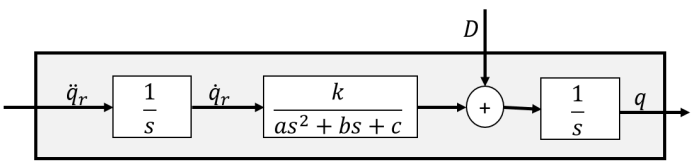

Figure 4.3 UAV simulation model

The transfer functions for the two axes are as follows:

$$tf_x = \frac{33.62}{s^2 + 7.488s + 33.97} \tag{4.28}$$

$$tf_y = \frac{34.83}{s^2 + 7.25s + 35.04} \tag{4.29}$$

Figures 4.4 and 4.5 show plots of the input, experimental data, and the result of the system identification. We obtained a mean squared error of $1.1760e-04$ for the x axis and $9.405e-05$ for the $y-$ axis model.

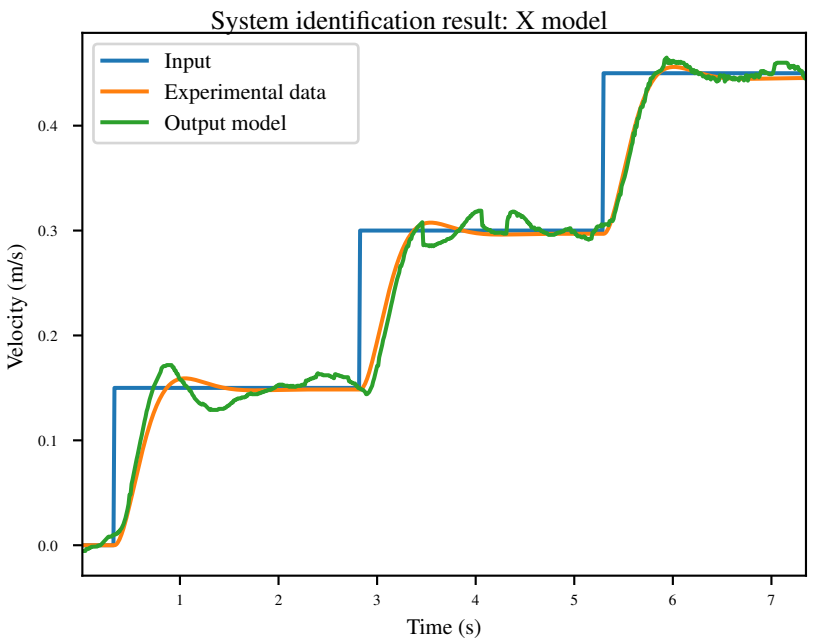

Figure 4.4 X-axis: comparison of input, experimental data, and output from identified model in Equation 4.28

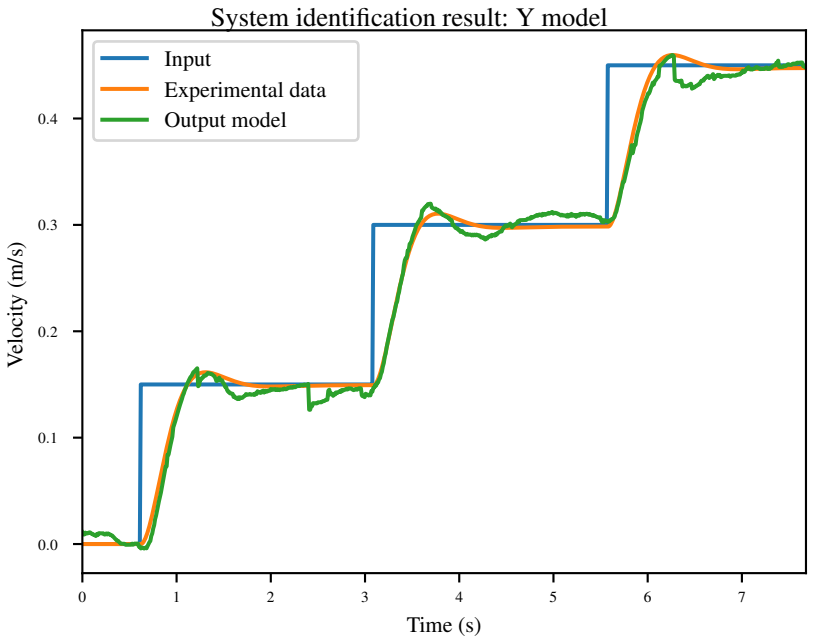

Figure 4.5 Y-axis: comparison of input, experimental data, and output from identified model in Equation 4.29

We modelled the UGV using one integrator. We did not perform identification of the UGV dynamics and its internal controller because at that stage, no controller was implemented in the robot. The model receives as input a velocity reference $\dot{q}_r$ and outputs position q. This model can seen in Figure 4.6.

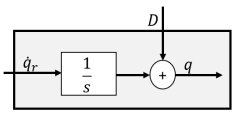

Figure 4.6 UGV simulation model

We model the disturbances as Gaussian noise with mean $\mu = 0$. We converted the two previous models to discrete time models and the integration in simulation was done using a first order Euler's numerical integration method.

5

Implementation

5.1 Simulation

To test the formation algorithm's effectiveness before implementing it in a real robotic system, we performed simulations in Matlab. We used the models explained in Section 4.2.1. Although we intended to implement the controller using the distributed architecture, we implemented the control law for all the robots in a single control loop similar to the centralized architecture. The downside of this implementation approach is that the effect of communication delay might not appear in the results. The code implementation can be found in [Omodolor, 2022a].

5.2 Real-time Experiment

To perform the experiments, we used the Crazyflie quadcopter from Bitcraze [Bitcraze, 1999], see Figure 5.1, and three-wheeled omni-wheel robots from the Department of Automatic Control in Lund, see Figure 5.4. The following section will explain each platform and what controller we used to control them. Finally, we give an overview of the communication structure. We wrote the whole system in ROS [Stanford Artificial Intelligence Laboratory et al., 2018] because of the communication layer and tools it provided.

5.2.1 Crazyflie

The Crazyflie is an open-source lightweight nano quadcopter platform developed by Bitcraze [Bitcraze, 1999]. In order to communicate with the quadcopter, we used the Crazyradio PA, an open-source USB radio bungle, see Figure 5.6. It can reach a range of 1 km in an ideal situation. To interact with the Crazyflie, we used the CrazySwarm library [Preiss et al., 2017] that allows the use of ROS as a middleware. It includes ROS topics that make it easy to communicate with the quadcopter. We used one Crazyradio PA for every two drones to ensure reliability in communication. When we used more than two Crazyflies per Crazyradio, there was a noticeable delay when receiving and sending commands.

Figure 5.1 Crazyflie

The experiments were performed indoors using the Lighthouse positioning system [Taffanel et al., 2021], see Figure 5.2. It uses the streamVR base station of Valve Inc and the lighthouse deck that is placed on the quadcopter to estimate its own position in a global coordinate system by using an extended Kalman filter.

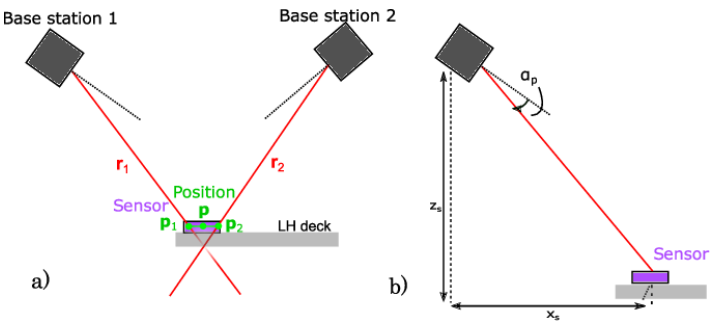

Figure 5.2 Lighthouse positioning system [Taffanel et al., 2021].

5.2.2 Velocity Controller: Crazyflie

This section will explain the structure of the velocity control implemented in the Crazyflie. The controller in the Crazyflie firmware has four levels of control: attitude rate, absolute attitude, velocity and position. Currently, there are three types of controllers: cascaded PID, INDI and Mellinger controller. The main focus will be

on the cascaded controller, which consists of multiple controllers in which the input of one corresponds to the output of another. A block diagram of the PID controller implementation is shown in Figure 5.3.

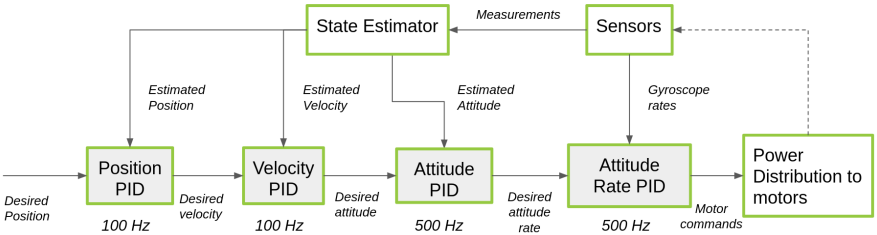

Figure 5.3 Different loops of cascaded PID controllers [Bitcraze, 1999].

The first level of control, the attitude rate PID controller, receives as input the attitude rate and, based on the sensor data from the gyroscope rates, generates the desired motor commands. The second level, the attitude controller, takes in as input the desired attitude and, with the attitude from the state estimator, computes the desired attitude rate. Like the attitude controller, the position and velocity PID controller receives as input the corresponding reference and data from the state estimator to compute the corresponding commands.

The Crazyflie also includes different types of controllers that are combinations of the controllers as shown in Figure 5.3. We took advantage of the controller already implemented in the Crazyflie firmware. For this project, we used the **hovertype** velocity controller. It allows the user to control the body velocity in the x and y plane and the z distance. In order to send velocity command in the global frame, we integrated the acceleration coming from the formation controller to obtain velocity in the global frame. Then we transform the global velocity to local body velocity using the following transformation matrix

$$\begin{bmatrix} \dot{x_G} \\ \dot{y_G} \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} v_{x,B} \\ v_{y,B} \end{bmatrix} \tag{5.1}$$

where ψ is the yaw angle of the drone in the global frame. One disadvantage of this approach is that for a more prolonged experiment period, the velocity reference and the actual velocity value might diverge. [Saif et al., 2019] proposed the use of the roll and pitch as command input to the drone by relating them with the acceleration obtained from the formation controller. We were not successful in implementing this approach because of the lack of stability of the drone positioning system.

5.2.3 Three-wheeled Omni-wheel Robot

The three-wheeled omni-wheel robot (see Figure 5.4) is a 3D printed mobile robot that uses the Dynamixel motors for actuation, the Crazyflie with a lighthouse deck for positioning (see Figure 5.2) and the Raspberry Pi as the main controller.

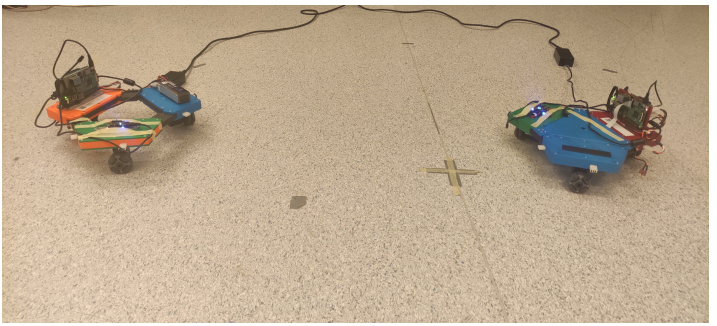

Figure 5.4 Three-wheeled omni-wheeled robot

We implemented using Python, a velocity controller running onboard the robot; the implementation can be found in [Omodolor, 2022b]. To communicate with the velocity controller from an external device, we implemented multiple ROS nodes that publish and subscribe to ROS topics. These topics include state values (position, velocity and acceleration) and velocity command. To control the Dynamixel motors, we used the library [Blomdell, 2022]. It is a re-implementation of the Dynamixel library in Python from the Department of Automatic Control, Lund. Unlike for the Crazyflie, we did not use the CrazySwarm library to log data but the Cflib: Crazyflie Python library by [Bitcraze, 1999].

The schematic representation of the velocity controller can be seen in Figure 5.5. It uses a PI controller. First, we obtain the current state of the robot from the lighthouse deck and apply a median filter to reduce the noise in the position, velocity and acceleration estimate. We compute the error between the reference and the current state and send that to the PI controller. The PI controller computes the velocity command for the robot. Using Equations (4.27) and (5.1), we compute $K^{-1}(.)$ which is the inverse kinematics of the omni-wheel in order to obtain the necessary angular velocity of the wheels. This controller is implemented for each of the axis being controlled, namely $\dot{x}$, $\dot{y}$ and $\dot{\psi}$.

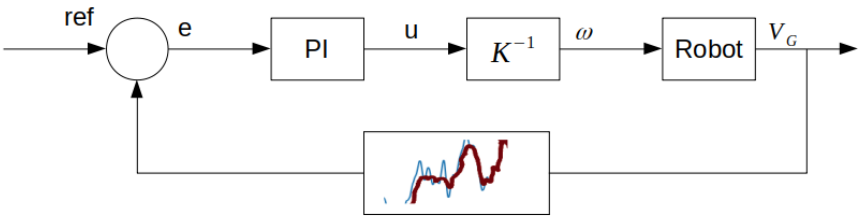

Figure 5.5 Velocity controller for the three-wheeled omni-wheels robot

5.2.4 Communication Architecture

In this section, we will explain the overall communication layer implemented to perform the formation control in a real robot, see Figure 5.6. Each mobile robot (ROS_SLAVE) runs its velocity controller onboard in a node and communicates with the central computer (ROS_MASTER) via ssh using a network router. As we stated earlier, the formation controller requires inter-agent relative distance; since there were no sensors to measure these relative distances. However we used the lighthouse system to extract this information.

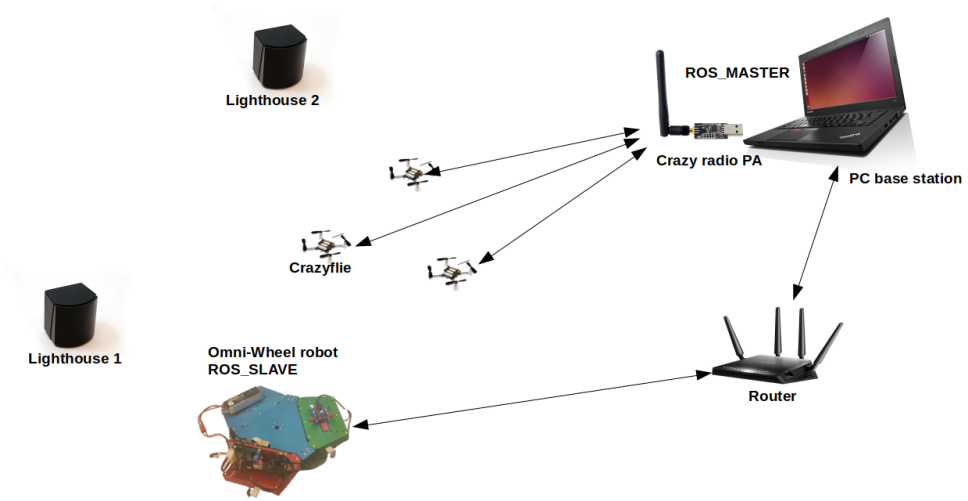

Figure 5.6 Communication architecture

Although we intended to run the formation control law onboard the UAVs, it was not possible to modify the Crazyflie firmware due to time constraints. We simulated the distributed architecture in the main computer as an alternative. The implementation consists of the mission controller, drone and omni-wheel nodes. We have as many nodes as UAVs and as many nodes as omni-wheel, running their robot formation controller separately. This way, we can isolate each system as in a distributed architecture. The mission controller node is responsible for sending the navigation group objectives. Each robot communicates only by subscribing to the corresponding topic that offers state information about robots in its neighbourhood. We implemented each nodes in C++, the implementation can be found in [Omodolor, 2022a]. Figure 5.7 shows how information is exchanged between the robots and the mission planner.

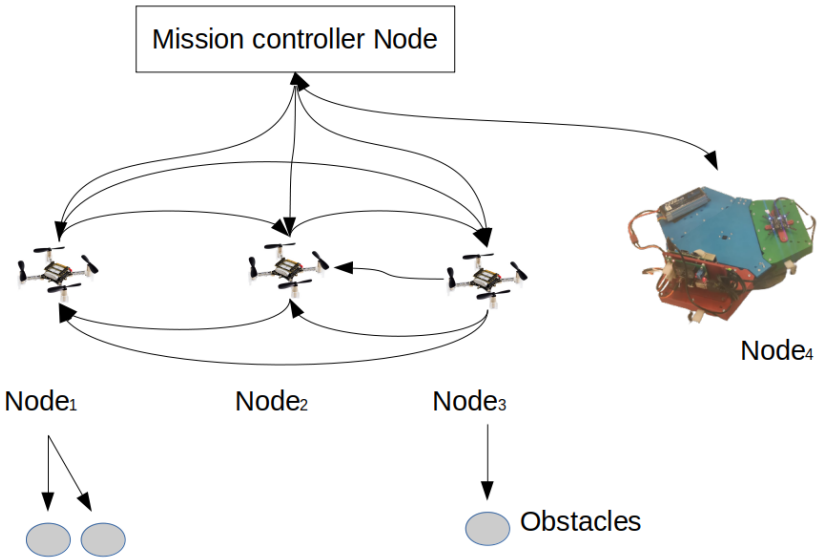

Figure 5.7 Distributed architecture

5.3 Bill of Materials

Table 5.1 shows the bill of material used during this project.

Table 5.1 Bill of materials

Parts	Quantity	Unit price € Inc VAT	Total Price €
Crazyflie 2.1	5	210	1050
Crazyradio PA USB dongle	2	35	70
Extra parts	1	100	100
Battery + charger	8	10	80
HTC Vive	2	250	500
Lighthouse deck	6	90	540
omni-wheel robot	2	900	1800
Netgear Router	1	50	50
Total cost			4190

6

Results

This chapter presents the results obtained from both simulation and experiment. We considered different scenarios for evaluating the effectiveness of the controller developed in this thesis. Each of these scenarios builds upon the previous. We considered the following scenarios:

- Free-space UAV formation.
- Free-space UAV formation with orientation.
- Free-space UAV formation and flocking.
- Obstacle-space UAV formation and flocking.
- Free-space UAV and UGV formation and flocking.

We performed all experiments in the 2D plane, but the algorithm is easily extendable to 3D space. As we stated in Chapter 5, each robot had its node and logged data from its node. However, the results presented only considered the data we logged from the first robot.

6.1 Simulation

6.1.1 Free-space UAV Formation

In this experiment, we performed collision-free formation control with four UAVs using Equation (3.30) extended with Equation (3.36). We investigated the effect of noise in two different scenario in Sections (6.1.1.1) and 6.1.1.2.

6.1.1.1 Scenario I. In this first scenario, we performed collision free formation with guassian noise obtained from experimental data in Section 4.2.1. We used the parameters shown in Table 6.1.

All the c_2 gains were obtained using the equation proposed in [Olfati-Saber, 2006], $c_2 = 2\sqrt{c_1}$.

Table 6.1 Parameters used to perform the experiments in Section 6.1.1.1

Parameter	Value	Parameter	Value
c_1^α	0.2	$a = b$	5
c_3^α	0.09	ε	0.1
c_1^γ	0.25	d	0.5 m
k	7	μ	0
h_α	0.2	σ	0.001
r	3.5		

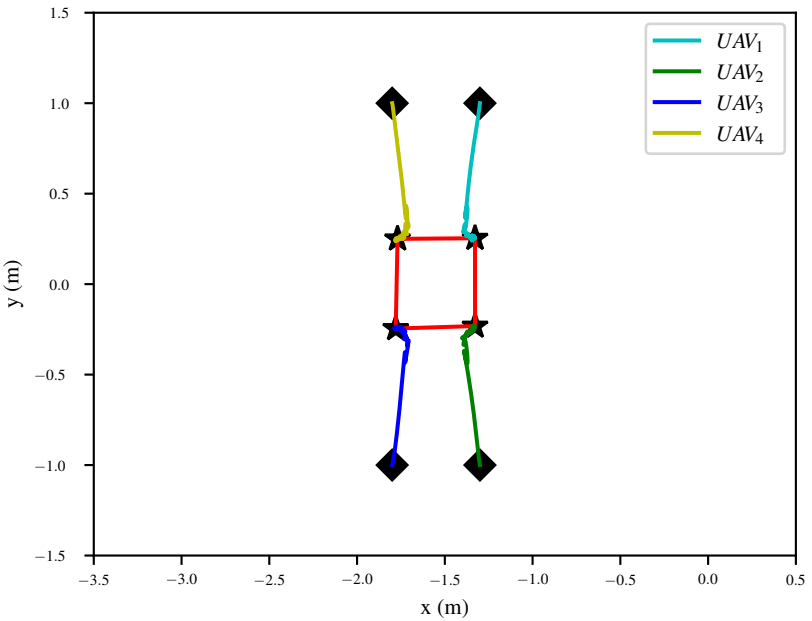

Figure 6.1 Path of the four UAVs for the free-space square formation in (6.1.1.1)

The initial and final positions of the robots are marked with a black **diamond** and black **star**, respectively. Figure 6.1 shows the path of each of the agents during the formation while Figure 6.2 shows the inter-agent distance. As we can see in the plot, the swarm converged to the desired formation. The inter-agent distances eventually converge to the desired value.

We set the navigation objective at the position (-1.55,0) m with velocity (0,0) m/s. Figure 6.3 shows a comparison between the current centroid position

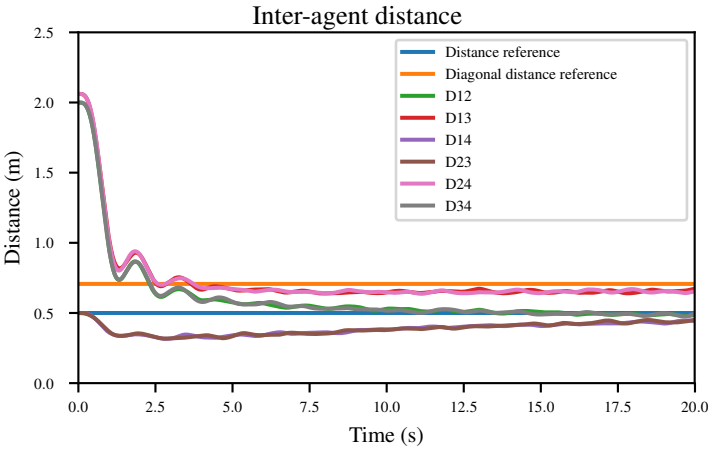

Figure 6.2 Inter-agent distance of the four UAVs for the free-space square formation in (6.1.1.1)

and the target navigation objective. As we can observe, the centroid was able to track the desired goal. The oscillation that appears in Figure 6.3 is mostly due to the Gaussian noise added to the system.

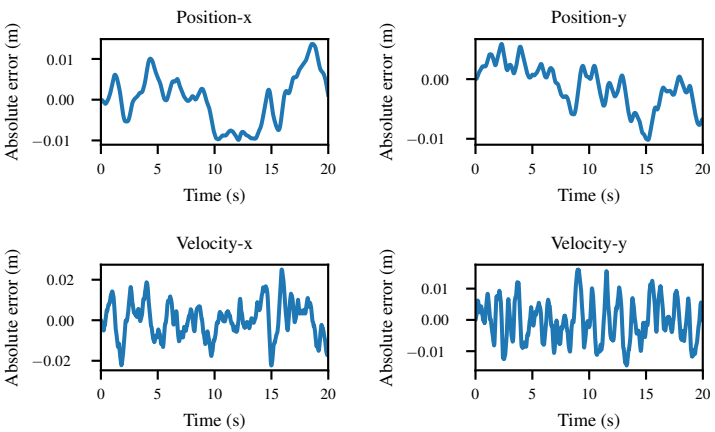

Figure 6.3 Absolute error (m) between the navigation goal (position, velocity) and the current centroid in (6.1.1.1)

Results for a three and five drone formation can be found in Appendix B.1.1 and B.2.1. The results of these experiments show that independent of the number of UAVs the system was still able to converge to the desired formation.

6.1.1.2 Scenario II. To investigate how the controller behaves when subjected to considerable noise, we increased the standard deviation of the noise to $\sigma = 0.04$. Figure 6.4 shows the path each agent followed. The other parameters used are the same shown in Table 6.1.

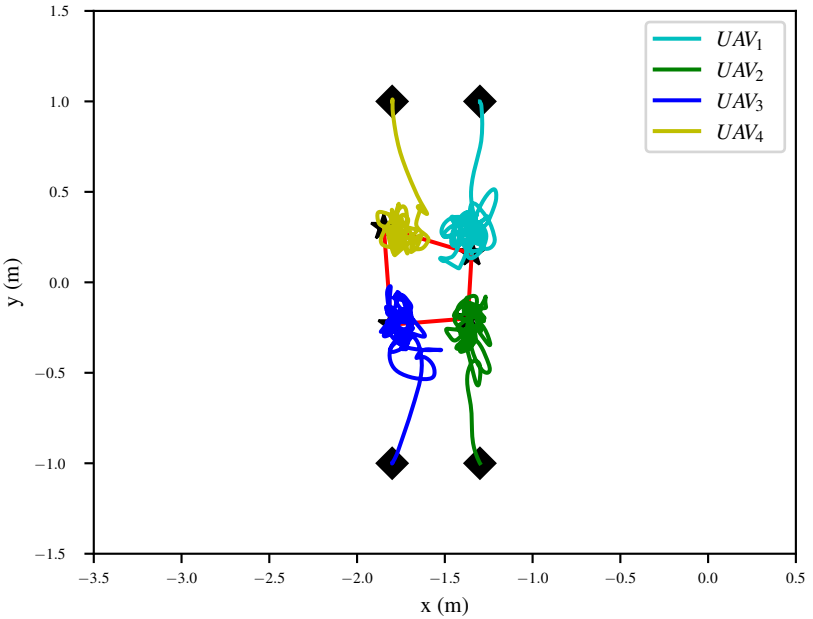

Figure 6.4 Path of the four UAVs for the free-space square formation with noise standard deviation $\sigma = 0.04$ in (6.1.1.2)

As we can observe in Figure 6.5, despite being subjected to significant noise, the agents can maintain the formation. Moreover, similar to the previous scenario, as shown in Figure 6.5, the inter-agent distance did not fully reach the desired value.

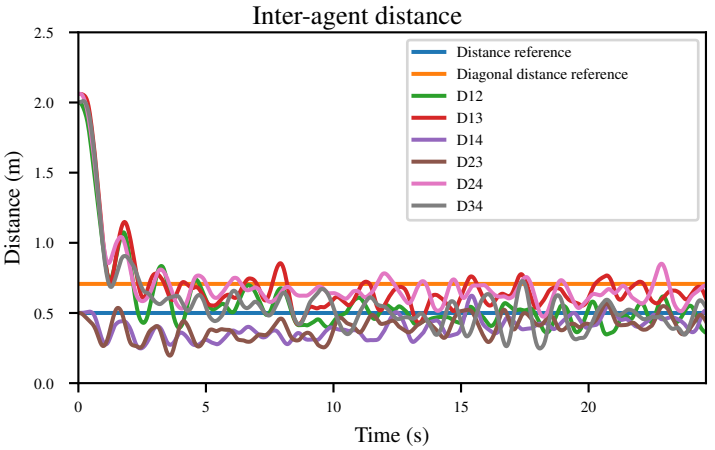

Figure 6.5 Inter-agent distance of the four UAVs for the free-space square formation with noise standard deviation $\sigma = 0.04$ in (6.1.1.2)

6.1.2 Free-space UAV Formation and Orientation

In this experiment, we performed a four drone formation control with orientation control using the Equation (3.43). We used the same parameters and navigation goal stated in Section 6.1.1.1 but the orientation gain $c_1^{\delta} = 0.25$. The orientation vector which defines the orientation of the formation is the vector from the centroid of the formation to UAV_1. We set the desired orientation angle to 0°with respect to the global frame of reference. Figure 6.6 shows the path of the robot. The agents' initial and final positions are marked with a black **diamond** and black **star**, respectively.

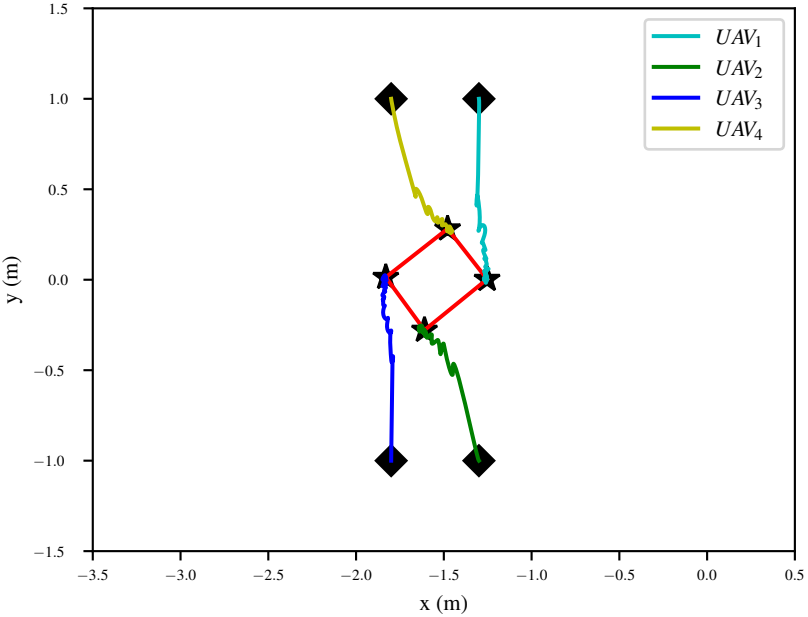

Figure 6.6 Path of the four UAVs for the free-space square formation with 0°orientation with respect to the global frame of reference in (6.1.2)

As we can observe, the system was able to converge to the desired formation while staying close to the desired inter-agent distance, see Figure 6.7. In this experiment, the controller was able to track the navigation goal, see Figure 6.8. With respect to the orientation, the formation was able to reach the desired value, see Figure 6.9.

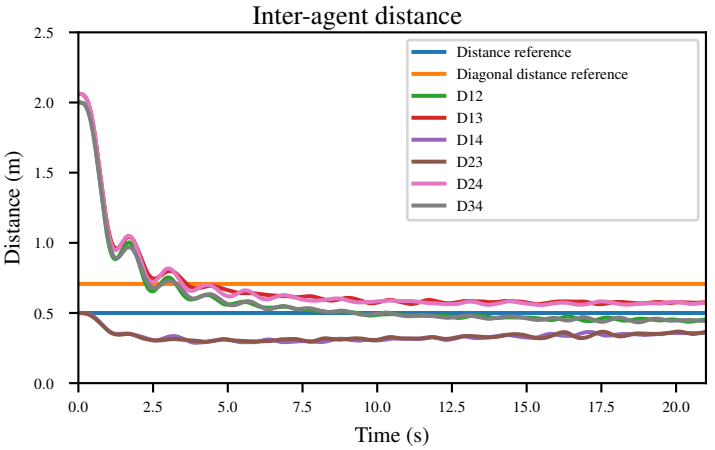

Figure 6.7 Inter-agent distance of the four UAVs for the free-space square formation with orientation in (6.1.2)

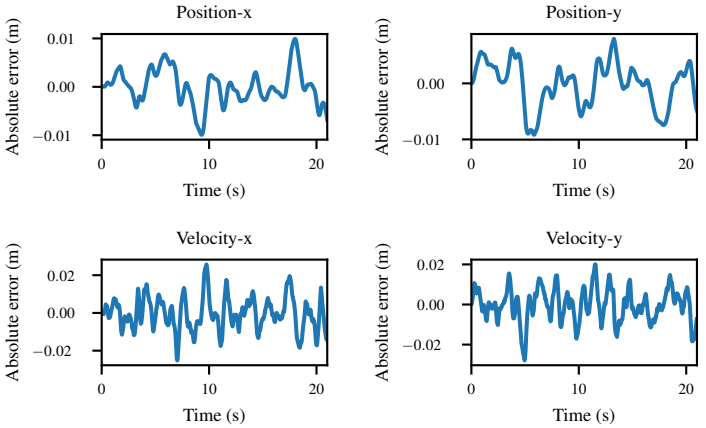

Figure 6.8 Absolute error (m) between the navigation goal (position, velocity) and the current centroid in (6.1.2)

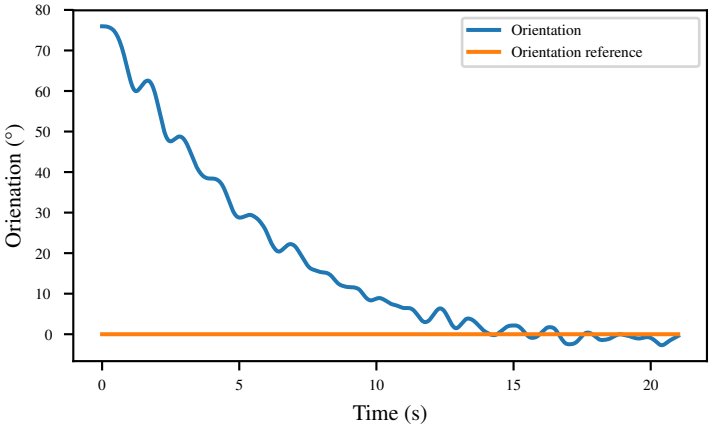

Figure 6.9 Evolution of the orientation in (6.1.2)

Results for a three and five drone formation can be found in Appendix B.1.2 and B.2.2. Similarly, as we observe the Figures in the corresponding Appendix, the swarm was able to converge to desired orientation while maintaining the formation and inter-agent distances.

6.1.3 Free-space UAV Formation and Flocking

In this experiment, we performed formation and flocking control using four UAVs. We used the same parameters shown in Table 6.1. Figure 6.10 shows the path of the robots. The agents' initial and final positions are marked with a black **diamond** and black **star**, respectively.

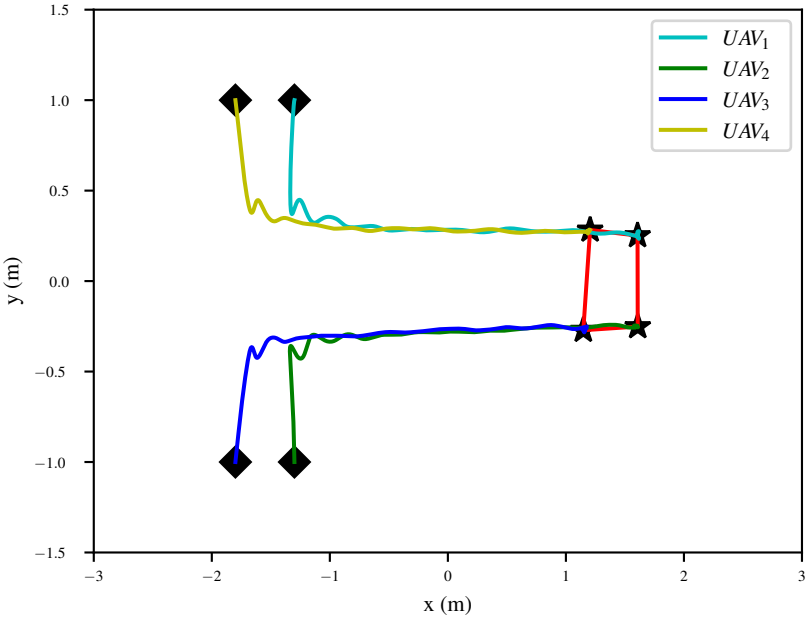

Figure 6.10 Path of the four UAVs for the free-space square formation and flocking in (6.1.3)

As we can observe in the Figure 6.10 and the distance plot in Figure 6.11, the UAVs were able to maintain a formation while flocking to the desired location. However, the inter-agent distance does not fully converge to the desired values. Nevertheless, the error is minor. One interesting observation about these results is that the formation term converged faster than the navigation term, even though the gain of the navigation term was slightly more significant than the gain of the formation term. We did not impose this Behaviour in the code. This Behaviour results from the formation gain influence being more substantial because it considers agents in the surroundings. The navigation goal is at position (1.42,0) m with flocking velocity of (0.0,0.0) m/s. As we can observe in Figure 6.12, the system converges to the desired navigation points.

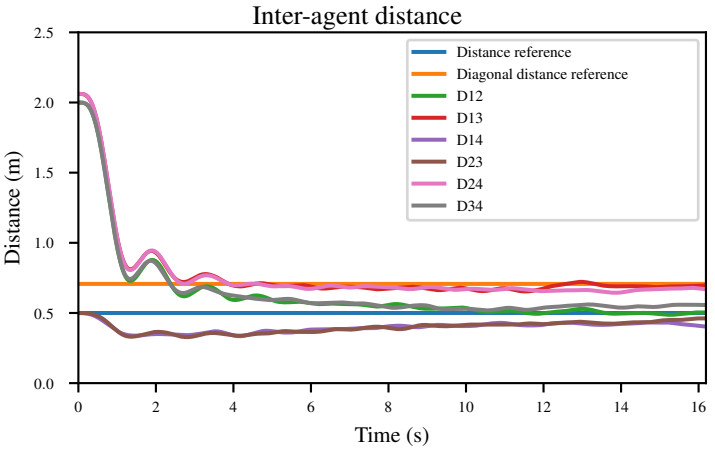

Figure 6.11 Inter-agent distance of the four UAVs for the free-space square formation and flocking in (6.1.3)

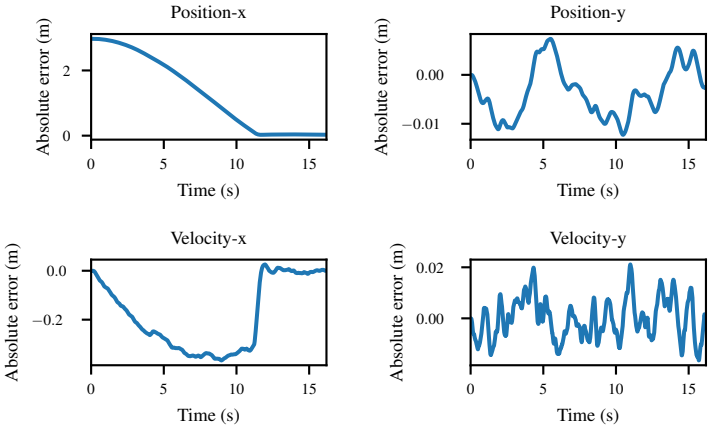

Figure 6.12 Absolute error (m) between the navigation goal (position, velocity) and the current centroid in (6.1.3)

6.1.4 Obstacle-space UAV Formation and Flocking

In this experiment, we performed formation and flocking control with obstacle avoidance with four UAVs. We investigated two different scenarios in Sections 6.1.4.1 and 6.1.4.2 where the position of the obstacle was changed.

6.1.4.1 Scenario I. In this scenario, we placed the obstacle close to the flocking path of the UAVs. The position of the obstacle and path of the agents are shown in Figure 6.14. The navigation goal in this case was (1.42, 0) m with a flocking velocity of (0.1,0) m/s. We placed the obstacle with a radius of 0.25 m at the position (0,0.75) m. The parameters used in this experiment is shown in Table 6.2.

Table 6.2 Parameters used to perform the experiments in Section 6.1.4.1

Parameter	Value	Parameter	Value
c_1^α	0.2	$a = b$	5
c_3^α	0.09	ε	0.1
c_1^γ	0.05	d	0.5 m
k	7	μ	0
h_α	0.2	σ	0.001
h_β	0.9	$ratio$	0.6
r	3.5 m	r_{obs}	2.1 m
d_{obs}	0.3 m		

With the ratio in Table 6.2, we compute the distance to the obstacles, d_{obs}. In addition, using the constant k, we can compute the obstacle's interaction range r_{obs}. Compared to experiment in Section (6.1.1.1), we used a lower c_1^γ gain, see Table 6.2. We did this because the formation otherwise collided with the obstacle.

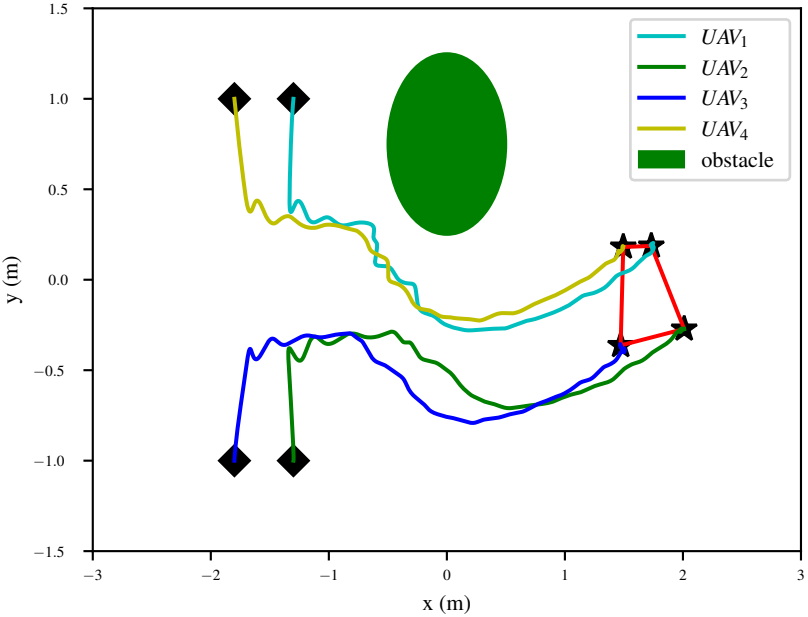

Figure 6.13 Path of the four UAVs for the free-space square formation and flocking with obstacle avoidance in (6.1.4.1)

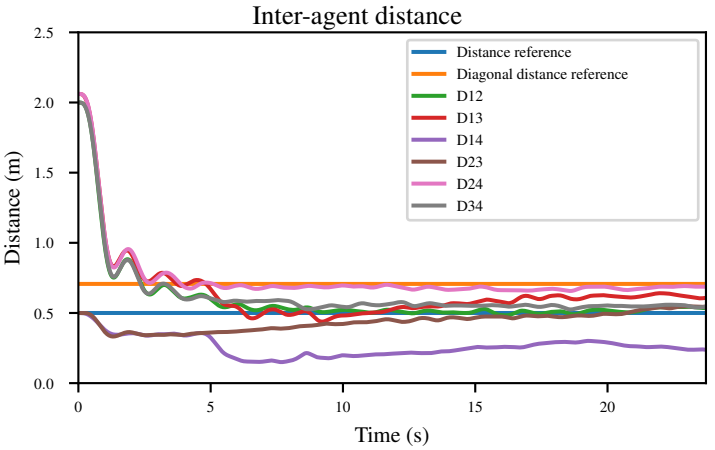

Figure 6.14 Inter-agent distance of the four UAVs for the free-space square formation and flocking with obstacle avoidance in (6.1.4.1)

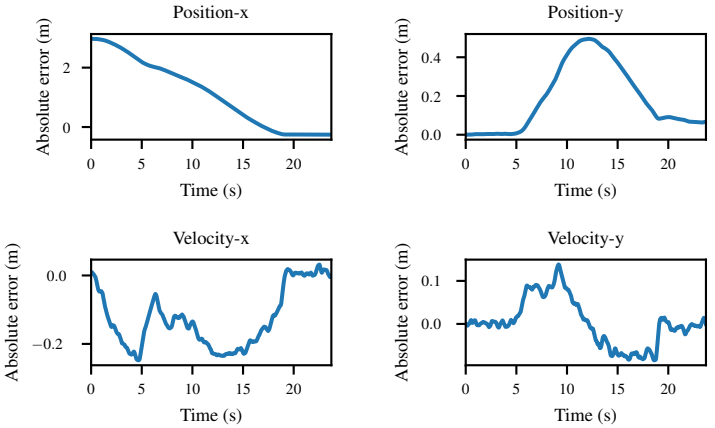

Figure 6.15 Absolute error (m) between the navigation goal (position, velocity) and the current centroid in (6.1.4.1)

In this first scenario the agents were able to flock towards the goal (Figure 6.15), while staying close to the desired inter-agent distance (Figure 6.14).

6.1.4.2 Scenario II. In this second scenario, we placed the obstacle directly on the flocking path of the UAVs. We placed an obstacle of a radius of 0.25 m at the position (0,0) m. The parameters used of those shown in Table 6.2. The path the agent followed is shown in Figure 6.16. As we can see in Figure 6.18, the agents were not able to reach the navigation goal and got stuck in front of the obstacles see Figure 6.15.

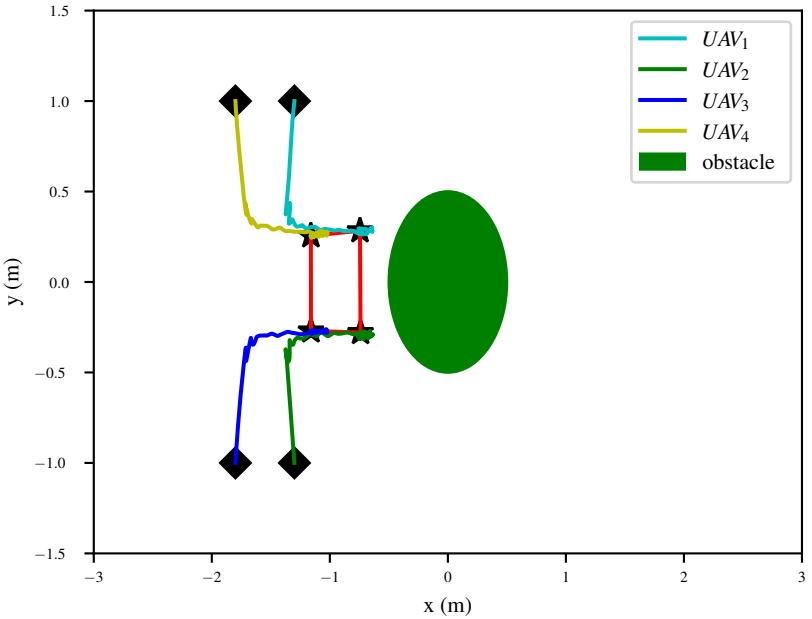

Figure 6.16 Path of the four UAVs for the free-space square formation and flocking with obstacle avoidance in (6.1.4.2)

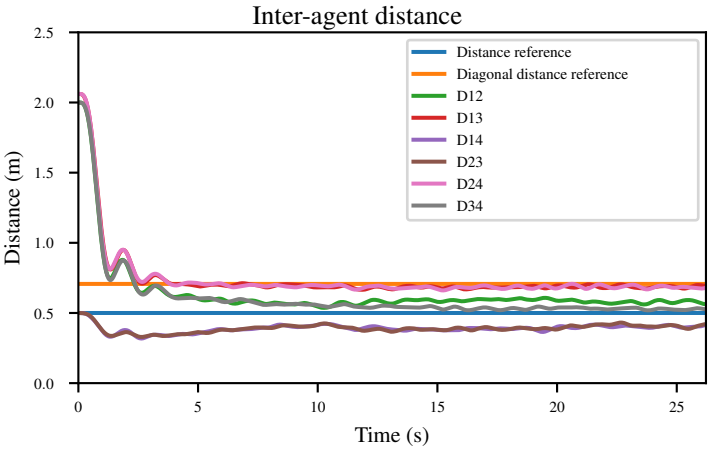

Figure 6.17 Inter-agent distance of the four UAVs for the free-space square formation and flocking with obstacle avoidance in (6.1.4.2)

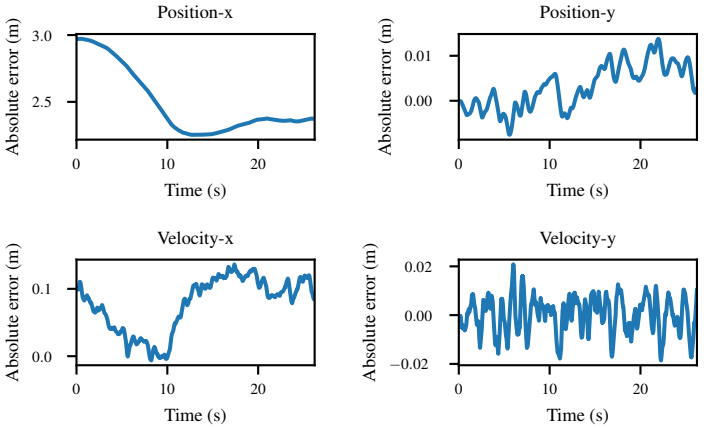

Figure 6.18 Absolute error (m) between the navigation goal (position, velocity) and the current centroid in (6.1.4.2)

This led to the conclusion that this kind of potential-function algorithm could ensure local stability but does not ensure that it converges to the global goal. Finding the ideal gain for the control algorithm's navigation, obstacle and agent terms was quite

challenging. The reason is that increasing one affects the others. For example, too high navigation gains cause the formation to collide with the obstacle.

In both scenarios shown in Sections 6.1.4.1 and 6.1.4.2, we can observe in Figures 6.14 and 6.17, we were able to maintain the desired inter-agent distance. In addition, the formation centroids was able to track the desired navigation goal position and velocity, see Figures 6.15 and 6.18.

6.1.5 Free-space UAV and UGV Formation and Flocking

In the final simulation, we performed formation and flocking control of the UAVs and the UGVs using the algorithm in Section 3.3. The UAV and UGV parameters are the same as the ones shown in Table 6.2, but the obstacle gain for the UGV was set to $c_1^\beta = 2$. We set the distance between the UGV and the cluster to 0.5 m. We also imposed a 180°angle between the cluster and the UGV. Figure 6.19 shows the path of each of the agents and the clusters computed. The UAVs' initial and final positions are marked with a black **diamond** and black **star**, respectively. The UGV's initial and final positions are marked with a black **square** and black **pentagon**, respectively. The clusters' initial and final positions are marked with a black **hexagon** and black **plus**, respectively.

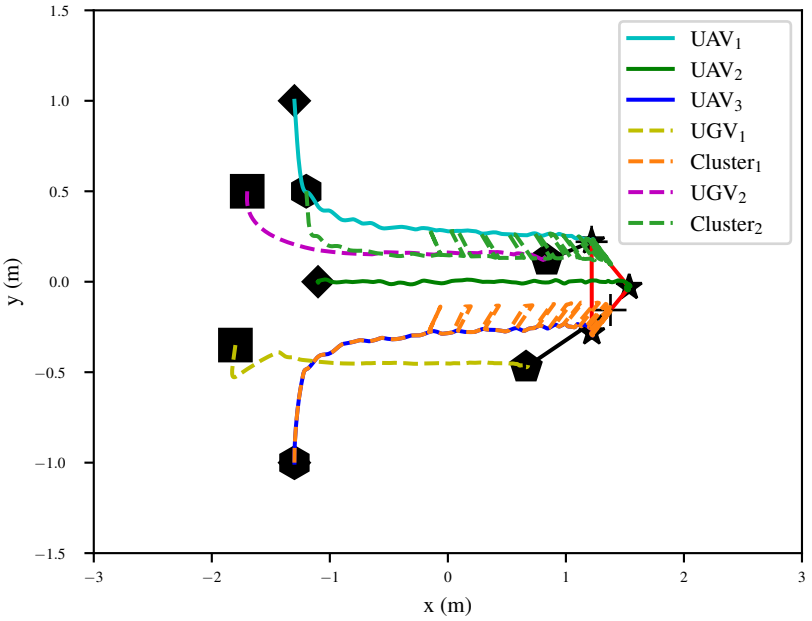

Figure 6.19 Path of the three UAVs, two UGVs and clusters for the free-space UAV and UGV formation and flocking in (6.1.5)

We can observe in Figure 6.19 that the UGVs followed the clusters successfully and the UAVs maintained their desired inter-agent distance while maintaining a safe distance between themselves, see Figure 6.20. We imposed a 180°orientation between the UGV and its corresponding cluster. One observation worth mentioning is the changes in the cluster configuration. We can observe this in Figure 6.19. This change is also noticeable in Figure 6.20, with the random changes in distance from the UGV to clusters. This effect is one of the downsides to the algorithm in Section 3.3. For certain cluster configuration, the result K-means++ clustering is not unique.For example, for a cluster with three UAVs and two UGVs, there are two possible solutions when applying the K-means++ algorithm, see Figure 6.21. UGV_1 can cluster with both C_{2A} or C_{2B} . Since the changes are quick, they did not cause any noticeable oscillation in the path of each of the UGVs. We can observe the distance between the UGV and its corresponding cluster in Figure 6.20.

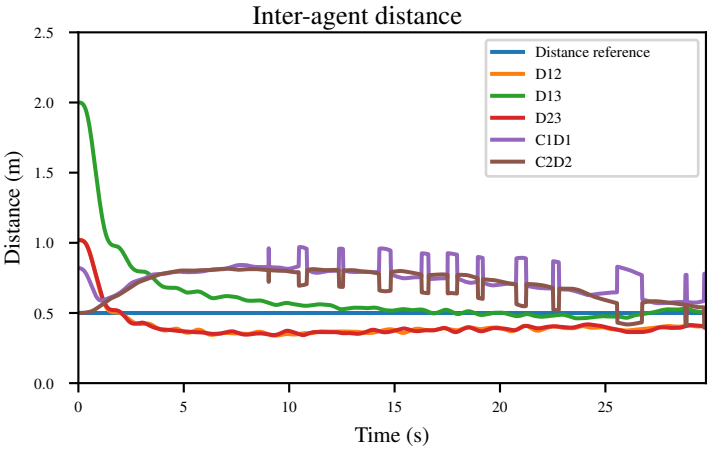

Figure 6.20 Inter-agent distance of the four UAVs and two UGVs for the Free-Space square formation, flocking and coordination of the multi-agent system, CXDY refers to distance from cluster$_x$ to UGV$_y$ in (6.1.5)

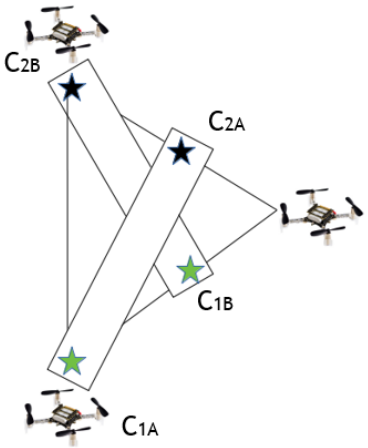

Figure 6.21 Cluster solutions : sub-index **1A** represents cluster 1 solution A in (6.1.5)

6.2 Real-time Experiments

6.2.1 Free-space UAV Formation

In this experiment, we performed a three-UAV formation control using the Equation (3.30) extended with Equation (3.36). We investigated the normal behaviour of the drones in Section (6.2.1.1) and the effect of external noise in Section (6.2.1.2).

6.2.1.1 Scenario I. In this first scenario, we performed collision free formation with three drones. We used the parameters shown in Table 6.3. All the c_2 gains were obtained using the equation proposed in [Olfati-Saber, 2006], $c_2 = 2\sqrt{c_1}$ as we did in the simulations.

Table 6.3 Parameters used to perform the experiments in Section 6.2.1.1

Parameter	Value	Parameter	Value
c_1^α	0.2	$a = b$	5
c_3^α	0.07	ε	0.1
c_1^γ	0.25	d	0.8 m
k	7	r	5.6
h_α	0.2		

We increased the inter-agent distance in the real-time robot for safety concerns. The navigation goal is centered at position (0.2,0) m with (0,0) m/s flocking speed. The initial and final positions of the robots are marked with a black **diamond** and black **star**, respectively. Figure 6.22 shows the path of each of the agents during the formation. A video demonstration of the video can be found in *free_space_3_UAVs_formation.mp4* and link `https://youtu.be/bioNf9_nF8o`.

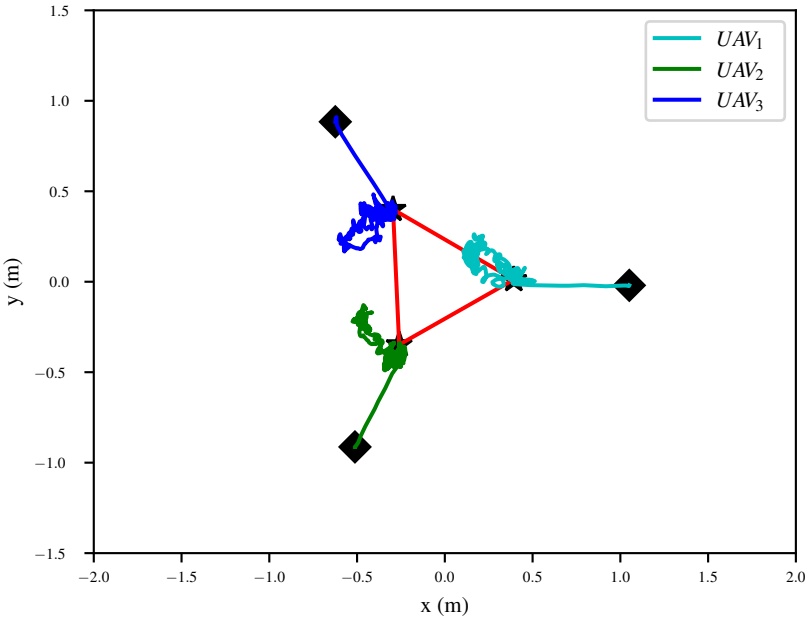

Figure 6.22 Real experiment: path of the three UAVs for free-space formation in (6.2.1.1)

As we can observe in Figure 6.22, the UAVs were able to converge to the desired formation shape. In addition, as we can see in Figure 6.23, the system converged to the desired inter-agent distance quite well, although this happened at the expense of the navigation term, which took longer to converge, see Figure 6.24. Finding the correct tuning parameters was quite challenging. On the one hand, increasing the gain of the navigation term will cause the system to converge to the desired mission goal but might break the formation. On the other hand, increasing the gain of the formation term might ensure we meet the inter-agent distances, but this comes at the expense of increasing the time the navigation goal takes to converge.

An interesting observation worth mentioning is how the controller prioritized the formation term. In Figure 6.23, as the formation converged initially, the absolute error between the navigation goal and the current centroid increased. However, eventually, both the velocity and the position converged. This phenomenon is because, although the navigation gain is higher than the formation gain, the influence of the formation term, which depends on the number of drones in its neighbourhood, has more weight. For a two-drone formation, this would not be the case.

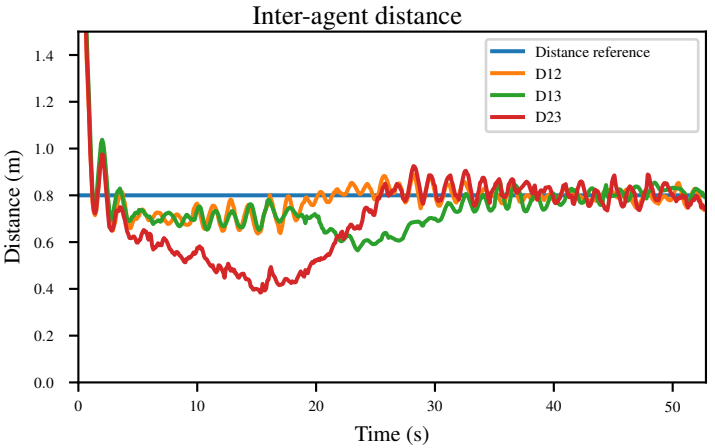

Figure 6.23 Real experiment: inter-agent distance of the three UAVs for free-space formation in (6.2.1.1)

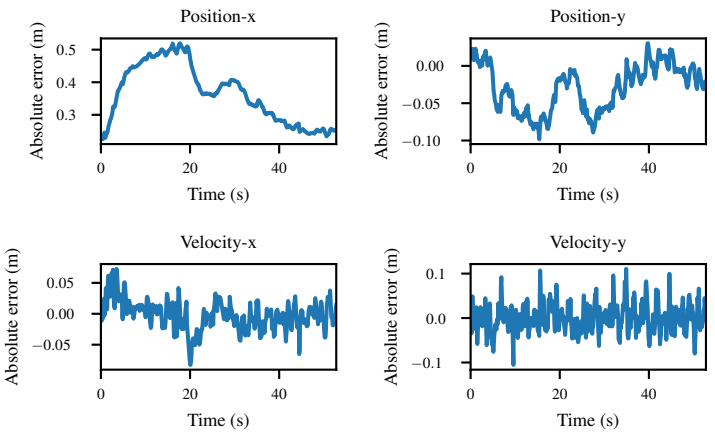

Figure 6.24 Real experiment: absolute error (m) between the navigation goal (position, velocity) and the current centroid of the three UAVs for free-space formation in (6.2.1.1)

6.2.1.2 Scenario II. In this scenario, we performed the same experiment in 6.2.1.1 but manually included external disturbance to investigate the stability of the controller. Figure 6.25 shows the path of each of the agents during the formation. A video demonstration can be found in *free_space_3_UAVs_formation_with_external_disturbance.m* and link `https://youtu.be/OeCVnf5mLak`.

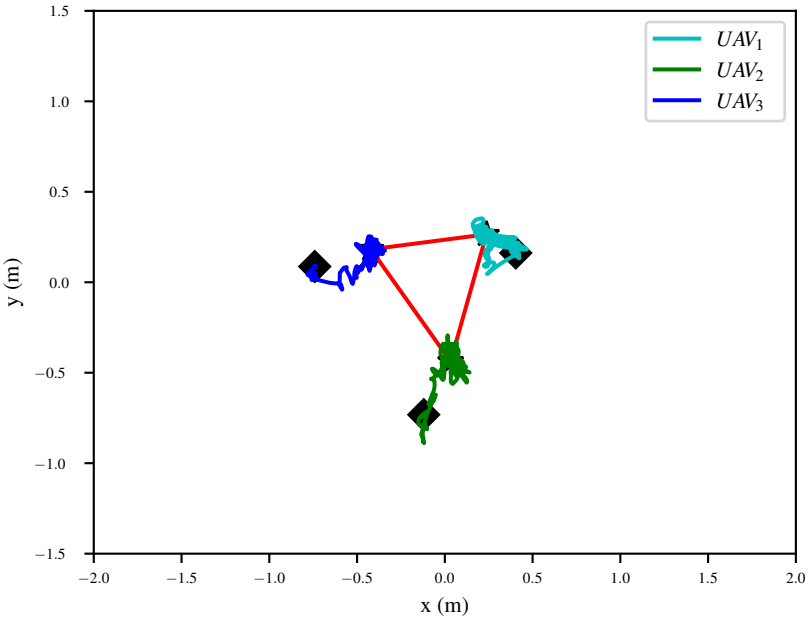

Figure 6.25 Real experiment: path of the three UAVs for free-space formation with external disturbance in (6.2.1.2)

Despite the added noise, the system was able to converge. Although, as we can observe in Figures 6.26 and 6.27, the system was able to recover when noise was added to the system. We can observe this effect in the plot by the sudden rise in absolute errors and the inter-agent distances.

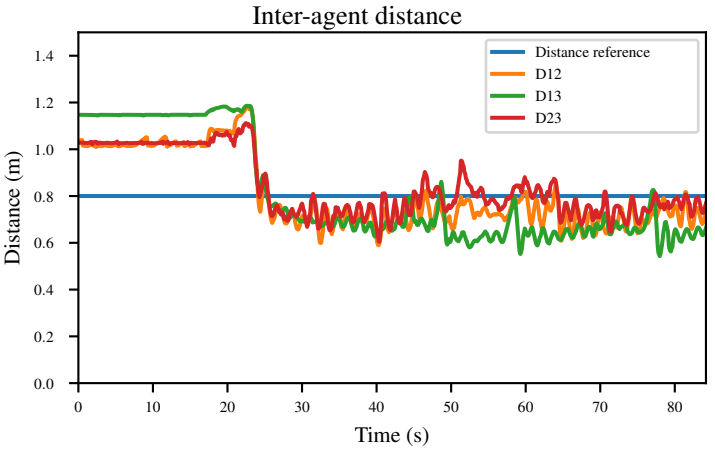

Figure 6.26 Real experiment result: inter-agent distance of the three UAVs for free-space formation with external disturbance in (6.2.1.2)

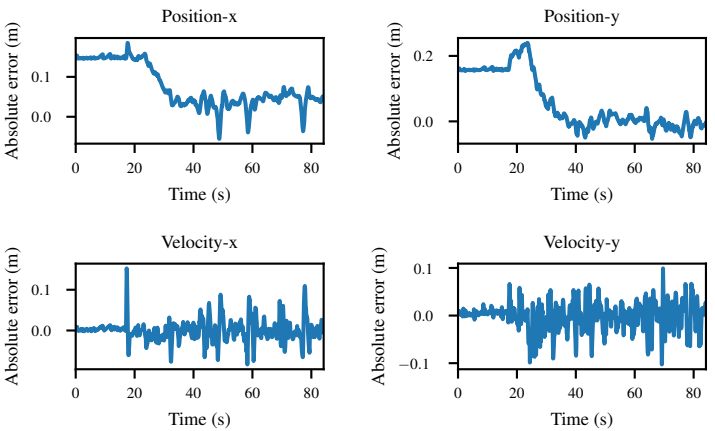

Figure 6.27 Real experiment: Absolute error(m) between the navigation goal(position, velocity) and the current centroid for free-Space square formation with external disturbance in (6.2.1.2)

6.2.2 Free-space UAV Formation and Orientation

In this experiment, we performed a three-drone formation control with orientation. We used the parameters shown in Table 6.3 with the orientation gain $c_1^\delta = 0.1$. As a reminder, the orientation vector is from the centroid to UAV_1. The navigation goal was set to position (0.9, 0) m and velocity (0,0) m/s For this experiment, we set the orientation to 0°with respect to the global frame. The orientation vector which defines the orientation of the formation is the vector from the centroid of the formation to UAV_1. Figure 6.28 shows the path of each of the agents during the formation. The agents' initial and final positions are marked with a black **diamond** and black **star**, respectively. A video demonstration can be found in *free_space_3_UAVs_formation_with_orientation.mp4* and link `https://youtu.be/n9QEYacotDA`.

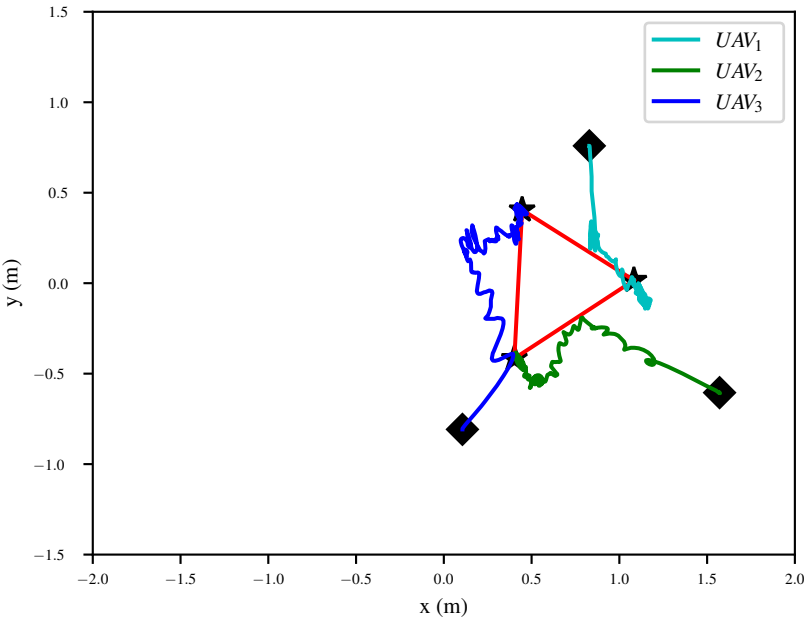

Figure 6.28 Real experiment: path of the three UAVs for free-space formation with 0°orientation in (6.2.2)

Concerning the formation shape, the agent was able to maintain the triangular shape while converging to the desired inter-agent distance, see Figure 6.29.

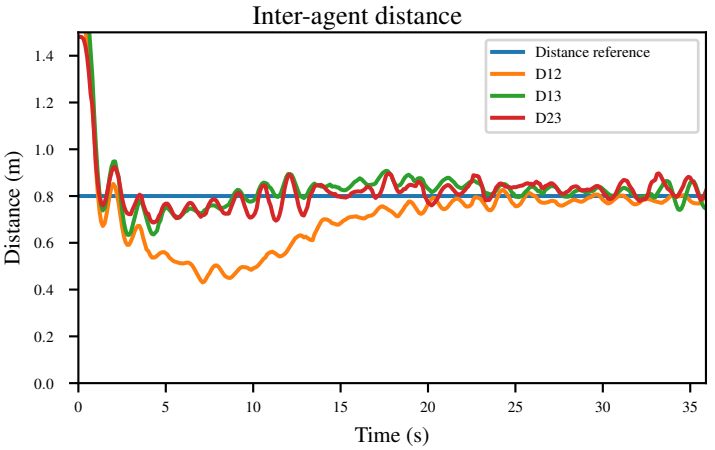

Figure 6.29 Real experiment result: inter-agent distance of the three UAVs for free-space formation with orientation control in (6.2.2)

with respect to the orientation, see Figure 6.31, the formation was able to converge, but at the expense of an increase in the absolute error with respect to the navigation goal, see Figure 6.30.

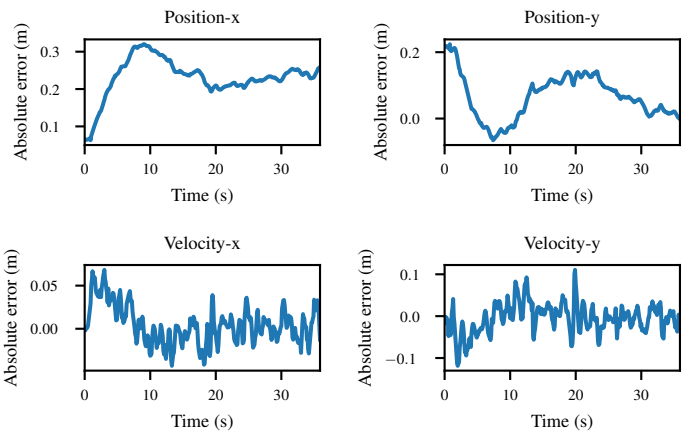

Figure 6.30 Real experiment: absolute error (m) between the navigation goal (position, velocity) and the current centroid for free-space formation with orientation control in (6.2.2)

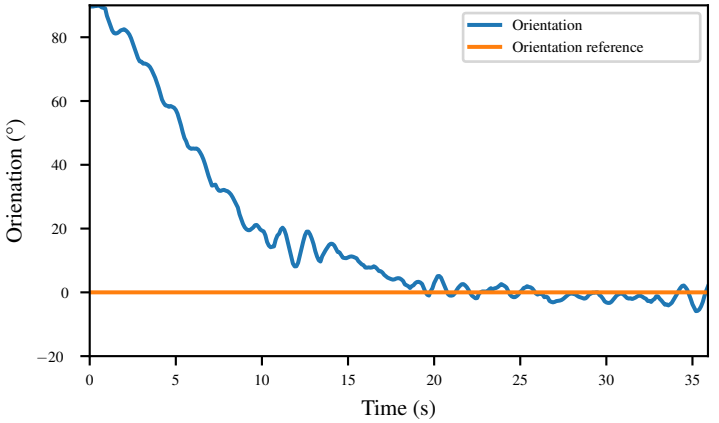

Figure 6.31 Evolution of the orientation in (6.2.2)

6.2.3 Free-space UAV Formation and Flocking

In this experiment, we performed formation and flocking control using three UAVs. We used the same parameters shown in Table 6.3, except for the navigation gain, which we set to $c_1^\gamma = 0.1$ to ensure the swarm maintains the formation while flocking. The navigation goal is at position (1.7, 0) m with a flocking velocity of (0.0,0.01) m/s. Figure 6.32 shows the path of the robots. The agents' initial and final positions are marked with a black **diamond** and black **star**, respectively. A video demonstration of the video can be found in *free_space_3_UAVs_formation_flocking_control.mp4* and link `https://youtu.be/6S0R5yL2P4g`.

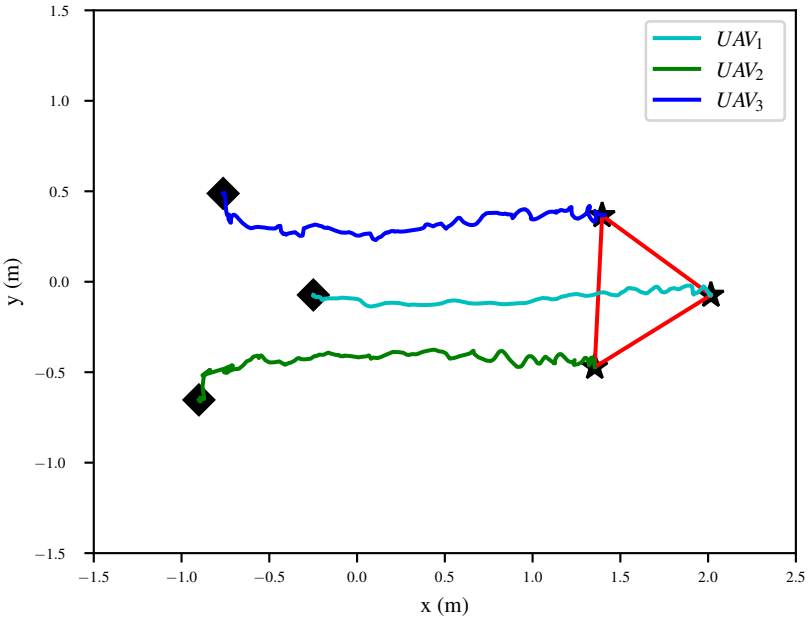

Figure 6.32 Real experiment: path of the three UAVs for free-space formation and flocking in (6.2.3)

As we can observe Figure 6.32 and the distance plot in Figure 6.33, the UAVs were able to maintain a formation while maintaining the desired inter-agent distances. As we can observe in Figure 6.34, the system converges to the desired navigation points. However, there was a lot of oscillation, especially in the velocity. This difficulty is because of the noise in the velocity estimation of the lighthouse positioning system used for the UAVs.

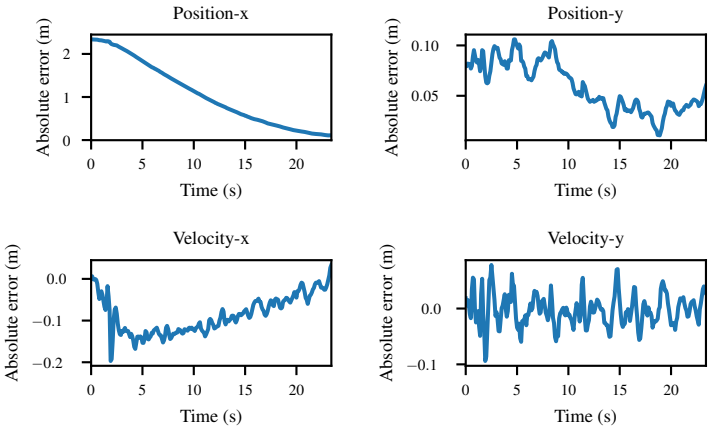

Figure 6.34 Real experiment: absolute error (m) between the navigation goal (position, velocity) and the current centroid for free-space formation and flocking in (6.2.3)

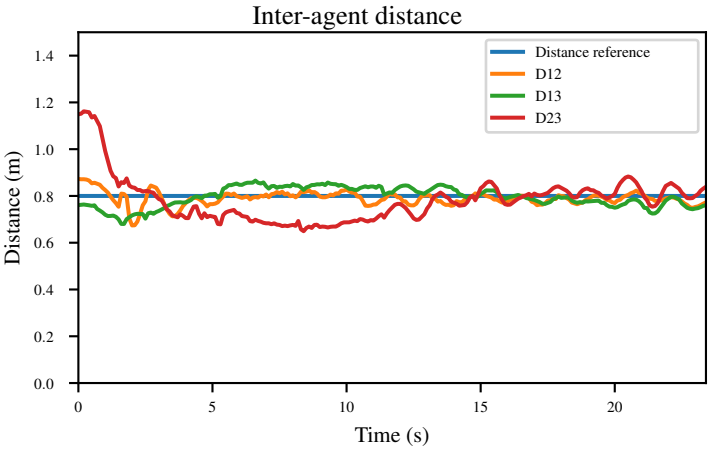

Figure 6.33 Real experiment result: inter-agent distance of the three UAVs for free-space formation and flocking in (6.2.3)

6.2.4 Obstacle-space UAV Formation and Flocking

In this experiment, we performed formation and flocking control with obstacle avoidance with three UAVs and one obstacle. We tested two different scenarios,

in Sections 6.2.4.1 and 6.2.4.2, one where the obstacle was static and the second with a dynamic obstacle. In both scenarios, we used a spare Crazyflie to obtain the obstacle's position.

6.2.4.1 Scenario I. In this first scenario, we placed the obstacle close to the flocking path of the UAVs. The position of the obstacle is shown in Figure 6.35. The navigation goal, in this case, was (1.7,0) m with a flocking velocity of (0.01,0) m/s. We assume the obstacle was of radius 0.25 m. We used the same parameters as shown in Table 6.3. The ratio between the desired distance and distance to obstacle was $ratio = 0.8$, $h_\beta = 0.9$. Given the ratio, we can compute the obstacle's interaction range r_{obs} and the distance to the obstacles, d_{obs}. Figure 6.35 shows the path of each of the agents and the obstacle during the formation. A video demonstration of the video can be found in *formation_and_flocking_with_static_obstacle_avoidance.mp4* and link `https://youtu.be/a_QhH9tYjm4`.

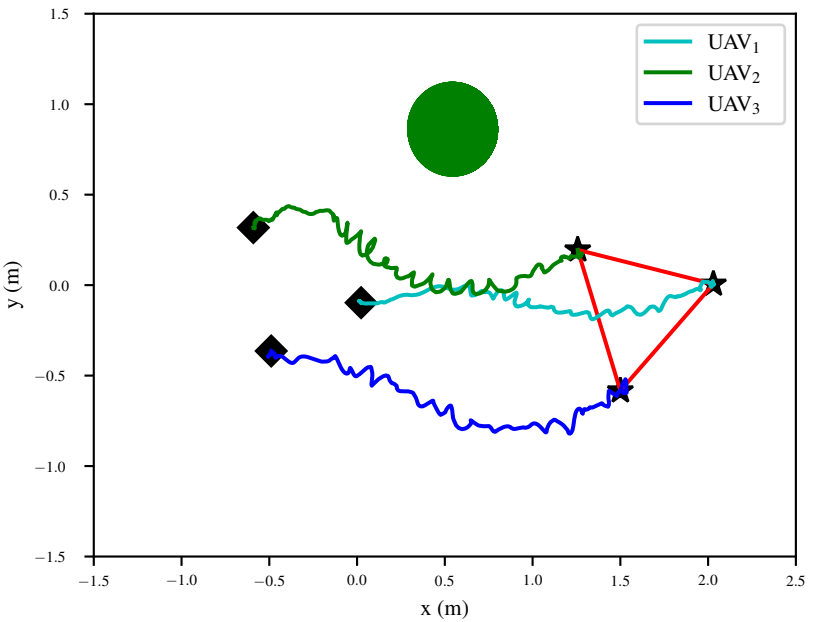

Figure 6.35 Real experiment: path of the three UAVs for free-space formation and flocking with obstacle avoidance in (6.2.4.1)

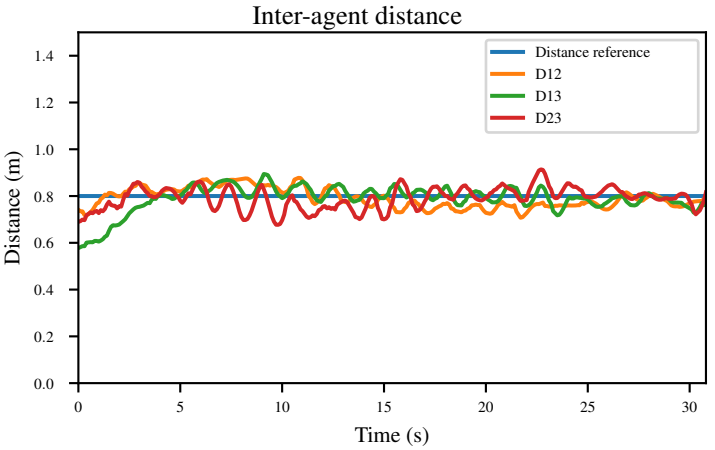

Figure 6.36 Inter-agent distance of the three UAVs for formation and flocking with static avoidance in (6.2.4.1)

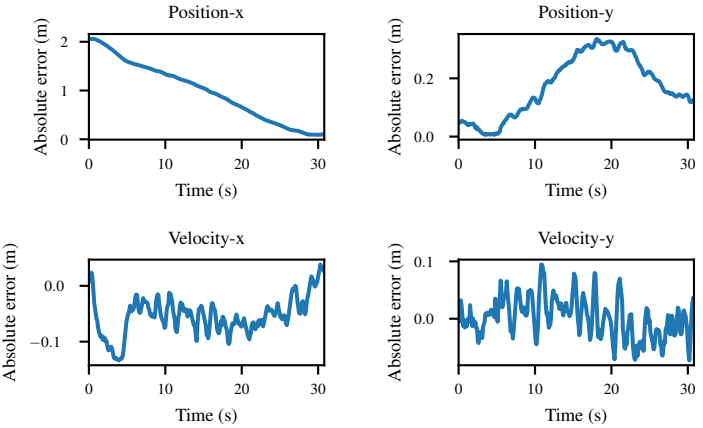

Figure 6.37 Absolute error (m) between the navigation goal (position, velocity) and the current centroid for formation and flocking with static obstacle avoidance in (6.2.4.1)

In the first scenario the agents were able to flock towards the goal (Figure 6.35) while staying close to the desired inter-agent distance (Figure 6.36). Concerning the obstacle, the agents could maintain the desired safety distance from the obstacle,

see Figure 6.38. In this figure, we can observe that all distances are larger than the safety distance. Figure 6.37 shows the absolute error of the mission goal. Similar to previous experiments, the system was able to follow the goal but diverged from the typical straight path it followed in the previous experiment because of the obstacle.

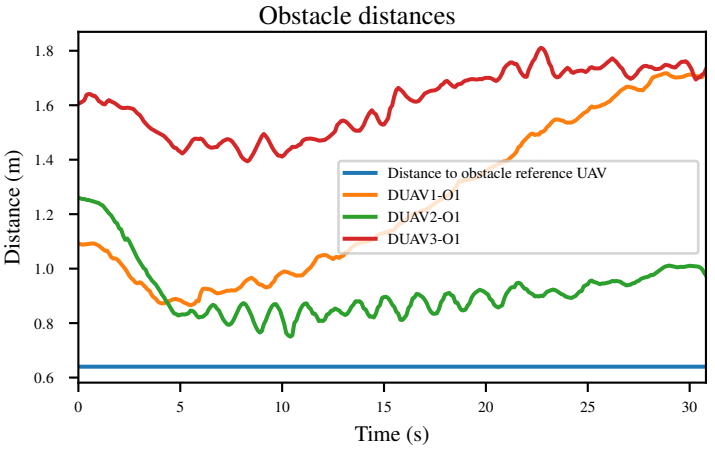

Figure 6.38 Distance of the three UAVs to the current obstacle for formation and flocking with static obstacle avoidance in (6.2.4.1)

6.2.4.2 Scenario II. In this second scenario, we used the same parameters shown in Table 6.3. However, we dynamically changed the location of the obstacle to see how the formation would react. With respect to the mission goal, we fixed this to position (0.0,0) m and velocity (0,0) m/s. We randomly changed the location of the obstacle to see how the formation would react. The agents' trajectory and the obstacle's positions are shown in Figure 6.39. A video demonstration of the video can be found in *formation_and_flocking_with_dynamic_obstacle_avoidance.mp4* and link `https://youtu.be/ajUXCxcN_j8`.

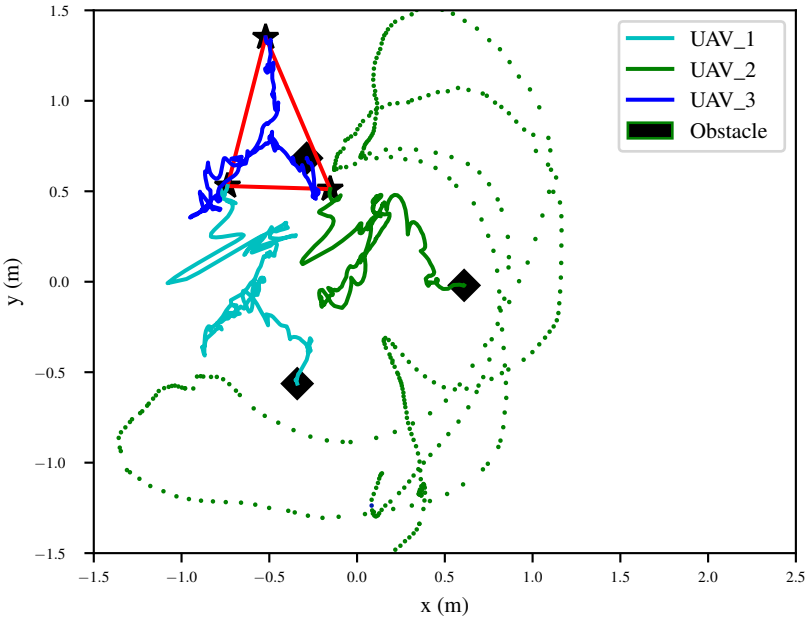

Figure 6.39 Real experiment: Path of the three UAVs for free-Space formation and flocking with dynamic obstacle avoidance second scenario in (6.2.4.2). The discontinuous green line is the position of the obstacle while the continuous one is the position of UAV_3

As we can observe in Figure 6.41, the agents were able to stay above the minimum safety distance, but at the expense of breaking the desired inter-agent distance, see Figure 6.40. In addition, we can observe in Figure 6.41 that the system could recover after the obstacle got too close to the UAV. The slow response time is because to fight against the obstacle; the obstacle potential must overcome the repulsive potential caused by agents in the neighbourhood. This slow response of the whole formation would hinder the application of this method for missions that require fast dynamics.

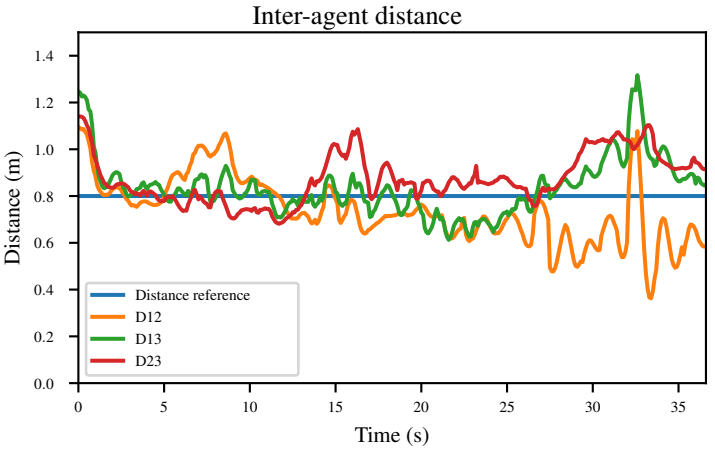

Figure 6.40 Inter-agent distance of the three UAVs for formation and flocking with dynamic obstacle avoidance in (6.2.4.2)

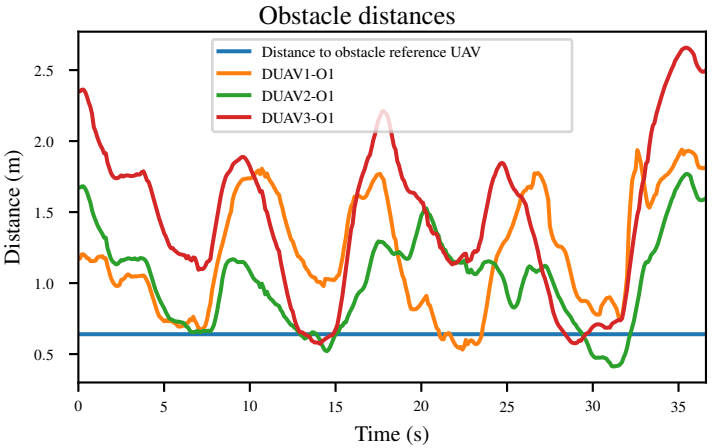

Figure 6.41 Distance of the three UAVs to the current obstacle for formation and flocking with dynamic obstacle avoidance in (6.2.4.2)

6.2.5 Free-space UAV and UGV Formation and Flocking

We performed formation and coordination control in an experiment with multiple UAVs and UGVs. We used parameters for the UAV similar to the ones used in

Table 6.3. As for the UGV, we used the paramters in Table 6.4.

Table 6.4 Parameters (UGV) used to perform the experiments in Section 6.2.5

Parameter	Value	Parameter	Value
c_1^α	1	$a = b$	5
c_3^α	0.07	ε	0.1
c_1^γ	0.25	d	0.9 m
k	7	r	6.3
h_α	0.2	h_β	0.9
$ratio$	0.8	d_{obs}	0.8
r_{obs}	5.6		

Given the ratio in Table 6.4, we can compute the distance to obstacle, d_{obs} and the interaction range of an obstacle. Given the constant k, we can compute the obstacle's interaction range r_{obs}.The navigation goal was set to position (1.70,0) m and velocity (0,0) m/s. As a reminder from Section 3.3, we perform two agent formation between the UGVs and their corresponding cluster. The distance between UGVs and their clusters is 0.9 m.

In order to evaluate the method in Algorithm 1, we tried two Scenarios in Sections 6.2.5.1 and 6.2.5.2.

6.2.5.1 Scenario I. In this first scenario, we did a formation and flocking between three UAVs and one UGV . The UAVs' initial and final positions are marked with a black **diamond** and black **star**, respectively. The UGVs' initial and final positions are marked with a black **square** and black **pentagon**, respectively. The cluster's initial and final positions are marked with a black **diamond** and black **plus**, respectively. Figure 6.42 shows the path of each of the agents during the formation and coordination including the computed clusters. A video demonstration of the video can be found in *formation_and_coordination_3_UAVs_1_UGV.mp4* and link `https://youtu.be/TUC9AUtMm7s`.

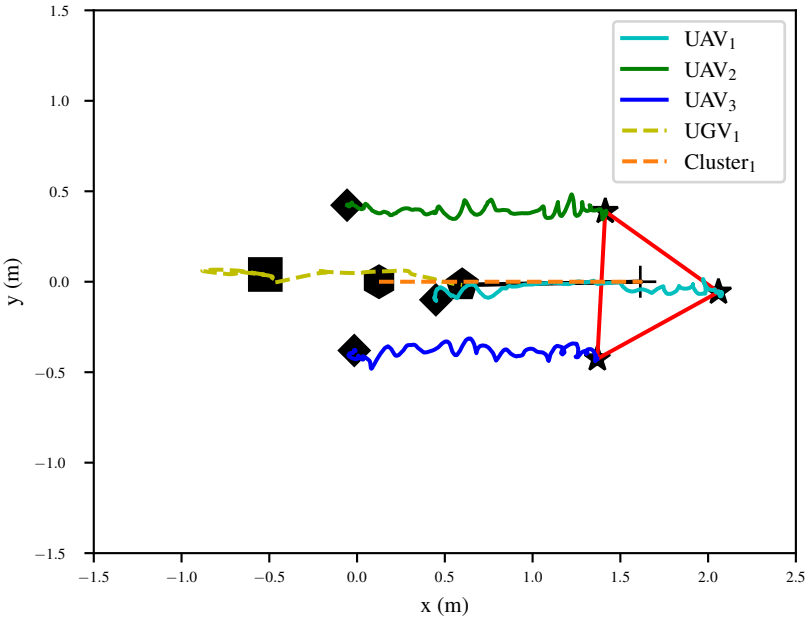

Figure 6.42 Real experiment: path of the three UAVs, cluster and one UGV for formation and flocking in (6.2.5.1)

As we can see in Figure 6.42, the cluster, in this case, corresponds to the centroid of the three UAV formation. In addition, the UGV could maintain the following distance to the cluster, see Figure 6.43.

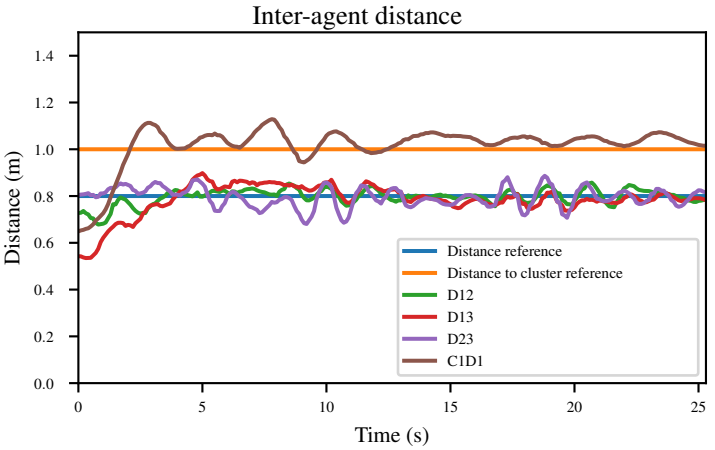

Figure 6.43 Inter-agent distance of the three UAVs, cluster and one UGV for formation and flocking in (6.2.5.1)

6.2.5.2 Scenario II. In this second scenario, we performed a formation and coordination between three UAVs and two UGVs. One UGV considers all UGVs in its neighbourhood as an obstacle. Figure 6.44 shows the trajectory of each of the agents and clusters. A video demonstration of the video can be found in *formation_and_coordination_of_3_UAVs_2_UGVs.mp4* and link `https://youtu.be/Pn2n2A01h_8`.

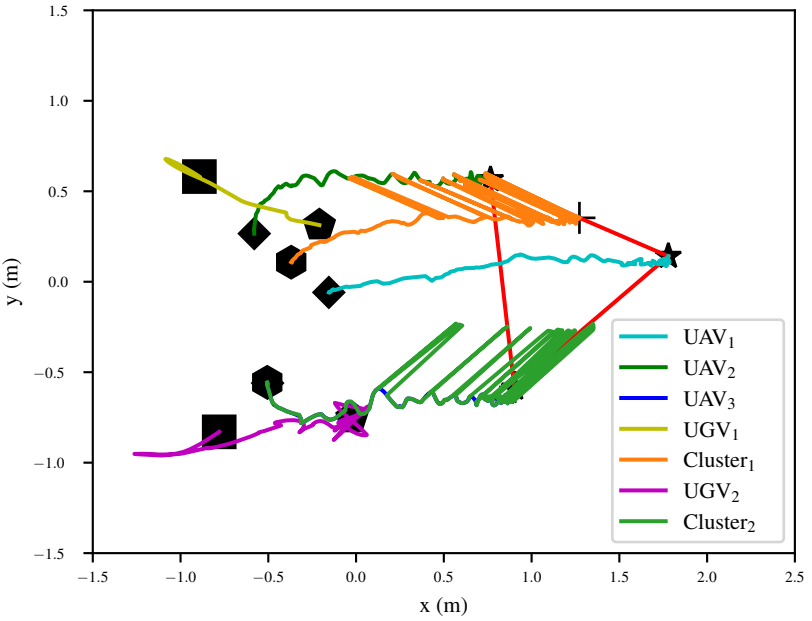

Figure 6.44 Real experiment: path of the three UAVs, cluster and two UGVs for formation and flocking in (6.2.5.2)

Similarly to the simulation in Section 6.1.5, the UGVs were able to follow their corresponding cluster. However, the issue of multiple local solutions is more prominent and causes noticeable oscillations in the UGV movements because of constant change in the cluster position, making it difficult for the distance UGV-to-cluster to converge, see Figure 6.45.

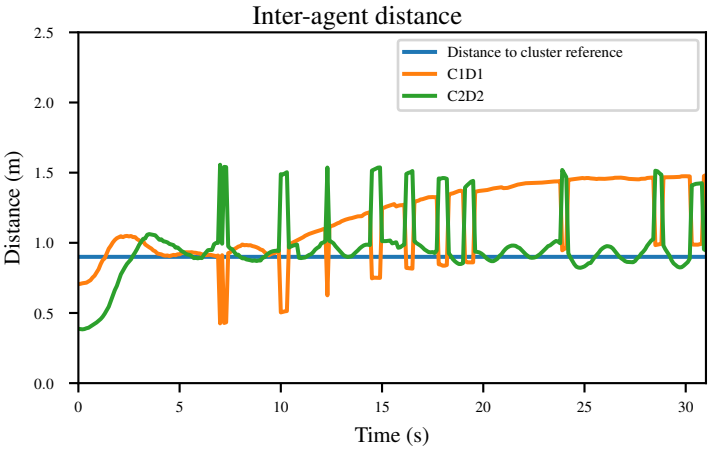

Figure 6.45 Inter-agent distance between UAV cluster and two UGVs for formation and flocking in (6.2.5.2)

7

Conclusion

The objective of this thesis project was to implement in a real robot system a distributed controller for formation and coordination between UAVs and UGVs. To accomplish this, we investigated the current approaches in literature. We then chose the one that best satisfies the condition based on our use case, flight duration, computation load and mission complexity and adopted it for out system.

With the first algorithm, we focused on the agents' formation, obstacle avoidance and mission goal for a second-integrator dynamic system. In order to consider the effect of non-linear dynamics in a real-world implementation, we introduced, using an approach in the literature, a second algorithm that included an integral term. To implement this on the UGV, we extended the original algorithm to a single-integrator dynamical system. We also proposed a new potential term, "Rotational potential", that makes it possible to rotate the formation with one of the agents, **orientation agent**, see Section 3.2.2. Finally, we developed an algorithm that allows the formation and coordination of UAVs and UGVs using the concept of **cluster configuration,** $\mathbb{C}_q$, comprising the energy to maintain said configuration and energy to change said configuration, see Section 3.3.

During the project, we performed multiple evaluations, both in simulation, using a simplified model of the UAVs and UGVs, and experiments on real hardware, Crazyflie and the three-wheeled omni-wheel robot.

Looking at the results obtained in the experiments of this project, we can conclude that we were able to accomplish the objective of this thesis stated in Chapter 1. However, there were many difficulties that arose during the implementation in this thesis. Out of all, the most challenging and time-consuming task was finding the ideal combination of gains for the distributed control law. This issue was noticeable when we included obstacles in the experiment. Every experiment that required to change the position of a static obstacle, we would require different tuning values to accomplish the formation, obstacle avoidance and navigation goal objectives. The tuning becomes very tedious when we have multiple agents, each of which would require its tuning parameters. In addition, the dynamics of the whole swarm formation was slow. This is typical of potential-based

approaches. This might hinder its application when a fast reaction time, for example, in an environment with an obstacle moving at a relatively high speed, is required.

With respect to the problems present in Section 6.1.4.1 about the difficulties tuning the gains when there is obstacle, one alternative solution, which is out of this project's scope, is using an external planning algorithm to re-plan the navigation goals. This way, we can combat the global stability issue. Another alternative is to make the formation gain flexible by applying an adaptive tuning. We can apply this adaptive tuning by relating the distance to the obstacle to the c_1^α gain. We can do this using the exponential decay function:

$$c_1^\alpha(d_{obs}) = c_1^\alpha(1-b)^{d_{obs}} \tag{7.1}$$

where b is the decay rate.

An additional alternative would be model predictive control (MPC) [García et al., 1989]. Thanks to its predictive nature, we can give information about the obstacles to the controller in the form of constraints. With prior knowledge of the environment's state and constraints, the controller could prevent situations like the one observed in Section 6.1.4.2, where the formation got stuck due to the obstacle. A downside to using MPC is the computational requirement which might hinder implementation in systems with relatively lean agents like the Crazyflie UAV.

7.1 Future Work

Although we accomplished the main objectives of the thesis, there are specific improvements that we can make to the current implementation in order to obtain a more reliable and distributed system.

The current implementation mimics the distributed sensing using the lighthouse localization system. One way to thoroughly investigate the developed algorithm in Secion 3.3 is by investigating how to implement hardware sensors that can measure inter-agent relative distances without needing a global positioning system. By doing so, effects like delay will become more prominent in the controller and this would require modification to the original algorithm.

One further research would be implementing a more advanced controller like MPC [García et al., 1989] to test and compare with the PID approach used in this project.

The current controller uses the internal velocity control to control the Crazyflie. However, we receive acceleration commands from the formation control law. We convert acceleration to velocity using the Euler integration method. The drift caused by this integration can become problematic for a prolonged experiment period. For further work, we want to investigate different acceleration controller designs and implementations for the firmware of the Crazyflie.

The current implementation does not consider the effect of the communication delay. More investigation would be needed to quantify how the delay affects the formation performance and how we can include this effect.

The current software implementation, although implemented in a distributed form, still runs on one computer. However, implementing the distributed control law in each robot would be challenging because it would require modifications in the robot's firmware.

In order to mitigate the local stability issue mentioned in Section 6.1.4.1, further investigation on different approaches is needed to include an external planner.

Finally, an interesting topic worth investigating is how the algorithm we developed for coordinating the UAV and UGV would work for a high number of agents. The current approach requires a centralized system to compute the clusters and send this back to each UGVs. Centralized approaches like this are prone to failure. Further investigation on distributing this computation requirement between the UGVs would also be needed.

Bibliography

Arthur, D. and S. Vassilvitskii (2007). "K-means++: the advantages of careful seeding". In: *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, New Orleans, Louisiana, pp. 1027–1035. ISBN: 978-0-898716-24-5.

Bakule, L. (2008). "Decentralized control: an overview". *Annual Reviews in Control* **32**, pp. 87–98. DOI: 10.1016/j.arcontrol.2008.03.004.

Bitcraze (1999). *Controller in the Crazyflie*. URL: https://www.bitcraze.io/documentation/repository/crazyflie-firmware/master/functional-areas/sensor-to-control/controllers/#overview-of-control (visited on 2022-03-01).

Blomdell, A. (2022). *Reimplemenation of Dynamixel library intended to be more Pythonic*. URL: git@gitlab.control.lth.se:anders_blomdell/dynamixel.git. (accessed: 14.05.2022).

Chen, Y. Q. and Z. Wang (2005). "Formation control: a review and a new consideration". In: *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Alberta, Canada, pp. 3181–3186. DOI: 10.1109/IROS.2005.1545539.

Diestel, R. (2010). *Graph Theory*. Fourth. Vol. 173. Graduate Texts in Mathematics. Springer, Heidelberg; New York. ISBN: 9783642142789 3642142788 9783642142796 3642142796. URL: https://doi.org/10.1007/978-3-662-53622-3.

Fionda, V. and L. Palopoli (2011). "Biological network querying techniques: analysis and comparison". *Journal of computational biology : a journal of computational molecular cell biology* **18**, pp. 595–625. DOI: 10.1089/cmb.2009.0144.

Galgamuwa, G. I. R. K., L. K. G. Liyanage, M. P. B. Ekanayake, and B. G. L. T. Samaranayake (2015). "Simplified controller for three wheeled omni directional mobile robot". In: *2015 IEEE 10th International Conference on Industrial and Information Systems (ICIIS)*, pp. 314–319. DOI: 10.1109/ICIINFS.2015.7399030.

García, C. E., D. M. Prett, and M. Morari (1989). "Model predictive control: theory and practice—a survey". *Automatica* **25**:3, pp. 335–348. ISSN: 0005-1098. DOI: https://doi.org/10.1016/0005-1098(89)90002-2. URL: https://www.sciencedirect.com/science/article/pii/0005109889900022.

Gigliotta, O. (2018). "Equal but different: task allocation in homogeneous communicating robots". *Neurocomputing* **272**, pp. 3–9. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2017.05.093. URL: https://www.sciencedirect.com/science/article/pii/S0925231217311177.

Greiff (2017). *Modelling and Control of the Crazyflie Quadrotor for Aggressive and Autonomous Flight by Optical Flow Driven State Estimation*. TFRT-6026. MSc Thesis. Department of Automatic Control, Lund University, Sweden. URL: http://lup.lub.lu.se/student-papers/record/8905295.

Hou, Z. and I. Fantoni (2015). "Distributed leader-follower formation control for multiple quadrotors with weighted topology". In: *2015 10th System of Systems Engineering Conference (SoSE)*, pp. 256–261. DOI: 10.1109/SYSOSE.2015.7151924.

Luukkonen, T. (2011). "Modelling and control of quadcopter". *Independent research project in applied mathematics, Espoo* **22**, p. 22. ISSN: Mat-2.4108. URL: https://sal.aalto.fi/publications/pdf-files/eluu11_public.pdf. (accessed: 15.04.2022).

Matlab (2022). *System identification toolbox*. URL: https://se.mathworks.com/products/sysid.html. (accessed: 10.04.2022).

Nex, F., C. Armenakis, M. Cramer, D. Cucci, M. Gerke, E. Honkavaara, A. Kukko, C. Persello, and J. Skaloud (2022). "UAV in the advent of the twenties: where we stand and what is next". *ISPRS Journal of Photogrammetry and Remote Sensing* **184**, pp. 215–242. ISSN: 0924-2716. DOI: https://doi.org/10.1016/j.isprsjprs.2021.12.006. URL: https://www.sciencedirect.com/science/article/pii/S0924271621003282.

Olfati-Saber, R. (2006). "Flocking for multi-agent dynamic systems: algorithms and theory". *IEEE Transactions on Automatic Control* **51**:3, pp. 401–420. DOI: 10.1109/TAC.2005.864190.

Omodolor, S. O. (2022a). *Code implementation of the swarm controller*. URL: https://gitlab.control.lth.se/stevedanomodolor/formation_coordination_uav_ugv.git. (accessed: 10.05.2022).

Omodolor, S. O. (2022b). *Velocity controller for the three-wheeled omniwheel robot*. URL: https://gitlab.control.lth.se/stevedanomodolor/omniwheels_controller.git. (accessed: 10.05.2022).

Preiss, J. A., W. Hönig, G. S. Sukhatme, and N. Ayanian (2017). "Crazyswarm: A large nano-quadcopter swarm". In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, Marina Bay Sands, Singapore, pp. 3299–3304. URL: https://doi.org/10.1109/ICRA.2017.7989376.

Rashid, A. and B. Issa (2019). "A survey of multi-mobile robot formation control". *International Journal of Computer Applications* **181**, pp. 12–16. DOI: 10.5120/ijca2019918651.

Reynolds, C. (1987). "Flocks, herds, and schools: a distributed behavioral model". *ACM SIGGRAPH Computer Graphics* **21**, pp. 25–34. DOI: 10.1145/280811.281008.

Saif, O. (2016). *Reactive navigation of a fleet of drones in interaction*. PhD Thesis 2016COMP2269. Université de Technologie de Compiègne. URL: https://tel.archives-ouvertes.fr/tel-01459859. (accessed: 30.04.2022).

Saif, O., I. Fantoni, and A. Zavala (2019). "Distributed integral control of multiple uavs, precise flocking and navigation". *IET Control Theory Applications* **13**. DOI: 10.1049/iet-cta.2018.5684.

Shi, H., L. Wang, and T. Chu (2005). "Virtual leader approach to coordinated control of multiple mobile agents with asymmetric interactions". In: *Proceedings of the 44th IEEE Conference on Decision and Control*, pp. 6250–6255. DOI: 10.1109/CDC.2005.1583163.

Soria, E., F. Schiano, and D. Floreano (2021). "Predictive control of aerial swarms in cluttered environments". *Nature Machine Intelligence* **3**. DOI: 10.1038/s42256-021-00341-y.

Stanford Artificial Intelligence Laboratory et al. (23, 2018). *Robotic operating system*. Version ROS Melodic Morenia. URL: https://www.ros.org. (accessed: 28.04.2022).

Sun, Z. (2016). *Cooperative Coordination and Formation Control for Multi-agent Systems*. PhD Thesis. Research School of Engineering, College of Engineering and Computer Science, The Australian National University. DOI: 10.25911/5d74e8717068e.

Taffanel, A., B. Rousselot, J. Danielsson, K. McGuire, K. Richardsson, M. Eliasson, T. Antonsson, and W. Hönig (2021). "Lighthouse positioning system: dataset, accuracy, and precision for UAV research". *CoRR* **abs/2104.11523**. arXiv: 2104.11523. URL: https://arxiv.org/abs/2104.11523.

Appendices

A

Calculation of Position and Velocity of β-agent

The β-agent is an agent that represents an obstacle which is used in the obstacle term in Equation 3.30. The position ($\hat{q}_{i,k}$) and velocity ($\hat{p}_{i,k}$) of a β-agent of an obstacle O_k represented by a wall or a sphere, see Figure 3.3, of an α-agents (q_i, p_i) [Olfati-Saber, 2006], are given as follows:

- For obstacles with hyperplane boundaries with unit normal a_k that pass through the point y_k, see Figure 3.3, we can compute the position and the velocity of the β-agent as:

$$\hat{q}_{i,k} = Pq_i + (I-P)y_k \hat{p}_{i,k} = Pp_i \tag{A.1}$$

  where P is the projection matrix and we can calculate it using the following equation:

$$P = I - a_k a_k^T \tag{A.2}$$

- For obstacle k with spherical shape with radius R_k that are centered at position y_k, we can calculate the position and velocity of the β-agent i as

$$\hat{q}_{i,k} = \mu q_i + (1-\mu)y_k, \hat{p}_{i,k} = \mu P p_i \tag{A.3}$$

  where μ, a_k and P are calculated as follows:

$$\mu = \frac{R_k}{\|q_k - y_k\|} \tag{A.4}$$

$$a_k = \frac{q_i - y_k}{\|q_i - y_k\|} \tag{A.5}$$

$$P = I - a_k a_k^T \tag{A.6}$$

B

Simulation Results

In this appendix section, we show the results we obtained from the simulation using the parameters in Table 6.1 but applied to three and five drone formations. The goal is to show the effectiveness of the control law applied to a different number of agents. In Appendix B.1, we show the result obtained for a three UAV formation with a static navigation goal and similar experimental parameters but adding orientation. Appendix B.2 shows similar results but applies to five UAVs. In both cases, we can observe that the proposed orientation control law was able to converge to the desired values, see Figures B.7 and B.11.

B.1 Three UAVs

B.1.1 Formation

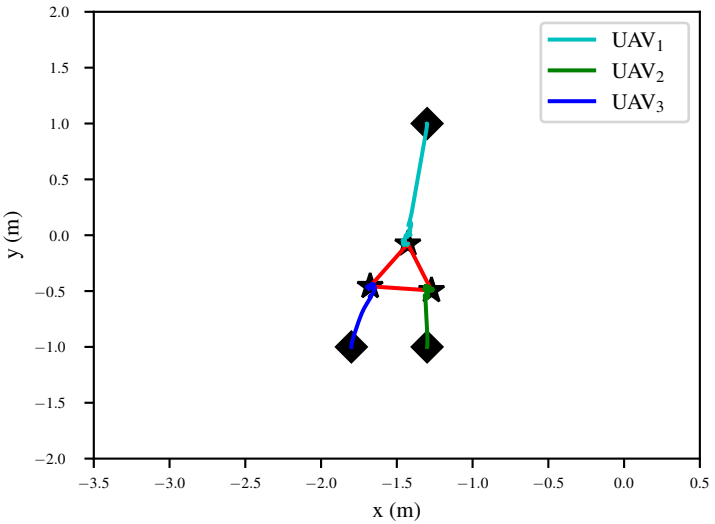

Figure B.1 Path of the three UAVs for the free-space formation

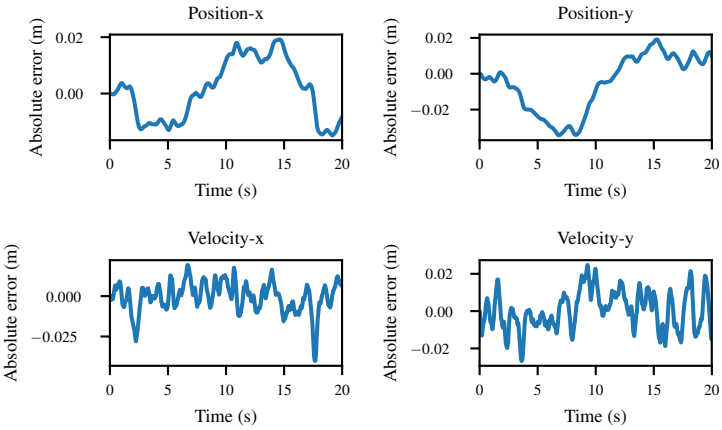

Figure B.3 Absolute error (m) between the navigation goal (position, velocity) and the current centroid for the three UAVs

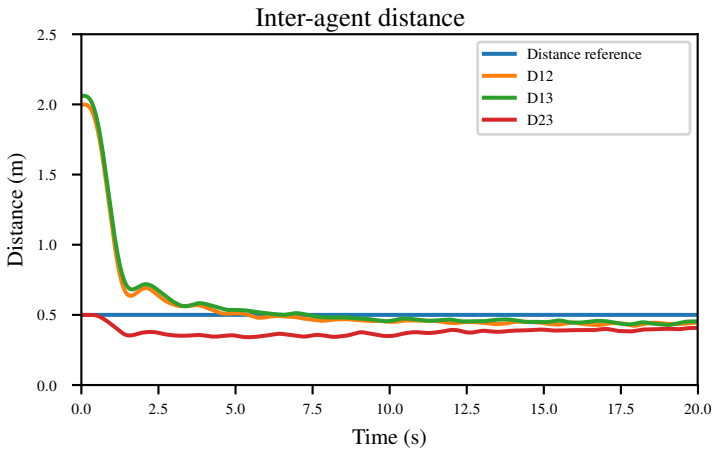

Figure B.2 Inter-agent distance of the three UAVs for the free-space formation formation

B.1.2 Formation with Orientation

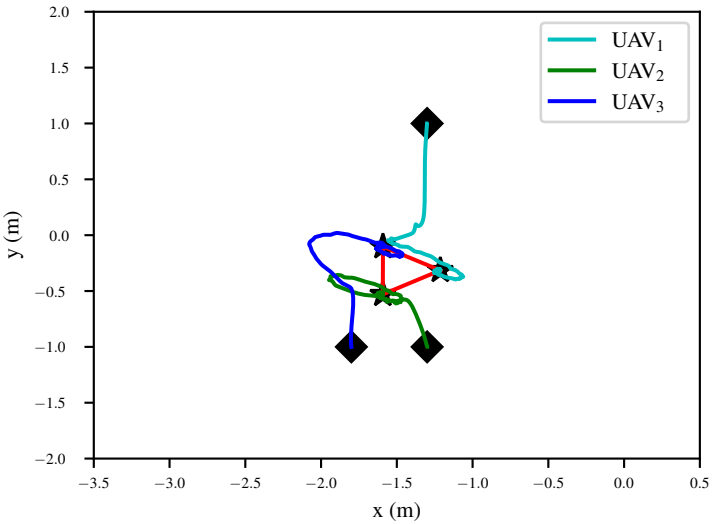

Figure B.4 Path of the three UAVs for the free-space formation with 90°orientation

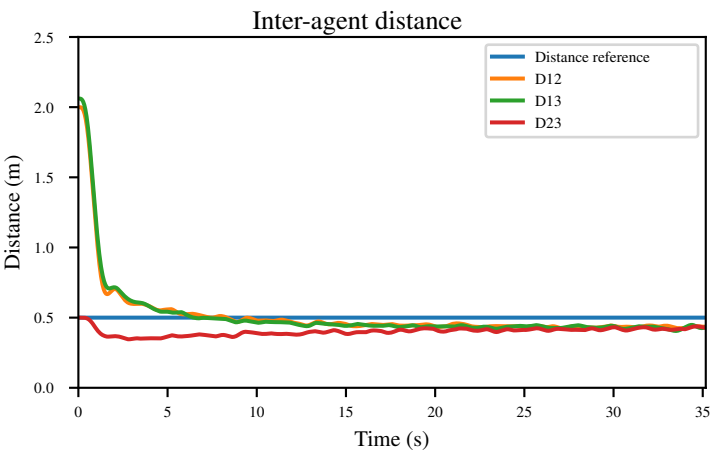

Figure B.5 Inter-agent distance of the three UAVs for the free-space formation

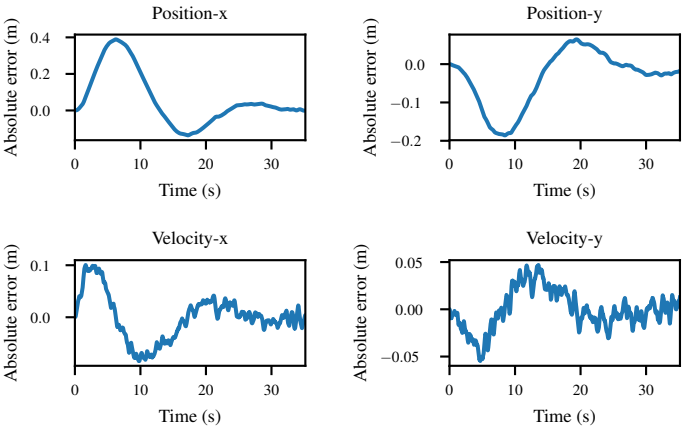

Figure B.6 Absolute error (m) between the navigation goal (position, velocity) and the current centroid for the three UAVs

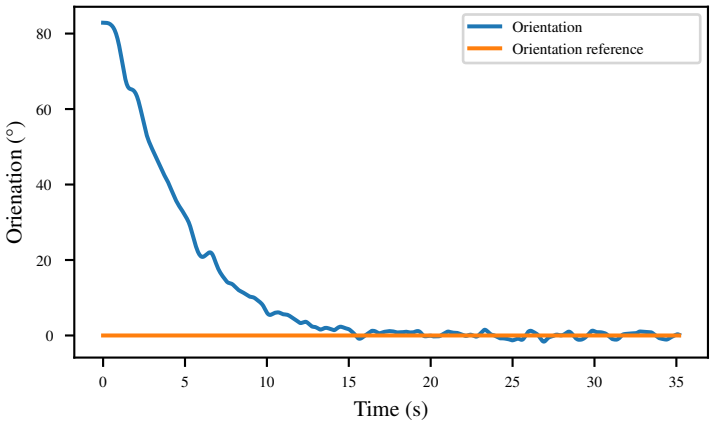

Figure B.7 Evolution of the orientation for the UAVs

B.2 Five UAVs

B.2.1 Formation

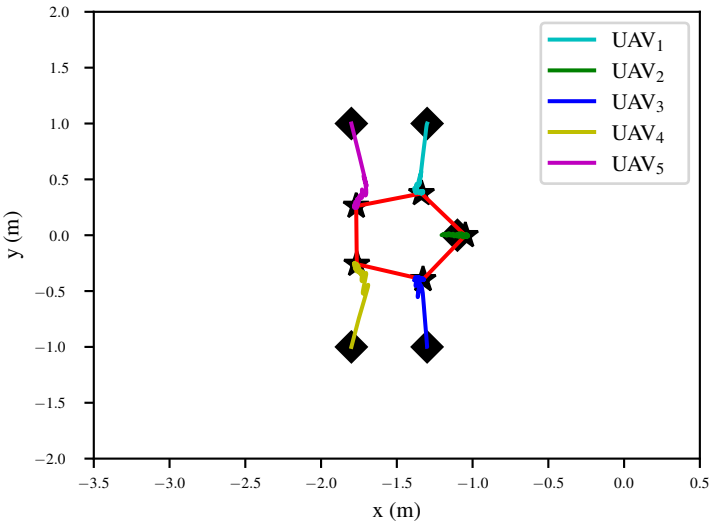

Figure B.8 Path of the five UAVs for the free-space formation

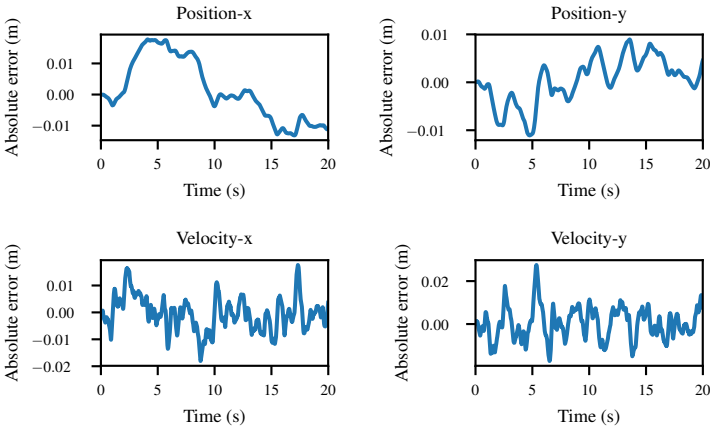

Figure B.10 Absolute error (m) between the navigation goal (position, velocity) and the current centroid for the five UAVs

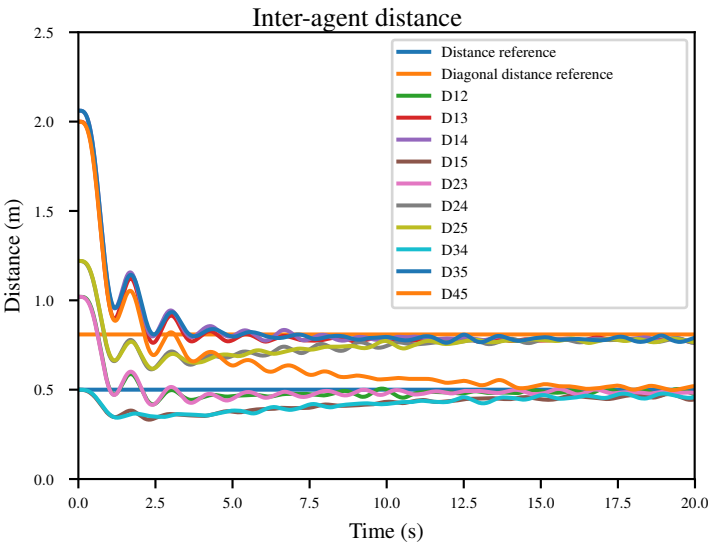

Figure B.9 Inter-agent distance of the five UAVs for the free-space formation

B.2.2 Formation with Orientation

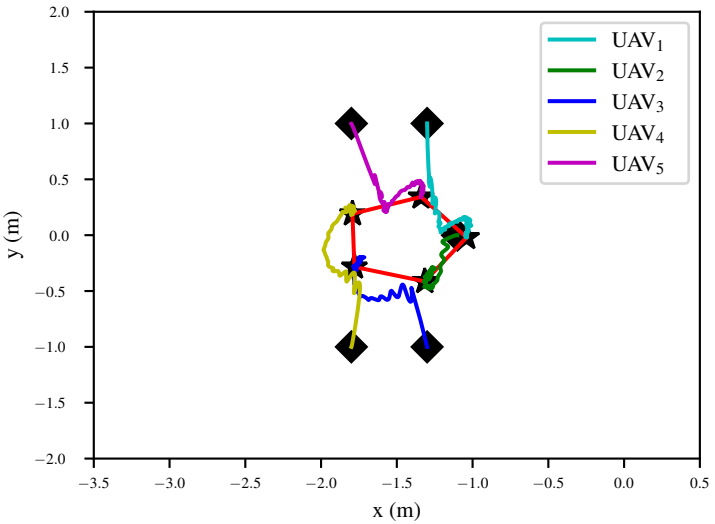

Figure B.11 Path of the five UAVs for the free-space formation with 90°orientation in (B.2.2)

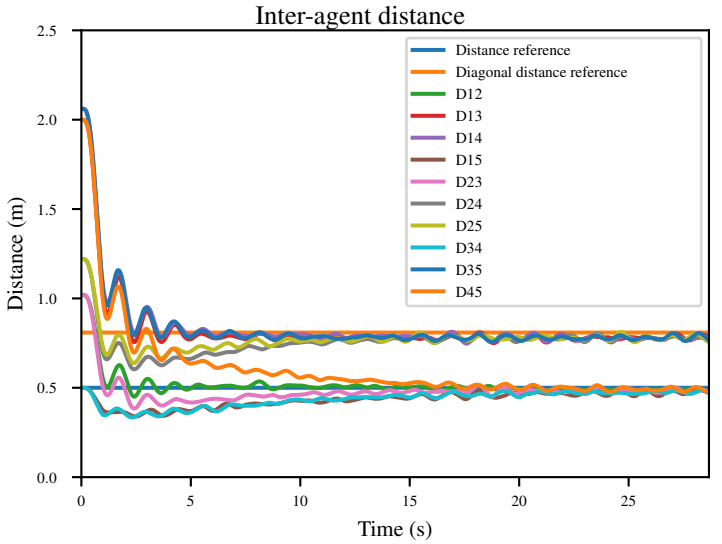

Figure B.12 Inter-agent distance of the five UAVs for the free-space formation

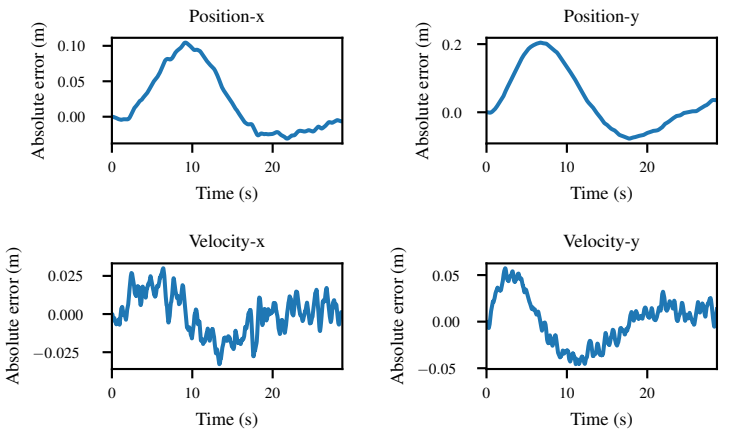

Figure B.13 Absolute error (m) between the navigation goal (position, velocity) and the current centroid for the five UAVs

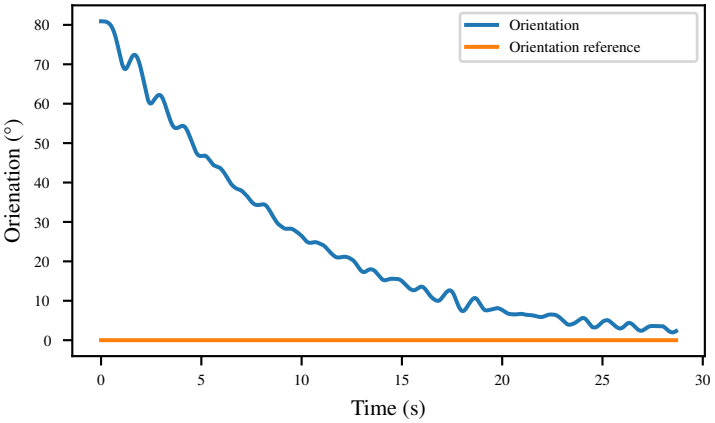

Figure B.14 Evolution of the orientation for the five UAVs

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER'S THESIS

Date of issue
August 2022

Document Number
TFRT-6181

Author(s)
Stevedan Ogochukwu Omodolor

Supervisor
Björn Olofsson, Dept. of Automatic Control, Lund University, Sweden
Anders Robertsson, Dept. of Automatic Control, Lund University, Sweden
Tore Hägglund, Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle

Distance and orientation-based formation control of UAVs and coordination with UGVs

Abstract

Nowadays, research in autonomous drones has increased thanks to the advancement of drone technology. Nevertheless, there are still limitations when performing specific missions due to flight duration, computational load and mission complexity. This thesis investigates ways to solve this problem by taking advantage of multiple UAVs and UGVs. This thesis aims to implement and evaluate strategies for formation and coordination of multiple UAVs and UGVs. Firstly, we present a brief review of state of the art on formation and flocking control, further specifying the advantages and limitations of each approach.

Secondly, we use a behaviour-based approach to obtain multi-UAV formation control. We adapt the algorithm to apply it to a single integrator system model to control the UGVs' formation. We then propose an extension to the original algorithm to consider orientation during formation and a leader-follower strategy to coordinate the interaction between the units using a cluster-based approach.

Finally, we tested our proposed control laws in simulation and in experiments. The simulations were done in Matlab, while the real-implementation experiments were performed using the Crazyflie quadcopters and three-wheeled omniwheel robots.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-108

Recipient's notes

Security classification