
# Bioinspired Algorithm of Robot Network for Assistance in Search and Rescue Operations

**Gustavo Andres Cardona Calderon**

Universidad Nacional de Colombia
Engineering Faculty, Electronic and Electric Engineering Department
Bogotá, Colombia
2023

# Bioinspired Algorithm of Robot Network for Assistance in Search and Rescue Operations

**Gustavo Andres Cardona Calderon**

Thesis presented as partial requirement for applying to the title of:
**Magister en Ingeniería - Automatización Industrial**

Director:
Ph.D.Eduardo Alirio Mojica Nava

Research Line:
Control and Robotics
Research Group:
PAAS - Processing, Acquisition, and Analysis of Signals

Universidad Nacional de Colombia
Engineering Faculty, Electronic and Electric Engineering Department
Bogotá, Colombia
2023

## Dedication

To my little sister, who was the most precious support in my life.

# Acknowledgements

I would like to thank the Universidad Nacional de Colombia for the opportunity and the financial support that made this project possible through the Teacher Assistant in Electrical Machines for the Undergraduate Mechanical Engineering program. To Professor Eduardo Mojica-Nava for advising me, for his patience, and support in the whole process as an undergraduate and graduate student. I thank my friends for encouraging me to develop this and many other projects, especially Duvan Tellez, David Yanguas, Felipe Arevalo, Nestor Ospina, and Juan David Pabon, for their great discussion on many ideas that end up being papers. Last but not least, I would like to thank my mother and father for supporting me throughout my studies and life. Thanks to my uncle Juan Calderon, this and many other projects in my life could not have been done without his guidance and support.

# Resumen

**Algoritmo bioinspirado para redes de robots en la asistencia en operaciones de busqueda y rescate**

Esta tesis propone un algoritmo bioinspirado para redes de robots que asisten en las operaciones de escenarios de búsqueda y rescate. Consideramos a las hormigas como animales sociales para estudiar y abstraer comportamientos que pueden ser útiles en el marco de la búsqueda y rescate mediante robots. Consideramos tres temas principales para abordar cuando se utilizan robots para ayudar a los rescatistas.

Primero, la exploración y mapeo de las zonas de desastre. Para esto, consideramos los mecanismos e interacciones de las hormigas para explorar su entorno, buscar comida, evitar depredadores y explorar mejores lugares para establecer un nido. Luego, desplegamos robots para explorar el entorno y disuadimos a los robots de ingresar a regiones que otros robots han explorado usando feromonas como marcadores para los robots. También abstraemos la aleatoriedad que usan las hormigas para explorar e implementar un algoritmo Q-learning que permite a los robots explorar regiones no visitadas.

En segundo lugar, la navegación y detección de víctimas. Una vez que se ha explorado el entorno, usamos las reglas de Reynolds para permitir que la navegación de los robots cree cohesión, atracción hacia los objetivos y repulsión hacia los obstáculos y las colisiones entre agentes. Luego, usamos una red neuronal para determinar si lo que detectan los robots es una víctima. Por último, utilizamos un enfoque de consenso para clasificar a las víctimas o no víctimas en función de la información distribuida.

Por último, las hormigas han sido famosas por llevar cargas que superan su tamaño y capacidad de carga al cooperar. Consideramos quadrotors para transportar cargas de manera cooperativa que pueden ser suministros médicos o víctimas en búsqueda y rescate.

**Palabras clave: 1) Robots heterogéneos, 2) redes de robots, 3) Control en tiempo real, 4) Búsqueda y rescate**.

# Abstract

**Bioinspired Algorithm of Robot Network for Assistance in Search and Rescue Operations**

This thesis proposes a bio-inspired algorithm for robot networks assisting in the operations of search and rescue scenarios. We consider ants as social animals to study and abstract behaviors that can be useful in the framework of search and rescue using robots. We consider three main topics to address when using robots to assist rescuers.

First, the exploration and mapping of the disaster zones. For this, we consider the mechanisms and interactions of ants to explore their environment, look for food, avoid predators,

and explore better places to establish a nest. Then, we deploy robots to explore the environment and discourage robots from entering regions other robots have explored using pheromones as markers for the robots. We also abstract the randomness ants use to explore and implement a Q-learning algorithm that allows robots to explore unvisited regions.
Second, the navigation and victim detection. Once the environment has been explored, we use Reynolds rules to allow the navigation of robots to create cohesion, attraction to target goals, and repulsion to obstacles and inter-agent collisions. Then, we use a neural network to determine whether what robots are detecting is a victim. Lastly, we use a consensus-like approach to classify victims or no victims based on distributed information.
Lastly, ants have been famous for carrying loads that surpass their size and payload capacity by cooperating. We consider quadrotors to carry loads cooperatively that can be medical supplies or victims in search and rescue.
**Keywords: 1) Heterogeneous Robots, 2)Robots Networks, 3) Real Time Control, 4) Search and Rescue**.

# Contents

**Acknowledgements** iv

**Resume** v

**1 Introduction** 2
1.1 Motivation . . . 2
1.1.1 Exploration and Mapping . . . 3
1.1.2 Navigation and Victim Detection . . . 3
1.1.3 Cooperative Load Transportation . . . 4
1.2 Summary of Achievements . . . 5
1.2.1 Organization . . . 6

**2 Exploration and Mapping of Unknown Non-Convex Spaces.** 7
2.1 Ant Exploration and Navigation Behaviors . . . 7
2.2 Exploration Using Reinforcement Learning . . . 9
2.2.1 Agent Model . . . 9
2.2.2 Reinforcement Learning Algorithm . . . 11
2.3 Exploration Using Voronoi Tessellation . . . 15
2.3.1 Fundamental Concepts on Graph Theory . . . 15
2.3.2 System Modeling . . . 17
2.3.3 Exploration Algorithm . . . 19
2.4 Fundamental Concepts on Hybrid Systems . . . 24

**3 Navigation and Victim Detection** 27
3.1 Ant Colony Searching for a New Nest Behavior . . . 27
3.2 Navigation in known non-convex spaces . . . 28
3.2.1 Swarm Navigation . . . 28
3.2.2 Sub-Swarm Generation . . . 28
3.2.3 Formation Control . . . 29
3.3 Victim Detection . . . 31
3.3.1 Fundamental Concepts on Neural Networks . . . 31
3.3.2 Distributed Victim Estimation Consensus . . . 31

**4 Cooperative Load Transportation** **36**
4.1 Ant Cooperative Load Transportation . . . 36
4.2 Geometric Control . . . 37
4.2.1 Agent Model . . . 38
4.2.2 Geometric Control Algorithm Design . . . 41
4.3 Adaptive Control . . . 43
4.3.1 Agent Model . . . 43
4.3.2 Adaptive $\sigma$-Modification for State Synchronization . . . 44
4.4 Bilevel Adaptive Control . . . 50
4.4.1 Problem Statement . . . 50
4.4.2 Load Transporting Problem . . . 51
4.4.3 Model Dynamics . . . 51
4.4.4 Control Design . . . 53

**5 Simulation and Results** **59**
5.1 Exploration and Navigation . . . 59
5.1.1 Reinforcement Learning . . . 59
5.1.2 Voronoi Tessellation . . . 66
5.2 Victim Detection Algorithm . . . 72
5.2.1 Swarm Navigation . . . 72
5.2.2 Sub-Swarm Formation Control . . . 80
5.2.3 Victim Detection Using Neuronal Networks . . . 81
5.2.4 Estimation Consensus . . . 83
5.3 Cooperative Load Transportation . . . 84
5.3.1 Geometric Control . . . 84
5.3.2 Adaptive Control . . . 92
5.3.3 Bilevel Adaptive Control . . . 94
5.4 Hybrid Systems . . . 96
5.4.1 High-Level Planning . . . 97

**6 Conclusions and Future Work** **101**
6.1 Future Work . . . 102

**Bibliography** **103**

# List of Figures


2.1 Robot axis versus inertial reference frame. . . . . 9
2.2 Parameter definition of standard wheel directionless in differential robots. . . 10
2.3 Decrease in the learning factor over time. . . . . 14
2.4 Example of how to model multi-agent systems using Graph Theory. . . . . 17
2.5 Heterogeneous multi-robot system, modeled as a Digraph. . . . . 18
2.6 Four Neighbor Flooding Distance (Blue Line), Eight Neighbor Flooding Distance (Orange Line), and Euclidean Distance (Green Dashed Line) . . . . . 19
2.7 Radius sensed, Line of sight, and are sensed by a single robot in a non-convex space. . . . . 20
2.8 Example of how the communication graph is formed. . . . . 21
2.9 Finite State Machine Diagram . . . . . 25
2.10 Finite State Machine Diagram with guards . . . . . 26

3.1 Sub-swarm generation. . . . . 29
3.2 Consensus around of a circle. . . . . 34
3.3 Digraph scheme. . . . . 35
3.4 Additive Gaussian noise applied to measurements in sensors. . . . . 35
3.5 Sensor network converging due to the consensus applied. . . . . 35

4.1 Quadrotor model: in the robot reference frame to the inertial reference frame. 39
4.2 Cooperative transportation: carrying a load by cooperating with a group of quadrotors. . . . . 40
4.3 Tension modeled as a Spring-Damping system. . . . . 44
4.4 Control Scheme. . . . . 45
4.5 Leader-follower communication graph. . . . . 45
4.6 Multiple quadrotors transporting a cable-suspended load. . . . . 50

5.1 The unknown simulation environment for the robot. . . . . 60
5.2 Pioneer 3-DX Robot. . . . . 61
5.3 Generated path by a robot over time in the learning process. . . . . 62
5.4 Robot crashes through the learning episodes. . . . . 63
5.5 Q values evolution through the learning process. . . . . 64
5.6 Explore and exploit policy through the learning process. . . . . 65
5.7 Global Map sequence, Gotten by the 5 robots. . . . . 66

5.8 Exploration Sequence - Robot 1. . . . . 67
5.9 Voronoi Cells Evolution - Robot 1. . . . . 68
5.10 Exploration Sequence - Robot 2. . . . . 69
5.14 Exploration Sequence - Robot 4. . . . . 69
5.11 Voronoi Cells Evolution - Robot 2. . . . . 70
5.12 Exploration Sequence - Robot 3. . . . . 71
5.17 Voronoi Cells Evolution - Robot 5. . . . . 71
5.13 Voronoi Cells Evolution - Robot 3. . . . . 72
5.15 Voronoi Cells Evolution - Robot 4. . . . . 73
5.16 Exploration Sequence - Robot 5. . . . . 74
5.18 Simulation in a realistic environment and robot platforms using V-Rep. . . . 75
5.19 Simulation of robots avoiding small obstacles. . . . . 76
5.20 Simulation of robots avoiding large obstacles. . . . . 76
5.21 Simulation of robots avoiding multi-obstacle scenario. . . . . 77
5.22 Victim localization in convex space. . . . . 77
5.23 Navigation and victims localization in an environment with the presence of obstacles. . . . . 78
5.24 Convergence analysis with variation in the number of agents and random initial values. . . . . 79
5.25 Formation control applied to the three agents sub-swarm generated by localization of a victim. . . . . 80
5.26 The covered area by a different number of drones with several relation values between $r$ and $d$. . . . . 81
5.27 Formation control applied to 6 agents sub-swarm generated by localization of a victim. . . . . 82
5.28 Formation control applied to the three agents sub-swarm generated by localization of a victim. . . . . 83
5.29 Victim Detection. . . . . 84
5.30 Digraph scheme. . . . . 84
5.31 Consensus Estimation based on single Quadrotor victim detection. . . . . 85
5.32 Consensus Estimation vs Quadrotors' sensing variability. . . . . 85
5.33 Drones estimation and Consensus Estimation in a fire environment. . . . . 86
5.34 Drones estimation, Consensus Estimation, and Max Consensus estimation in a fire environment. . . . . 87
5.35 Cooperative transportation: Simulation using conventional geometric control. 88
5.36 Cooperative transportation: Simulation considering improvements of positioning and minimization of effort. . . . . 89
5.37 Cooperative transportation: Position evolution in x,y, and z using conventional geometric control. . . . . 90

5.38 Cooperative transportation: Position evolution in x,y, and z considering improvements of positioning and minimization of effort. . . . . 90
5.39 Cooperative transportation: Control Signals, using conventional geometric control. . . . . 91
5.40 Cooperative transportation: Control Signals, considering improvements of positioning and minimization of effort. . . . . 92
5.41 DMRAC in $z_W$ states synchronization of multiple quadrotors. . . . . 93
5.42 States synchronization in $x_W$ considering fixed $\delta$'s. . . . . 94
5.43 Full simulation top view. . . . . 95
5.44 Cooperative transportation using multiple quadrotors simulation. . . . . 96
5.45 Multi-layer graph: Communication graph, and physical interaction graph. . . 97
5.46 Constant trajectory reference for the load and response of the quadrotors in the $x - axis$. . . . . 98
5.47 Sinusoidal trajectory reference for the load and response of the quadrotors in the $x - axis$. . . . . 99

# List of Tables


5.1 Convergence data depending on the number of agents. . . . . 76
5.2 Percentage of the covered area according to different rate values between $r$ and $d$. . . . . 79
5.3 Percentage of the covered area according to different swarm sizes and $r = d/4$. 80
5.4 Simulation Parameters . . . . . 87

# 1 Introduction

## 1.1. Motivation

In recent years, search and rescue operations have become increasingly critical in response to natural disasters, accidents, and other emergencies. Efficiently exploring and navigating complex environments while effectively coordinating tasks is critical for successful rescue missions [65]. To address these challenges, researchers have turned to nature for inspiration, seeking innovative solutions that can enhance the capabilities of robotic systems [30]. One intriguing source of inspiration lies within the remarkable behavior exhibited by ants, particularly in their exploration and navigation strategies [79]. Ant colonies demonstrate unparalleled efficiency in foraging and transporting food resources, overcoming obstacles, and adapting to changing environments [39]. These tiny creatures achieve remarkable feats through the collective actions of many individuals, working in unison to achieve a common goal [41].

This thesis aims to develop a bioinspired algorithm that harnesses the principles observed in ant colonies to design a network of robotic agents capable of assisting search and rescue operations. By studying the decentralized decision-making processes, division of labor, and coordination mechanisms observed in ant behavior, we aim to create a sophisticated system that emulates their efficient exploration, navigation, and load-carrying capabilities. One particularly intriguing aspect of ants' behavior is their ability to surpass their individual capacity when faced with large or heavy objects. Through collective transport, ants effectively distribute the load across multiple individuals, enabling them to overcome obstacles that would otherwise be insurmountable for a single ant. This adaptive and resilient approach to problem-solving holds great potential for enhancing the capabilities of robotic networks operating in challenging and unpredictable environments.

The proposed bioinspired algorithm will draw upon the principles of swarm intelligence, where individual robots communicate, cooperate, and coordinate their actions to achieve collective goals. By emulating the decentralized decision-making processes observed in ant colonies, our algorithm will enable the robotic network to dynamically adapt its behavior to changing conditions, optimize resource allocation, and efficiently explore the search area. Through this research, we aim to advance search and rescue operations robotic systems by integrating the natural intelligence observed in ants' behavior with state-of-the-art robotics technology.

### 1.1.1. Exploration and Mapping

In recent years, using multirobot systems has emerged as a promising approach to tackle the challenges associated with exploration and navigation in search and rescue missions. Deploying a network of autonomous robots that can work collaboratively and adaptively in complex and hazardous environments offers significant advantages over traditional single-robot approaches [58]. One notable area of research focuses on decentralized coordination algorithms for multirobot systems. Traditional centralized approaches, where a central controller directs the actions of all robots, often suffer from scalability issues and vulnerability to single points of failure. Decentralized coordination algorithms aim to distribute decision-making among individual robots, enabling them to work autonomously while achieving coordinated behavior. Various techniques, such as consensus algorithms, task allocation mechanisms, and swarm intelligence-inspired approaches, have been explored to enable effective coordination and collaboration among the robots [64].

Furthermore, localization and mapping are critical for exploration and navigation in search and rescue missions. The ability of robots to accurately perceive and map their environment in real time is essential for efficient exploration and effective decision-making. Simultaneous Localization and Mapping (SLAM) techniques [2], which involve constructing a map of the environment while simultaneously estimating the robot's position within that map, have been widely investigated for multirobot systems. Collaborative SLAM algorithms that enable robots to share information, fuse sensor data, and build a consistent global map have shown promising results in enhancing exploration capabilities [75]. In this thesis, we consider a decentralized algorithm that uses Voronoi tessellation to assign regions of exploration to robots, and we consider the use of markers to indicate regions that have already been explored and discourage robots from revisiting the region.

Moreover, advancements in machine learning and artificial intelligence have significantly improved the performance and capabilities of multirobot systems in search and rescue operation [8]. For example, reinforcement learning techniques have enabled robots to learn optimal exploration and navigation policies through interaction with the environment. In addition, deep learning approaches have been leveraged for perception tasks such as object recognition and semantic mapping, enabling robots to interpret and understand the environment more effectively. In this thesis, we also explore this type of tool for exploration by adapting a Q-learning approach to allow exploration of the environment while also learning to avoid collisions with obstacles.

### 1.1.2. Navigation and Victim Detection

Robot swarms, consisting of many autonomous robots collaborating and coordinating their actions, have shown great potential in search and rescue scenarios [76]. The ability of robot swarms to efficiently explore and navigate complex environments and detect and locate victims has garnered significant attention from researchers.

Navigation is critical to robot swarm operations in search and rescue missions. Swarm navigation algorithms aim to enable the robots to efficiently explore the environment while avoiding obstacles and ensuring adequate coverage [34]. Various approaches have been studied, including potential fields, artificial potential-based methods, and behavior-based techniques [9]. In this thesis, we have developed an algorithm that allows a swarm of robots to navigate a previously explored environment while avoiding obstacles. Furthermore, we use the potential function to generate repulsion behaviors towards obstacles and attraction to detected victims.

Victim detection and localization are fundamental tasks in search and rescue scenarios. Robot swarms offer advantages, as they can cover larger areas and search in parallel, increasing the likelihood of locating victims quickly [74]. Vision-based techniques, such as object recognition and tracking algorithms, have been employed to detect and identify victims in the swarm's surroundings. Thermal and infrared sensors also detect body heat signatures, enabling the swarm to locate victims in low-visibility conditions. In this thesis, we consider cameras on quadrotors to perform the detection of victims in the environment. Machine learning and artificial intelligence techniques have enhanced victim detection and localization. We use a neuronal network to perform the classification and detection of victims [69].

Communication and coordination within the swarm are crucial for efficient victim detection and navigation. Swarm robots typically communicate with each other through local interactions and exchange information about the environment, victim detections, and navigation plans. Communication protocols and algorithms, such as consensus and distributed control mechanisms, have been developed to facilitate information sharing and coordination among the swarm robots.

### 1.1.3. Cooperative Load Transportation

Quadrotors, with their maneuverability, stability, and ability to hover, are well-suited for carrying and transporting payloads. In cooperative load transportation, multiple quadrotors lift and transport objects too heavy or oversized for a single quadrotor. This approach enables the transportation of heavier payloads, enhances robustness, and provides redundancy in case of failures [78].

Flight control algorithms play a crucial role in cooperative load transportation. These algorithms coordinate the quadrotors' motions and ensure stability while carrying the load. Control strategies, such as PID (Proportional-Integral-Derivative) controllers, model predictive control, and optimal control, have precisely controlled the quadrotors' positions and orientations during load transportation [77]. In addition, adaptive control techniques have also been investigated to handle uncertainties and changes in the payload characteristics. In this thesis, we develop two approaches: one that uses geometric control to allow quadrotors to follow a trajectory while carrying a load. Then, we consider that the object's mass can be unknown and that unknown forces come from the interaction of quadrotors pulling the

load. To overcome this, we propose adaptive controllers that adapt the inputs to account for uncertainties and disturbances.

## 1.2. Summary of Achievements

**Journals papers derived from this thesis**

1. Cardona, G. A., Ramirez-Rugeles, J., Mojica-Nava, E., & Calderon, J. M. (2021). Visual victim detection and quadrotor-swarm coordination control in search and rescue environment. International Journal of Electrical and Computer Engineering, 11(3), 2079.

2. Cardona, G. A., & Calderon, J. M. (2019). Robot swarm navigation and victim detection using rendezvous consensus in search and rescue operations. Applied Sciences, 9(8), 1702.

**Conferences papers derived from this thesis**

1. Cardona, G. A., D. Tellez-Castro, J. Calderon and E. Mojica-Nava,."Adaptive Multi-Quadrotor Control for Cooperative Transportation of a Cable-Suspended Load,"2021 European Control Conference (ECC), 2021, pp. 696-701, IEEE.

2. Cardona, G. A., Arevalo-Castiblanco, M., Tellez-Castro, D., Calderon, J., & Mojica-Nava, E. (2021, May). Robust adaptive synchronization of interconnected heterogeneous quadrotors transporting a cable-suspended load. In 2021 IEEE International Conference on Robotics and Automation (ICRA) (pp. 31-37). IEEE.

3. Arevalo-Castiblanco, M. F., Tellez-Castro, D., Cardona, G. A., & Mojica-Nava, E. (2019, December). An adaptive optimal control modification with input uncertainty for unknown heterogeneous agents synchronization. In 2019 IEEE 58th Conference on Decision and Control (CDC) (pp. 8242-8247). IEEE.

4. Cardona G. A., D. Yanguas-Rojas, M. F. Arevalo-Castiblanco, and E. Mojica-Nava,."Ant-Based Multi-Robot Exploration in Non-Convex Space without Global-Connectivity Constraints,"2019 18th European Control Conference (ECC), 2019, pp. 2065-2070, IEEE

5. Cardona, G. A., Bravo, C., Quesada, W., Ruiz, D., Obeng, M., Wu, X., & Calderon, J. M. (2019,April). Autonomous navigation for exploration of unknown environments and collision avoidance in mobile robots using reinforcement learning. In 2019 SoutheastCon (pp. 1-7). IEEE.

6. Cardona, G. A., Tellez-Castro, D., & Mojica-Nava, E. (2019). Cooperative transportation of a cable-suspended load by multiple quadrotors. IFAC-PapersOnLine, 52(20), 145-150.

### 1.2.1. Organization

The remainder of this thesis is organized as follows.

Chapter 2 tackles the problem of exploration and mapping using multi-robot systems. First, using the Voronoi tessellation approach and potential functions presented in [80] and [23]. Then, we consider a Q-learning algorithm for exploring unknown environments while avoiding obstacles presented in [11].

Chapter 3 deals with the problem of navigation and victim detection of a previously explored environment. We consider potential functions for generating attractive and repulsive forces for objectives and avoiding collisions. This section was presented in [56] and [13]. Then, for doing a distributed victim detection, we use neuronal networks and a consensus approach to determine whether what robots detect are victims, presented in [18].

Chapter 4 Considers the problem of cooperative load transportation using quadrotors attached through cables to a load. We consider geometric controller to track a reference trajectory and also adaptive control to overcome uncertainties and external disturbances presented in the following papers [12], [1], and [20].

Chapter 5 presents the conclusions and future research directions.

# 2 Exploration and Mapping of Unknown Non-Convex Spaces.

Exploration and navigation tasks are the first assignments a robot assisting in SAR operations can improve, like reducing the time required to cover a huge space while identifying desired targets in a terrain. Exploration, as its name suggests, implies acquiring new information about the environment, allowing an agent to move within the space as needed.

## 2.1. Ant Exploration and Navigation Behaviors

Social foraging behaviors have been described in many species, including birds, fish, mammals, and insects, as reviewed in [42]. In such species, the information communicated between agents and the information obtained by the group can modify the predictable behavior of each individual. In the same way, collective behavior results from the accumulation of local interactions and constraints from the environment. Although several species behave collectively, ants present one of the most interesting and well-studied approaches that have been replicated and emulated in many applications in today's world, such as manufacturing, process optimization, networks routing, ant colony optimization [5], and robotics among others. It is worth mentioning that although the ants' behavior has been widely studied, there are still many types of ants whose behaviors have not been researched yet or several behaviors that have not been emulated to any application. Many researchers want to collect more information and gather more scientists to collect more information on different colonies and kinds of ants that might inspire the scientific community to solve or tackle today's challenges.
Despite the existence of many ant species, almost all of them agree on nests in cavities abandoned made by insects, in dead wood, or in broken, either live or dead trees. Trees used for nests include Ficus, Acacia, Ipomoea wolcottiana, Guapira macrocarpa, and Guazuma ulmifolia. However, it is important to notice that there are differences depending on the type of ants, causing how they navigate, explore and behave in the environment to vary according to their resources. Collective search, as explained in [26], is a task that involves both exploitation and exploration, which can be interpreted as a negotiation between searching thoroughly and covering as much area as possible, which depends on the number of scouters. As is known, ant colonies operate without central control or communication. Consequently, they can only acquire local information, mostly chemical and tactile, so they work collectively to find resources and monitor the nest environment. The goal of the collective search

is to find resources by the movement around of independent individuals and then adjust using local information and interactions to guarantee the area has been covered by at least one agent so that there is something to find one agent will encounter it. For instance, *C. gonoiodontus* species tend to travel just for one trail and are not likely to switch to another because this species usually lives in dry environments. In contrast, harvesters' ants always switch trails and create new paths because their source food is dynamic; the seeds can be moved by the wind or by the movement of other animals. To start differentiating among ant species the *Cephalotes goniodontus* live in the dry forests like the ones located in western Mexico, where ants collect plant-derived food and insect exudates. In [45] is shown how each path made by ants is traveled by different groups of ants. This makes foraging activity resilient. When a path turns impassable, a group of ants can create a new different path. This particular species is also categorized as Polydomous, a colony with at least two spatially separated but socially connected nests. At least 166 species of ants in 49 genera have colonies that are Polydomous [32]. According to [25], information sharing can be communicated between individuals increasing the exploitation of known resource locations in socially foraging Polydomous species ants. Nonetheless, exploitation alone does not guarantee success unless balanced with exploring new locations with resources. Studies in exploration versus exploitation predict that the time spend in the exploration process seeking new resources directly depends on the quality of actual food and quantity of food in known locations [73].

On the other hand, arboreal ants travel only through trees and create links between the nest and a possible food source. The usual habitat of this type of ant lies in the tangled canopy, where the trails can be easily destroyed by rain, other animal movements, or wind. In [24], is shown an algorithm capable of, in a distributed way, reconstructing or finding alternative paths when the links have broken for any reason. To succeed in finding the best path to food resources and avoiding dead-ends, they must generate as many paths as possible and then, through pheromones, converge to one or a few of them over time. The authors call this process "pruning,"preventing the colony from getting lost or separated from the rest. The collective behavior in harvester ants is regulated through interactions between outgoing foragers and returning foragers ants using brief antennal contacts. In [31], ants wait in the entrance chamber inside the nest, using its interactions to decide whether to stay or leave the nest to forage. Demonstrating that the time necessary to send more ants to forage depends directly on the rate at which successful foragers return with the food. The work was extracted by field experiments monitoring 300 colonies where the age of each colony is known in Rodeo, NM, USA with the red harvester ant *Pogonomyrmex barbatus*. Finally, It is important to define that environment does not exist independently of organisms. What environment means is the way or how organisms interact with it, as shown in [46]. The more food is available in the space surrounding the nest, the more quickly foragers find it and return to the nest. So, the contact rate between returning foragers and potential foragers depends on food availability. The algorithm that harvesters' ants use to regulate foraging activity relies on the rate of successful foragers returning with food, then the potential forager leaves the nest to collect

more food following the trails left by scouters. However, if all scouts remain on the same path, they cannot find something off it. Here, random movement increases the probability that a scout encounters novel resources and creates new paths to this place, as shown in [68].

## 2.2. Exploration Using Reinforcement Learning

This sub-section explains how to use reinforcement learning to explore and navigate an unknown environment. First, the robot model used is shown, followed by the reinforcement learning algorithm.

### 2.2.1. Agent Model

The agent type considered in this section for exploring unknown environments using reinforcement learning is robots with differential traction, which are widely used in the industry and academy. The kinematic model for this kind of robot based on [71] is explained in this section. As shown in Figure. 2.1 the robot can move in a two-dimension space taking $Y_W$, $X_W$ as the inertial reference frame and $Y_R$, $X_R$ as the robot reference frame.

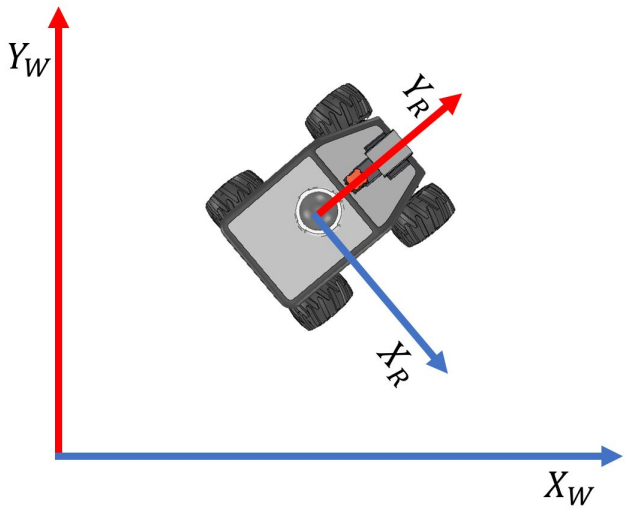



**Figure** 2.1: Robot axis versus inertial reference frame.

These robots use wheels, as shown in Figure. 2.2, which are standard wheels directionless. Where $r$ is the radio of the wheel, $l$ is the distance from the Center of Mass (CoM) of the robot to the wheel, $\alpha$ is the angle between the local robot axis $X_R$ and the line that goes from the CoM to the wheel, $\beta$ is the angle formed by the line that goes from the CoM to the wheel and the tangent line from the wheel axis, $v$ is the linear velocity, and $\dot{\varphi}$ is the angular variation of the wheel.
The kinematic model can be expressed in compact form as,

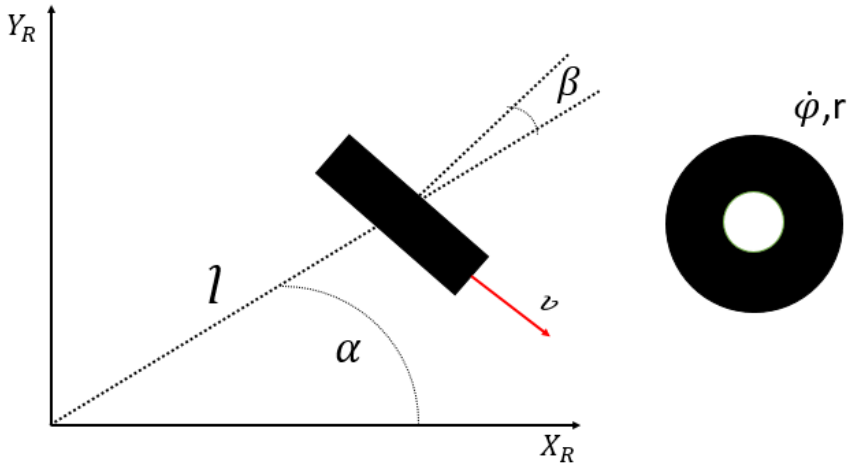



**Figure** 2.2: Parameter definition of standard wheel directionless in differential robots.

$$\dot{\xi}_W = R(\theta)^{-1} J_1^{-1} J_2 \dot{\varphi} \tag{2.1}$$

where $\dot{\xi}_W$ is the vector that contains $\dot{x}_R, \dot{y}_R$, and $\dot{\theta}_R$ the linear velocity in the axis $x$, $y$, and angular velocity of the robot in the robot reference frame, $R(\theta)$ is the rotation matrix to translate the robot reference to the inertial reference frame, $J_2 \in \mathbb{R}^{nw \times nw}$ is a diagonal matrix that contains the radios of the number of wheels $nw$, and $J_1 \in \mathbb{R}^{nw \times nw}$ captures the other parameters of the wheel.
When considering the rolling restrictions on the wheel we can say it is as follows,

$$[\sin(\alpha+\beta), -\cos(\alpha+\beta), (-l)\cos(\beta)] R(\theta)\dot{\xi}_W - r\dot{\varphi} = 0 \tag{2.2}$$

while the sliding restrictions do not have movement as expected due to the standard wheel can not move freely on that axis, so the equation is,

$$[\cos(\alpha+\beta), \sin(\alpha+\beta), (l)\sin(\beta)] R(\theta)\dot{\xi}_W = 0 \tag{2.3}$$

So when matching (2.2) and (2.3) the matrix form of (2.1) can be found as

$$\begin{bmatrix} \sin(\alpha_1+\beta_1) & -\cos(\alpha_1+\beta_1) & (-l_1)\cos(\beta_1) \\ \sin(\alpha_2+\beta_2) & -\cos(\alpha_2+\beta_2) & (-l_2)\cos(\beta_2) \\ \cos(\alpha+\beta) & \sin(\alpha+\beta) & l\sin(\beta) \end{bmatrix} R(\theta)\dot{\xi}_W = \begin{bmatrix} J_2\dot{\varphi} \\ 0 \end{bmatrix} \tag{2.4}$$

where just one sliding restriction is considered since the robot used has differential traction. This means both wheels are aligned by the same axis, so one restriction is linearly dependent on the other, so it does not contribute new information to the matrix. To know the direct

kinematic, it is necessary to isolate $\dot{\xi}_W$ from (2.4), and considering there are parameters that already have values due to corresponding to the inherent robot model such as $\alpha_1 = -\pi/2$, $\beta_1 = \pi$,$\alpha_2 = \pi/2$, $\beta_2 = 0$. By doing so the direct kinematic is,

$$\dot{\xi}_W = R(\theta)^{-1} \begin{bmatrix} 1 & 0 & l_1 \\ 1 & 0 & -l_2 \\ 0 & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} r_1 & 0 & 0 \\ 0 & r_2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\varphi}_1 \\ \dot{\varphi}_2 \\ 0 \end{bmatrix}$$

finally, doing the operations, the direct kinematic model of a differential robot is the following,

$$\dot{\xi}_W = R(\theta)^{-1} \begin{bmatrix} \frac{1}{2} r_1 \dot{\varphi}_1 + \frac{1}{2} r_2 \dot{\varphi}_2 \\ 0 \\ \frac{1}{2l} r_1 \dot{\varphi}_1 - \frac{1}{2l} r_2 \dot{\varphi}_2 \end{bmatrix}. \tag{2.5}$$

### 2.2.2. Reinforcement Learning Algorithm

Reinforcement learning appears in the literature [40] as an algorithm that implements Q-Learning, here, a mobile robot can find the optimized paths in an unknown and non-convex space. Humanoid robots can reduce energy consumption in several movements, as shown in [38],[37]. It is worth mentioning that there are many applications in robotics where Reinforcement learning can be applied and, in many cases, improve performance. The use of learning systems by reinforcement and Cerebellar model articulation controllers (CMACs) allows for reducing learning time by making it faster. Nevertheless, this approach still works with the maze challenge. In [57], the use of an artificial neural network and Q-Learning is proposed to optimize the path planning of a mobile robot. Where reinforcement learning allows the environment sampling to be used by the neural network to achieve path planning in the unknown environment. This sub-section is part of the work [11], which makes an agent learn more efficiently while navigating in a non-convex and unknown space avoiding collisions.
The main purpose of this algorithm is based on some behaviors that ants use when exploring near areas to the nest. As explained before, some present random behaviors when exploring, guaranteeing to know new places to exploit. This is the principle of reinforcement learning, a set combination between exploration or exploitation, in ants these behaviors meaning searching randomness for a food source, and at the moment, at least one ant found the nest and will make sure to exploit it as much as possible while maintaining few ants still exploring for more locations [68].
The reinforcement learning system implemented here allows a robot to learn to navigate despite ignoring the information about the objects which surround it. Since the navigation system does not depend on the robot's spatial location, the robot can learn to navigate in an unknown space facilitating the process of exploration and adaptation to time-varying environments by using the Q-Learning algorithm. The learning process is based on switching the robot through a state space $s$ by selecting actions. The action selection can be done randomly or deterministically according to the behavior wanted in the robot. Once the

action is taken and the robot updates its current state, the new state is evaluated, then a value of reward and punishment is generated according to the convenience of the new state. The reward generates changes in the $Q \in \mathcal{R}^{s \times s}$ matrix, which determines the quality of the action taken in a previous state. This $Q$ matrix is the basis of the acquired knowledge and will determine the robot's behavior in the long term.
For this algorithm is important to define some basic concepts such as the states, actions, learning rules, and exploration policy, which are defined below.

### States

The states of the algorithm are defined by the sensors' measurement and the number of sensors in the robot. Due to the states must be discrete, the information coming from the sensors is discretized. The dimensionality of the system states is depicted by (2.6),

$$SD = (D_m)^{n_s}, \tag{2.6}$$

where $D_m$ is the quantity generated by the discretization of the distance measurement by the sensor, and $n_s$ is the number of sensors in the robot. Here, the state dimensionality refers to the state quantity of the system. Therefore, it indicates the complexity of the system. It is worth considering that the state's quantity grows exponentially according to the number either the sensor and the discretization of the measurements.

### Actions

The actions are all the possible movements that can be performed by the robot, such as forward, backward, rotate either left or right. In the particular case of omnidirectional robots, the actions are the discretized version of all the possible directions in which the robot can move. The system states, and actions define the size of the $Q$ matrix. As a result, it is necessary to keep the dimension of system states and actions as small as possible to simplify the learning process and reduce the computation cost.

### Reinforcement Learning

This sub-section uses the classic Q-learning equation based on the Belman-Equation. However, the way Q values are updated is modified. Since the states are defined by the measurement of the distances of the sensors. It is not possible to know what will be the new state of the robot before executing the action. As a consequence, the update of the Q values is updated after the action is executed. Taking this into account, the value to be updated is the value of the previous state $Q(s_{t-1}, a_{t-1})$ based on the new state $Q(s_t, a)$, and reward policy as shown by (2.7),

$$Q(s_{t-1}, a_{t-1}) \leftarrow (1 - \alpha)Q(s_{t-1}, a_{t-1}) + \alpha(R + \gamma \text{max}Q(s_t, a)), \tag{2.7}$$

where $Q(s_{t-1}, a_{t-1})$ is the value of the previous state in the chosen action, $\alpha$ is the learning rate, $R$ is the reward policy, $\gamma$ is the discount factor, and $Q(s_t, a)$ are the $Q$ values for all actions in the current state. Equation (2.7) depicts that the update is performed backward in time. This is a very similar way to learn in real life. "We only know the efficiency of our actions after having executed it and evaluated the new state to which that action has led us"We call this experience. The parameters involved in the update of the Q values are going to be explained below.

### Reward Policy

The classic version of reinforcement learning uses a reward associated with each state. The objective is to reach the state where the maximum reward is acquired. The proposed algorithm does not have the states associated with the robot location. Therefore, no exact procedure leads the robot to reach the optimal state. The goal in navigation is to keep the robot as far away as possible from obstacles while it is freely navigating, avoiding obstacles. Since the states are determined by the sensors' measured distance and the state transition is based on the change of measured distance, the reward policy is based on maximizing the distance to the obstacles. The reward policy is based on the time derived from the sensor measurements and the transition of states as shown by (2.8),

$$R_{(t-1)} = \sum_{i=1}^{n} \underbrace{a_i \frac{dm_i}{dt}}_{Dynamic} + \underbrace{b_i m_{i(t)}}_{Static}. \tag{2.8}$$

The reward equation (2.8) is represented by the sum of the dynamic and static factors of the robot sensors measurements, where $n$ is the number of distance sensors used by the robot. The dynamic section of the reward equation refers to a dynamic reward that depends on the time derivative of the sensor measurement. The time factor is given by the state transition time as shown by $dm_i/dt$ where $m_i$ is the measurement of the $ith$ sensor. The dynamic reward is multiplied by a factor $a_i$, which can vary according to the sensor position (e.g., a front sensor could have larger values than side sensors). The value of the derivative will be positive when the robot moves away from the obstacles. In this case, the value becomes a reward. Otherwise, when the robot approaches the obstacles the value of the derivative is negative. It can be translated as a punishment for the learning system, which will affect the future behavior of the robot. To simplify the calculations of the dynamic reward, it is assumed that the state transition is carried out at a constant time rate. This leads to converting the calculation of the derivative into a simple difference of values between the state measures, as shown by the (2.9),

$$\frac{dm_i}{dt} = m_{i(t)} - m_{i(t-1)}, \tag{2.9}$$

where $m_{i(t)}$ is the $ith$ sensor measurement in the current state and $m_{i(t-1)}$ is the $ith$ sensor measurement in the previous state. The static part of the reward equation refers to the

sensor's measurement in the robot's current state $m_{i(t)}$. This value is a reward for positions away from obstacles, and if it is close to an obstacle, the result obtained is a punishment. The static part is multiplied by the $b_i$ value, which varies according to the sensor that is evaluated by giving again greater values to the front sensors.

### $\alpha$-Learning Rate

The learning rate ($\alpha$) indicates the changing rate for updating the Q values in the learning process. Ideally, alpha is expected to start with values close to 1, but it will decrease as time passes to values close to 0 but never reach it. The $\alpha$ decrements are related to the number of times that the value of $Q(s_{t-1}, a_{t-1})$ has been updated. The $\alpha$ variable should have the form as shown by (2.10),

$$\alpha = \frac{1 - \alpha_\infty}{1 - e^{(\alpha_s n_{(s,a)} \alpha_{50})}} + \alpha_\infty, \tag{2.10}$$

where $\alpha_\infty$ is the expected value of $\alpha$ when time tends to infinite ($\alpha_{(t\to\infty)}$), $\alpha_s$ indicates the expected decrease rate of $\alpha$, $\alpha_{50}$ is the number of updates of $Q(s_{t-1}, a_{t-1})$ required by $\alpha$ to reach 50 % of its initial value, and finally $n_{(s,a)}$ is the number of times the value of $Q(s_{t-1}, a_{t-1})$ has been updated.
Figure. 2.3 shows the value of $\alpha$ over time with parameters of $\alpha_\infty = 0{,}1$, $\alpha_{50} = 5$, and different values of $\alpha_s = 0{,}5, 0{,}8, 1{,}5$.

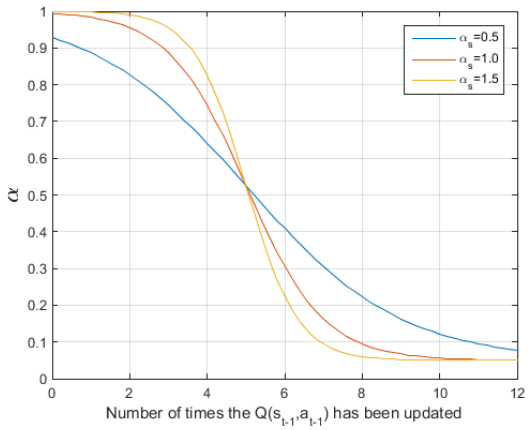



**Figure** 2.3: Decrease in the learning factor over time.

### $\gamma$-Discount Factor

The discount factor ($\gamma$) determines the impact of the reward of the current state for the proposed approach. A constant value of 0,8 is assigned for the present case, allowing the robot to update the learning values in a small proportion.

#### Exploration Policy

This policy allows the agent to choose between the best action found so far or try another action, these selections are called exploitation and exploration, respectively. The objective of this policy is to provide the agent possibility to explore when the learning process begins, but as the agent evolves, it will end up choosing the action with the best $Q$ (exploitation), (2.11) describes the exploration policy.

$$a_t = \max\left(Q(s_{(t)}, a) + \frac{k_a}{n_{(q)} + 1}\right), \tag{2.11}$$

where $a_t$ is the action to be chosen, $Q(s_{(t)}, a)$ is the Q value of all the possible actions of the current state, $n_{(q)}$ is the vector with the number of times that each possible action of the current state has been historically chosen, and $K_a$ is a factor that helps to alternate between the action with the best $Q$ value and the other actions. Usually, $K_a$ has to be greater than $Q$. However, for the present proposal, $K_a \to \max(R)$. The maximum reward is obtained by evaluating the rewards (2.8). The objective of (2.11) is to switch the action selection between the best action and the other possible actions of the current state. Initially, the values of $Q$ are zero, and $K_a$ starts divided by 1, given that $n_{(q)}$ is also zero for all actions. This initial parameter causes any of the possible actions to be chosen. However, as time passes, $n_{(q)}$ grows according to the number of times a particular action has been selected. When $n_{(q)}$ grows the $K_a$ factor tends to zero, and the policy will begin to choose the optimal action more frequently.

## 2.3. Exploration Using Voronoi Tessellation

In this section, the exploration of an unknown space using multi-robot systems is achieved by applying Voronoi Tessellation and some concepts from Graph Theory. Because of this, the system modeling and some fundamental concepts in graph theory are developed below just before showing the main algorithm.

### 2.3.1. Fundamental Concepts on Graph Theory

Graph theory has been widely studied recently because it helps to model the interaction in multi-agent systems, which can be either in communication or control. When graph theory is used for both characteristics, the word used to relate is multi-agent coordination, a term used in this work. Through the development of this work, many terms from graph theory are needed, which is why in this section, basic concepts are shown based on [63]. Let's begin by explaining what a graph is. In a dynamic world where agents (robots, humans, animals, devices, etc.) constantly interact with their environment and other agents, the interaction dynamic can be captured by the representation of it in a graph. The interaction can be

communication or other parameters that help each agent take action or update control laws that interact with their environment. Here each agent, let's say the robot is an element of the set of nodes $\mathcal{V} = \{v_1, v_2...v_n\} \in \mathbb{R}^n$, where $n$ is the number of robots interacting in the workspace. Knowing which robots interact with which robot is really important, so these variables are encapsulated in the links set $\mathcal{E} = \{v_1v_1, v_1v_2...v_1v_j, v_2v_j...v_iv_j\}$, where $v_i$ is the $ith$ agent and $v_j$ is the $jth$ neighbor. Each component in the matrix indicates if there is any interaction between agents. Finally, a graph is formed by nodes and links $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Both relations allow us to model and know how a multi-agent system works. Although the definition of a graph is important, it is not enough to know what is happening, so it is necessary to show this information in another way that can be used for further analysis. First, the adjacency matrix $\mathcal{A}(\mathcal{G}) \in \mathbb{R}^{n\times n}$, this matrix contains the information of how the graph is connected and is filled as follows,

$$[\mathcal{A}(\mathcal{G})]_{ij} = \begin{cases} 1 \; if \; \mathcal{V}_i\mathcal{V}_j \in \mathcal{E} \\ 0 \; Otherwise \end{cases}, \tag{2.12}$$

the degree matrix $\mathcal{D}(\mathcal{G}) \in \mathbb{R}^{n\times n}$ that represents the number of agents that have a link connection with each robot in the main diagonal as shown bellow,

$$\mathcal{D}(\mathcal{G}) = \begin{bmatrix} d(1) & & \\ & \ddots & \\ & & d(n) \end{bmatrix}. \tag{2.13}$$

Finally, there is one matrix that contains the entire information about the graph and is made from the subtraction of (2.12), and (2.13), called the Laplacian matrix $\mathcal{L}(\mathcal{G}) = \mathcal{D}(\mathcal{G}) - \mathcal{A}(\mathcal{G}) \in \mathbb{R}^{n\times n}$

$$\mathcal{L}(\mathcal{G}) = \mathcal{D}(\mathcal{G}) - \mathcal{A}(\mathcal{G}), \tag{2.14}$$

For instance, if we consider the graph formed by the five quadrotors shown in Figure 2.4 where lines that connect the circles enclosing the drones represent those quadrotors that have communication. The matrix (2.12), (2.13), and (2.14) are equal to

$$\mathcal{A}(\mathcal{G}) = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \quad \mathcal{D}(\mathcal{G}) = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathcal{L}(\mathcal{G}) = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 3 & -1 & -1 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \end{bmatrix}.$$

It is worth mentioning that up to this point, everything that has been mentioned about the graph considers homogeneous agents, which means that all of them have the same dynamics

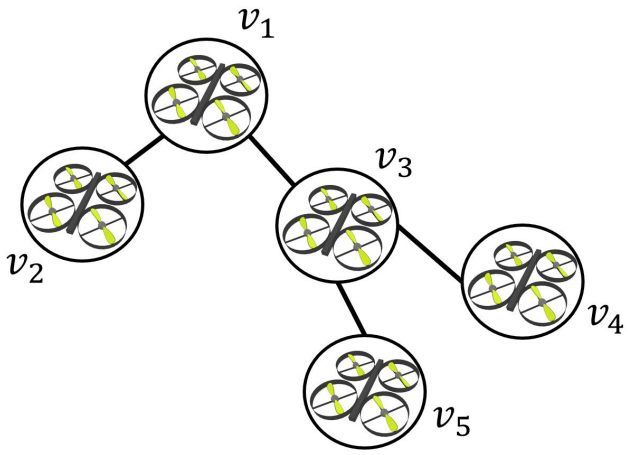



**Figure** 2.4: Example of how to model multi-agent systems using Graph Theory.

and capabilities. Moreover, all of the links were bidirectional, which means both of the agents' sharing a link have access to the information of the other. On the other hand, when more than one kind of robot is used, the multi-robot system can be considered heterogeneous because it might have different dynamics or capabilities. Furthermore, when two robots share a link but just one of these can access the information of the other, it is considered a Digraph $\mathfrak{D}$, which means now the links can have directions and are represented by arrows. Figure 2.5 shows an example of how a heterogeneous directed graph looks. Here there are two different kinds of robots. First are the quadrotors, which can move in three dimensions through the air. Second, the Turtlebots are terrestrial robots that can only move on a surface using their wheels. The degree matrix of a Digraph is different due to the links that relate to who shares information with who differs from the graph when both robots can obtain information. Here, it is assumed that the agent with the tail shares information with the agent where the head of the arrow reaches. Then (2.13) is replaced by the in-degree matrix $D(\mathfrak{D})_{in}$ which is filled as $[D(\mathfrak{D})]_{ii} = d_{in}(v_i) for\ all\ i$, where $d_{in}(v_i)$ is the grade of the node $v_i$ considering the in-degree case which means the number of agents where the $ith$ agent receives information from. The Laplacian matrix and the Adjacency matrix remain the same as a graph.

## 2.3.2. System Modeling

### Environment

The exploration task involves using a team of $n$ robots with ideal radial sensing to explore, map, and navigate an unknown, non-convex, and bounded space $M \in \mathbb{R}^2$. Where is possible to differentiate between already explored areas $S \subset M$, areas clear to move $S_F \subset S$, obstacles $S_O \subset S$, and which areas are still unexplored $S^C$, noting that $S_f \cup S_O = S$ and $S \cup S^C = M$. Taking into account that the space where the robot is navigating is non-convex, the distances from one position to another position in the space are not always Euclidean. This concept is

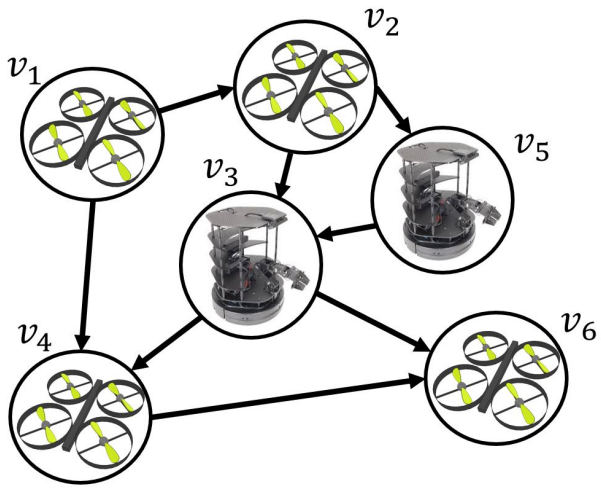



**Figure** 2.5: Heterogeneous multi-robot system, modeled as a Digraph.

illustrated in Figure 2.6, just supposed the quadrotor wants to approach the terrestrial robot position. Knowing the distance between robots is not as easy as calculating the Euclidean distance. As a consequence, robots simply cannot move through obstacles. Because of this, the way to calculate the geodesic distance between two points is by using the so-called Flooding Distance, which is based on the breadth-first search algorithm. In Figure 2.6 there are shown all the distances. The dashed green line is the Euclidean Distance for the 2-dimensional case. The solid blue line is the four-neighbor flooding distance (4NFD), also known as (Manhattan Distance) due to only permits movements up-down and right-left like moving into the blocks in Manhattan, New York City. The solid orange line is the Eight Neighbor Flooding Distance (8NFD) that, in addition to the movement that 4NFD allows, the main diagonals are added. It is worth mentioning that the 4NFD and 8NFD shown in the image are not the optimal ones, they are just one example to illustrate how each method allows the robot to move. Even though they are not optimal it is noticeable that 8NFD is shorter than 4NFD, reducing the error in relation to the optimal distance in a two-dimensional non-convex space.

### Dynamic Robot Model

The dynamic of the robots considered for this algorithm is a simple integrator, considering in robotics, most of the robot platforms can be transformed into this dynamic without the loosing of generality as shown in [10]. It is possible from the integrator dynamics in (2.15) employing a kinematic control to most robots.

$$\dot{x}_i = u, \tag{2.15}$$

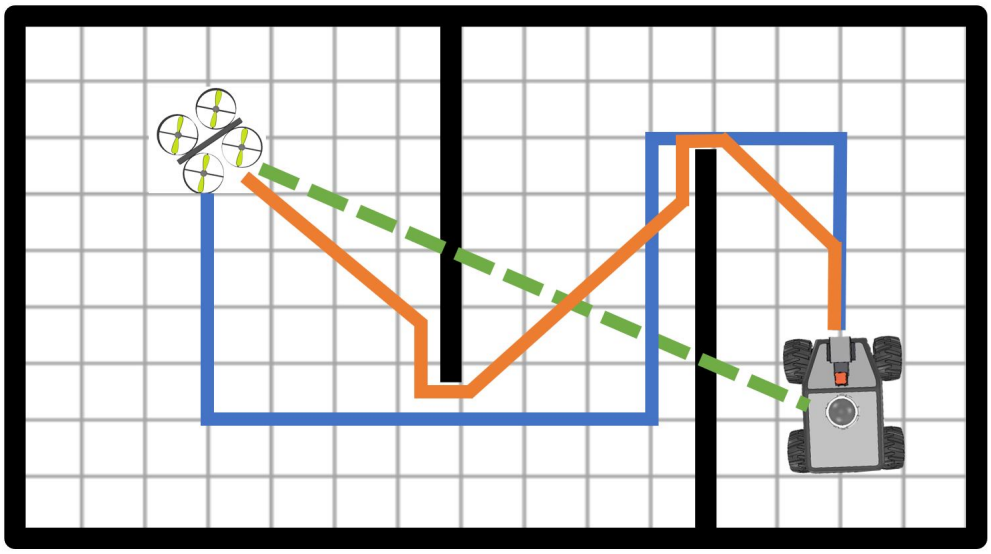


**Figure** 2.6: Four Neighbor Flooding Distance (Blue Line), Eight Neighbor Flooding Distance (Orange Line), and Euclidean Distance (Green Dashed Line)

where $\dot{x}_i \in \mathbb{R}^2$ is the linear velocity of the $ith$ robot and $u \in \mathbb{R}^2$ is the signal control to be designed to reach the desired behavior. Considering that the environment is a non-convex space which means there are obstacles in it, the sensing of the robot has some considerations that are worth stating, such as the effective area sensed by one robot $A_{s_i}$. As can be seen in Figure 2.7, the area sensed $A_{s_i}$ not only does depend on the sensing radius but also the line of sight $L_{s_i}$ of the robot. Where $L_{s_i} = \{q \in S_F | \forall p \in \{\alpha x_i + (1-\alpha)q\}, p \notin S_O\}$ where $\alpha \in [0, 1]$ is a scalar allowing parametrizes the straight line between the points $x_i$ and $q$ that might be obstructed by an obstacle in the space.
In Figure 2.7, the yellow area represents the ideal radius sensed area that the robot has, blue area represents the line of sight of the robot, considering that all the black lines are objects $S_O$ where the robot cannot see through it, and the white area is the space clear to move $S_F$. The effective area sensed by the robot is then determined as the intersection of the two subsets and can be described as follows,

$$A_{s_i} = \{q| \ |x_i - q| < r_{s_i}\} \cap L_{s_i}. \tag{2.16}$$

As expected, when space where the robot is convex, the effective sensed area will be equal to the sensing radius. On the other hand, when the robot navigates in a non-convex space, every time the robot has an obstacle in its radius of sense, the effective area sensed will be reduced and affected by the changes in the line of sight.

### 2.3.3. Exploration Algorithm

The exploration task purpose is to allow robots to navigate through a known environment to achieve any other task that might emerge in the rescue mission. The algorithm developed here

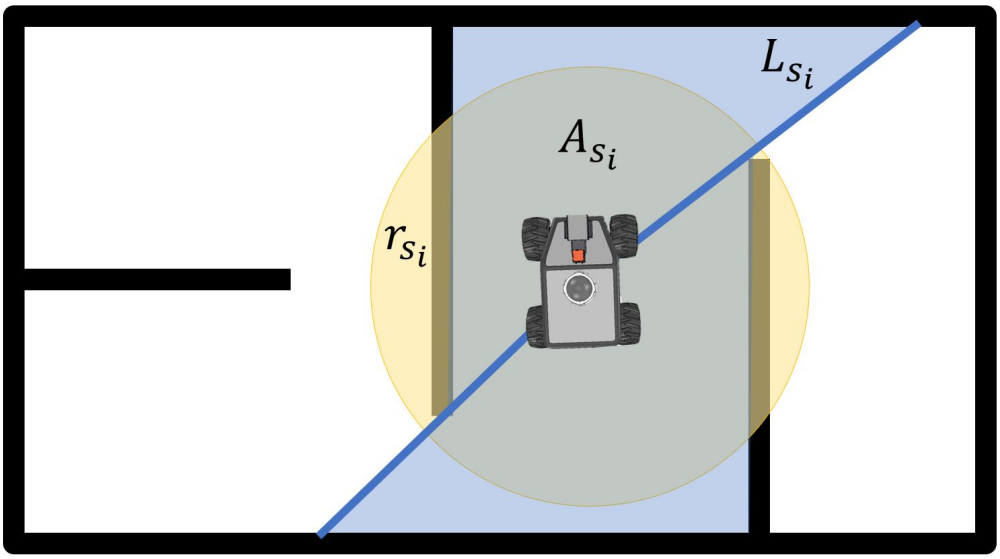



**Figure** 2.7: Radius sensed, Line of sight, and are sensed by a single robot in a non-convex space.

is based on the work presented by [47] with some modifications due to the original algorithm contemplating the homogeneous agent case, a convex space, and connection restriction. In contrast, in [23], which is the work where this section is based, it indicates how to relax these conditions while exploring the area shown below.
The algorithm works by robots tracking their exploration frontier $\delta S$ all the time. Where this frontier is the limit that separates the known and the unknown spaces in each robot in the environment. The selection of the interesting point to be followed by robots in the exploration task is made in several steps. Through the use of the well-known Voronoi Tessellation technique, where each robot determines which points in the space can be reached fastest than others as follows,

$$V_i = \{q | \text{dist}(x_i, q)/\bar{v}_i < \text{dist}(x_j, q)/\bar{v}_j, \forall j \in N_i\}, \tag{2.17}$$

where $V_i$ is the Voronoi cell of the $i^{th}$ robot, $dist(p_1, p_2)$ is the geodesic distance between the points $p_1$ and $p_2$ estimated employing the Flooding algorithm, and $\bar{v}_i$ is the average speed of the $i^{th}$ robot. It is important to notice that (2.17) is different from the original Voronoi Tessellation equation in the sense that here it allows the robot to have different capabilities in terms of velocities and that the distance measure is using Flooding Distance instead of Euclidean. After the Voronoi Tessellation is reached, each robot determines the frontier $\delta S_i$ that belongs to it, so now they can determine a weight function $\phi_i(q, \delta S)$ over its own cell as $\phi_i(q, \delta S_i) = \exp\left(-\frac{1}{2\sigma^2}\text{dist}(q, \delta S_i)\right)$, then used to determine each Voronoi cell centroid as,

$$V_{ci} = \frac{\int_{V_i \cap C_{ir}} \phi(q, \delta S_i) q \, \text{dA}}{\int_{V_i \cap C_{ir}} \phi(q, \delta S_i) \, \text{dA}}, \tag{2.18}$$

where $q$ is the interesting point within the space, $\sigma$ is a constant that determines how fast the weight function decreases with the distance, $V_{ci}$ denotes the centroid of the $ith$ robot Voronoi cell, dA is the area differential and $C_{ir}$ denotes a circle of center $x_i$ with radius $r$. Once the centroid is found, the task becomes to track it, so the first component of the controller $u_i$ is designed as a simple proportional controller $u_{exp_i} \in \mathbb{R}^2$ that leads to the target point as the iteration increases, $u_{exp_i} = K_{exp}(V_{ci} - x_i)$ where $k_{exp}$ is a proportional constant guaranteeing stability.

To navigate safely in space, it is necessary to give some rules that will make robots avoid collisions with either other robots or obstacles in the environment. These rules are based on the Reynolds rules, which are in charge of keeping a minimum distance to prevent animals in a pack, swarm, or school of fish from colliding with another member while keeping a maximum distance allowance to avoid a member getting lost. Artificial Potential functions have the equivalent behavior as the Reynolds rules for ants that can be applied to robots. Which can either maintain the maximum communication distance among robots by using $x_i \in \{q||q - x_j|_2 < \rho_2,\ \forall j \in N_i\}$ where $x_i$ denotes the position of the $ith$ robot, $\rho_2$ denotes the communication range, and $N_i$ denotes the neighborhood of the $ith$ robot. Even though this behavior is not interesting in this application because it will require the robots to keep communicating with at least one other robot, so they can receive information from all the robots directly or indirectly. And in this way, the communication graph can be formed as explained in Figure 2.8 where every time each communication radio (blue radius on each robot) intersects with another robot communication radio (darkest blue areas), then a communication link between these robots is established as shown in the graph on the right.

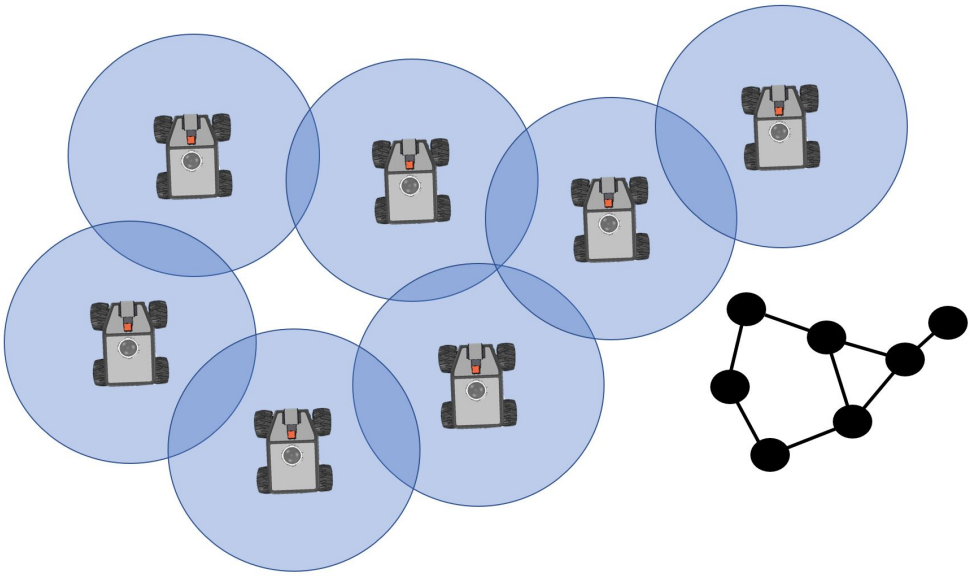


**Figure** 2.8: Example of how the communication graph is formed.

In contrast, what is expected for the exploration behavior is to break and create commu-

nication as needed to share information but allow robots to go in different directions and explore wider areas as fast as possible. So, what is required here is a rule that establishes collision avoidance with other robots or obstacles as,

$$x_i \in \{q||q - x_j|_2 > \rho_1,\ \forall j \neq i\} \cap \{q||q - x_o|_2 > \rho_1,\ \forall o \in S_O\}, \tag{2.19}$$

where $\rho_1$ denotes the security radius of the robots and $S_O$ denotes the obstacles in the environment. The control signal that guarantees those conditions is the following,

$$u_{ac} = \sum_{j \in N_i} -\frac{\hat{x}_{io}}{(|x_i - x_o|_2 - \rho_1)^2} - \frac{\hat{x}_{ij}}{(|x_i - x_j|_2 - \rho_1)^2}$$

where $\hat{x}_{ij}$ corresponds to the unitary vector that points from $x_i$ to $x_j$, $u_o$ corresponds to the control signal component responsible for the static obstacle avoidance, $x_o$ corresponds to the closest point of any obstacle to the robot, $\hat{x}_{io}$ corresponds to the unitary vector that points from $x_i$ to $x_o$, and $u_{ac}$ corresponds to the artificial potential component of the control signal.

In the previous Section, some behaviors regarding ant exploration showed how using some communication channels, either mechanical or chemical, can help ants explore an area in an optimized way. Some of them are brief antennal contacts that proportionate local information and pheromones secretions which can give ants a piece of global information about what other ants suggest to do. These mechanisms allow ants to indicate where the resources are for the colony to exploit or which areas to avoid due to some predators in it. The accumulation of pheromones permits the creation of paths to be followed or avoided. This approach can inspire search and rescue operations using mobile robots to solve simple problems like coverage algorithms. So, here robots deploy markers periodically in the environment, emulating pheromones' responses in ants. But to indicate if any robot has already explored the area. If a robot detects a marker, it will provoke the robot to lose interest in the zone around it, generating the robot to give more importance to other unexplored areas. This will reduce the probability of repeatedly overlapping and exploring the same area due to a lack of communication. This concept modifies the Voronoi cell of each robot as follows,

$$V_{a_i} = V_i - P_j \cap V_i, \tag{2.20}$$

where $V_{a_i}$ corresponds to the new Voronoi cell of each robot, $V_i$ is the Voronoi cell used in (2.17) and $P_j \in \mathbb{R}^2$ which represents the area cover by the markers deployed from other robots. When a robot reaches the position of a marker that does not correspond to itself, it unassigned the area within a radius $r_s$ of the marker from the Voronoi cell. On the other hand, the local information that ants share when they cross their antennas is emulated by creating temporal communication links. Allowing robots to share their map with neighbors, receive the known map from others, and build a map that could be closer to the global one any time another robot is in range of communication.

**Algorithm 1** presents the main loop employed for the environment exploration mission. Here, it is presented as part of the original Heterogeneous Discoverage presented in [80] with the difference that in this case, artificial potentials do not consider the connectivity maintenance, and also, it incorporates the detection and deployment of markers on the mission area. As mentioned earlier, each robot can realize the exploration task without agent communication. Therefore, it is not necessary to set robots initially connected to guarantee the mission success; considering that each robot has a determined sensing and communication range, it is clear that the larger the robot number, the faster the exploration process will be cleared. It is worth mentioning that the main purpose of this work is to introduce the bio-inspired complement of the pheromones trace (markers) and the information-sharing methodology.

**Algorithm 1** Main Loop

```
1: function MAIN LOOP(Robots, scene)      ▷ Where Robots - Robots Set, scene - matrix
2:     for r ∈ robots do
3:         if mod(t, T_m) = 0 then
4:             Deploy marker
5:         end if
6:         Sense surroundings
7:         Update Map
8:         Update Flooding
9:         Update Line of Sight
10:        for rn ∈ robots - {r} do
11:            D_rn = sqrt((p_rx − p_rnx)^2 + (p_ry − p_rny)^2)
12:            if D_rn ≤ r_C then
13:                Share information between r and r_n
14:            end if
15:        end for
16:        Update Artificial Potentials
17:        Evaluate Voronoi Cell(s)
18:        Unassign areas covered by other robots' markers
19:        Determine the Voronoi Centroid
20:        Track the Voronoi Centroid
21:    end for
22: end function
```

## 2.4. Fundamental Concepts on Hybrid Systems

It is worth mentioning that the whole theory regarding Hybrid Systems in this thesis is based on [43]. Here Hybrid Systems are defined as a combination of continuous-time and discrete-time dynamical systems. There are a lot of examples where the merge of these two dynamical system modeling can describe in a better way the physical phenomena. For instance, as we all know, a body that experiments with changes in velocities and momentum due to collisions or juggling dynamics the system moves continuously. Still, all of the sudden changes can be described as a discrete event. In biology, we can see hybrid systems in the way an organism behaves smoothly or continuously doing a specific task, which could be modeled as a continuous-time dynamic, but at the moment when something generates a stimulus or disturbance, changing the behavior is where discrete-time dynamical modeling takes place. So every organism that leads to switching behaviors when a motivation appears can be modeled as a Hybrid System. The mutual interaction in modeling the continuous-time and discrete-time dynamics leads to rich modeling of any cyber-physical system not found that working purely with just one of the dynamics. Consequently, many challenges appear while doing so, such as instability cases, propagation of mistakes, and robustness of the controllers, among others. To start analyzing how the Hybrid Systems work, clarifying some concepts as definitions explained below is essential.

A comprehensive notation used to model a continuous-time dynamical system for a first differential equation is $\dot{x} = f(x)$, where $x \in \mathbb{R}^n$ for a physical interpretation $\dot{x}$ is a velocity of a rigid body and $f(x)$ the changing of positions while varying time. Although this differential equation can have a different interpretation as $\dot{x} = f(x)$ and $x \in C$, where $C \subset \mathbb{R}^n$, this is the case where there exist state constraints. For instance, a location that a ball can not go due to there being another object there or the presence of perturbations. On the other hand, the notation widely known for the discrete-time dynamical system is the first order equation $x^+ = g(x)$, where $x \in \mathbb{R}^n$, meaning that the updating law of the state depends on the current value of the state. In the same way, as differential equations have constraints is extendable to think in constrained difference equations and difference inclusions, which leads to $x^+ \in G(x)$, where $G(x) \subset \mathbb{R}^n$. Since a Hybrid System is composed of both the continuous-time and discrete-time dynamics, additionally with the set where these dynamic works in the Hybrid Systems are included, both the differential equation and the difference equations are constrained, leading to the following form to describe it,

$$\begin{aligned} \dot{x} &= F(x) \quad x \in C, \\ x^+ &\in G(x) \quad x \in D, \end{aligned} \tag{2.21}$$

where $C$ is called the flow set, $F$ is the flow map, $D$ is the jump set, and $G$ is the jump map. Considering that the switching dynamics or controllers can be modeled as a hybrid system and a Finite State Machine (FSM). As is well known, an FSM is a system that might go through different states or behaviors and is a method in which a system changes its behavior

when a stimulus occurs. For instance, car transmission allows the car to experiment with different velocities and torque capacities depending on what the car needs when the user is in charge of the transition states in the standard case or to the car in the automatic transmission system. An FSM might contain the following elements:

- A set of inputs $v \in \Sigma$ where $\Sigma$ is an input alphabet, and $v$ takes the input values.
- A finite set of states $Q$, such that $q \in Q$.
- An output set of values $\Delta$, such that $\varrho \in \Delta$.
- A transition function $\delta : Q \times \Sigma \to Q$.
- An output function $k : Q \to \Delta$.

Whenever a transition function is needed, it takes an input $v$ and current state $q$ and maps it to the next state $q^+$. For example, Figure 2.9 shows how the transition system in an FSM works, where $k(q)$ is the output function that chooses which state will be the next one. It is essential to clarify that the transition function $\delta$ is expressed for deterministic mapping in the case of a non-deterministic model. The transition function becomes $\delta : Q \times \Sigma \rightrightarrows Q$, meaning that the transition function takes a current state and an input mapping to any of a set of possible states. To guarantee the well functioning of the FSM is necessary to

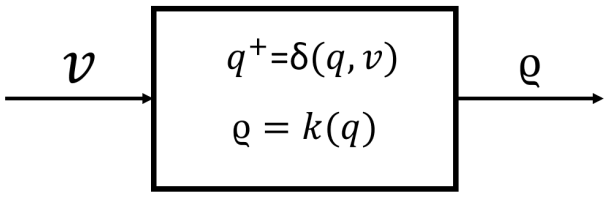



**Figure** 2.9: Finite State Machine Diagram

complement it with guards function $l(q, v, \varrho)$, which contains the laws in which one state, input, and transition function will lead to the next state, avoiding to fall into mistakes of transitioning or Zeno behavior.
Figure 2.10 is shown how the FSM is modeled using Hybrid Systems where the blue block contains the continuous dynamic of the system, which in most cases corresponds to the physics that govern the system. On the other hand, the green block shows how the transitioning of the states works, which is considered a discrete dynamic system. Considering the entire system is appreciable how the continuous and discrete dynamics combine to increase the approximation modeling of the system.

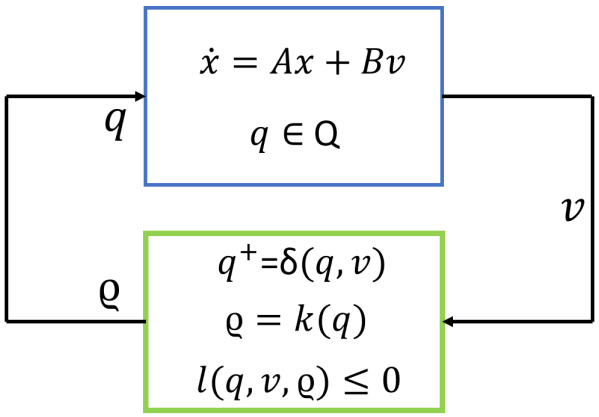



**Figure** 2.10: Finite State Machine Diagram with guards

# 3 Navigation and Victim Detection

## 3.1. Ant Colony Searching for a New Nest Behavior

Japanese ant, Myrmecina nipponica, is a small colony in wet temperature forests consisting of approximately 40 ants with one winged queen. Quorum threshold is the technique ants use to perform a consensus on decision-making that affects the entire colony as a house-hunting process that decides the new place where to live. In [28] is shown how the colony's size directly affects the mean of the threshold needed to reach a quorum consensus and take a decision. The house-hunting process begins with scouts traveling repeatedly from the nest to the new sites and backward. Deploying pheromones while moving, they try to recruit ants to the place that some scouts want, and the transportation of the brood does not begin until a consensus has been reached, which occurs when a quorum of ants is present at the new site. Once the consensus has been reached, scouts change their task to transport the brood to the new site, and when they finish, it is assumed that the relocation has ended.

Social insects may change the nest when the present site becomes unsuitable. In [27], the relation between private and social (pre-established pheromone trails) information is addressed when ants try to perform house-hunting. Being well-informed matters while making decisions. It can go from good to bad. Animals usually base their decisions on information acquired in previous experiences, which can be considered as private information, and cues from other animals, which are considered social information. On the one hand, private information seems to be costly or difficult to obtain but, in some cases, more reliable on the other hand, social information may be cheap to acquire but, in some cases, unreliable and outdated. Opening the possibility of obtaining negative outcomes via information cascades means that individuals copy the behavior of other animals without considering the environmental cues or private information that lead to that behavior, provoking sub-optimal outcomes. Considering the importance of weight private and social information this work presents how ants balance social and private information when the social information conflict at different levels. Since private information sometimes may help colonies avoid making costly wrong decisions. For instance, when they are house-hunting, if during the process of relocation, they agree that the new site does not satisfy the requirements and scouts found a suitable place, the cost could be high due to the brood and the queen being exposed to predators hunting, desiccation and getting lost.

## 3.2. Navigation in known non-convex spaces

### 3.2.1. Swarm Navigation

We consider a set of quadrotors $\mathcal{N} = \{1, 2, ...n\}$ whose interaction are modeled via graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{E}$ represents the communication between quadrotors. Each quadrotor $i \in \mathcal{N}$ has a corresponding state variable $x_i \in \mathbb{R}^3$, which is the location of the quadrotor regarding each axis of the space $\mathcal{S} \in \mathbb{R}^3$. It is necessary to state that $S$ is a non-convex space which means that it is composed of either area clear to move by the quadrotors $\mathcal{S}_f \in \mathbb{R}^3$ and areas with obstacles $\mathcal{S}_o \in \mathbb{R}^3$ that disallow quadrotors to go thru, noting that $\mathcal{S} = \mathcal{S}_f \cup \mathcal{S}_o$. The quadrotor dynamics considered here are determined by a single integrator dynamic $\dot{x}_i = u_i$, where $\dot{x}_i$ are the linear target velocities of each quadrotor and $u_i$ is the signal control to be designed. The approach used to give the desired target position to quadrotors is artificial potential functions, which emulates the attraction and repulsion behaviors presented in nature as in Reynolds rules. Allowing quadrotor $i \in \mathcal{N}$ to maintain a comfortable distance to obstacles and neighbor quadrotors $j \in \mathcal{N}_i$, where $\mathcal{N}_i$ is the neighborhood of quadrotor $i^{th}$. The control signal is a summation of both attraction and repulsive forces as,

$$u_i = u_{a_i} + u_{r_i} + u_{o_i},$$

where, $u_{a_i} = -k_{a_i}(x_i - x_j)$ is the attraction force, $k_{a_i} \in \mathbb{R}_{>0}$ is an established constant. On the other hand, the repulsion force when a comfortable distance is reached is defined as $u_{r_i} = \big( - k_{r_i}(\|x_i - x_j\| - \Delta)\big)(x_i - x_j)$, in which, $k_{r_i} \in \mathbb{R}_{>0}$ is an established constant, $\Delta \in \mathbb{R}_{>0}$ is a minimum distance allowed between quadrotors and $\|x_i - x_j\|$ is the euclidean distance in $\mathbb{R}^3$. In addition, it is necessary to avoid obstacles $x_o = \{x \in \mathbb{R}^3 \mid x \in S_o\}$ in the environment so another repulsion force is needed,

$$u_{o_i} = k_{o_i} \exp\left(-\frac{1}{2}\frac{\|x_i^d - x_o\|^2}{r_s^2}\right)(x_i^d - x_o),$$

where $k_{o_i} \in \mathbb{R}_{>0}$, is an established repulsion constant, and $r_s \in \mathbb{R}_{>0}$ is the security radius in which the Quadrotor avoid collisions and depend on the obstacle size. Considering the generation of desired locations to reach, the objective is to navigate in a known space, being attracted to interesting points while avoiding collisions with other quadrotors and obstacles.

### 3.2.2. Sub-Swarm Generation

Every agent is exposed to different attraction and repulsion forces, which help to keep the swarm cohesion and, at the same time, push the swarm towards the goal. The victims located in the environment permit an attraction force over close quadrotors, which can disturb the normal behavior of the swarm, reducing the velocity of the quadrotors close to the victims. The $k_r$ magnitude was set to stop the closest quadrotors to the victims. When the quadrotors navigate close to the victims, at least one agent must stop near the victim. Once at least

one quadrotor has stopped close to the victim and alerted the swarm about the finding of a victim. Which will then start the generation of a sub-swarm around the localization of the possible victim. A sub-swarm is created by employing the $K-nearest$ algorithm approach. The $K-nearest$ algorithm behaves as a classifier by selecting as his name indicates the $k$ closest $j^{th}$ quadrotors to the $i^{th}$ quadrotor. When the neighborhood has been established, a graph $\mathcal{G}_{ss} \subset \mathcal{G}_s$ where $\mathcal{G}_s$ is the main swarm graph. Thus, $\mathcal{G}_{ss} = (\mathcal{V}_{ss}, \mathcal{E}_{ss})$ where $\mathcal{V}_{ss}$ is composed by the quadrotors belonging to the sub-swarm and $\mathcal{E}_{ss}$ is generated considering a weighted function $\mathcal{W}_{ss} : \mathcal{E}_{ss} \to \mathbb{R}$ such that, $\mathcal{W}_{ss} = \frac{1}{q_{ij}}$, where $q_{ij} = \|q_i - q_j\|$ is the distance between the $i^{th}$ and $j^{th}$ quadrotors. Allowing closer robots is more important to classifying the $i^{th}$ quadrotor between taking itself to the sub-swarm or remaining in the principal swarm.

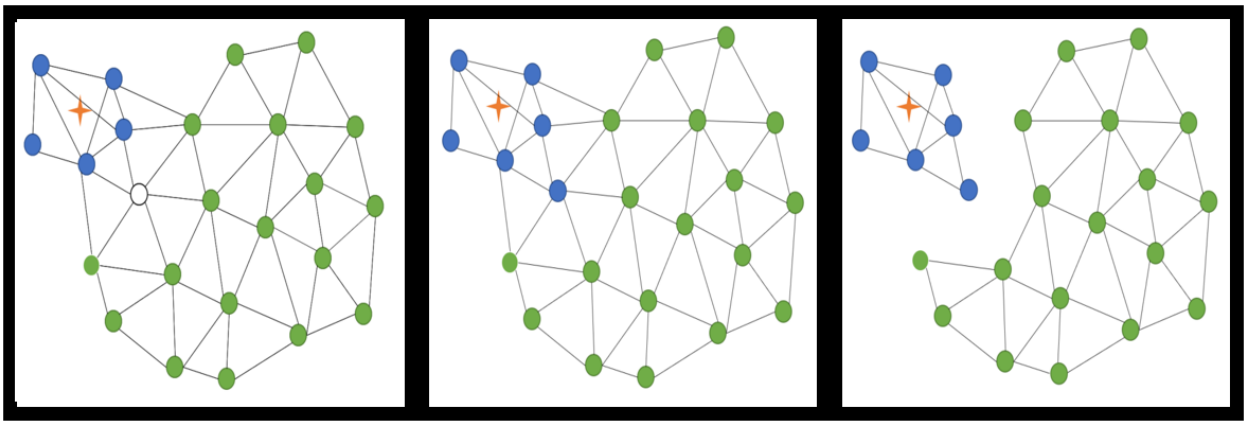


**Figure** 3.1: Sub-swarm generation.

As seen from Figure 3.1, the sub-swarm process is given when some robots in the network detect a possible victim. When a victim is detected, the main inconvenience is deciding which robots will remain in the swarm and which will generate a new one. Consequently, with the K-nearest algorithm in Figure 3.1 is possible to appreciate when a non-filled dot represents the robot that will be classified as a member of the sub-swarm blue dots or not green dots. It is possible to note that he is assigned to the sub-swarm, becoming a blue dot and breaking communication links with the rest of the main swarm.

### 3.2.3. Formation Control

After the sub-swarm generation is established, an algorithm capable of relocating all the robots in the sub-swarm around to the possible victim position is carried out. Increasing the accuracy in the measurements made by the sensor network on the detection of the possible victim. Besides, having a well-classifying method, even though when far quadrotors might add noise into the consensus. This algorithm is capable of putting quadrotors closer to the victims using the well-known rendezvous algorithm, which the original algorithm is explained in [63]. When a certain modification is made to this algorithm, it is possible to take

quadrotors to the perimeter of a circle. The Laplacian matrix and the state vector are used, giving rise to the consensus equation that models the dynamics of the system as follows,

$$\dot{x}_i = -\mathcal{L}x_i,$$

where $x_i \in \mathbb{R}^3$ and $\mathcal{L}$ is the Laplacian of the graph $\mathcal{G}_{ss}$. According to [63], the consensus equation shown above will have a space of agreement $A = \{X \in R^n \mid x_i = x_j \forall i, j\}$ in which, when $\dot{x}_i = 0$ there will be a consensus. On the other hand, in the time domain, $x(t) = e^{-\mathcal{L}t}x_0$ in which, when $t \to \infty$ is evaluated and performing an expansion in Taylor Series, is possible to note that except for the first term, all the others are zero. The first term turns out to be $x(t) = e^{-\lambda_1 t}((u_1)^\top x_0)u_1$ where the first eigenvalue is zero because the whole exponential part tends to 1 which causes the following expression $((u_1)^\top x_0)u_1$. Extending it to all the agents in the graph, we have that the consensus of a non-directed graph is the average of the system's initial conditions. This means that the consensus can be written as follows,

$$consensus = \sum \frac{x_{0i}}{n}.$$

Considering that this rendezvous consensus allows all agents to reach the same position in space, the next step is guaranteeing both connectivity and collision avoidance. To maintain and guarantee the communication among quadrotors in $\mathcal{G}_{ss}$, avoiding collisions among quadrotors and obstacles, respectively, while approaching the target position. Here artificial potential functions are used based on [81], which guarantee both requirements and principles of attraction and repulsion forces inspired by biology known as Reynolds rules. It can be modeled as follows,

$$\psi_{ij} = \frac{1}{\rho_2^2 - ||x_{ij}||^2} - \frac{1}{||x_{ij}||^2 - \rho_1^2}, \tag{3.1}$$

where $\psi_{ij}$ is the artificial potential function between the $i^{th}$ and $j^{th}$ quadrotors, $\rho_2$ corresponds to the connectivity radius permitted among quadrotors and $x_{ij}$ corresponds to the distance between the $i^{th}$ and $j^{th}$ quadrotors. From 3.1, the dynamic of the quadrotors can be imposed as follows,

$$\dot{x}_i = -\sum_{j \in N_i^\sigma} \nabla_{x_i} \psi_{ij},$$

note that this function tends to be infinite when the distance between the agents is $\rho_2$. The more the distance between agents increases, the more the intensity of the attraction among the robots increments, avoiding communication breaks.
The artificial potential functions approach demonstrates connectivity maintenance by evaluating the total energy of the graph connections given by

$$\psi = \frac{1}{2}\sum_{i=1}^{n} \psi_i = \frac{1}{2}\sum_{i=1}^{n}\sum_{j \in N_i^\sigma} \psi_{ij}, \tag{3.2}$$

where $\psi$ is the total energy of all the links and $\psi_i$ corresponds to the energy of all the links concerning the $i^{th}$ quadrotor. From this total energy, it is ascertained whether it is growing, maintaining, or reducing with time from its derivative concerning time according to

$$\dot{\psi} = \frac{1}{2} \sum_{i=1}^{n} \sum_{j \in N_i^{\sigma}} \dot{x}_{ij}^T \nabla_{x_i} \psi_{ij}, \tag{3.3}$$

demonstrating that the total energy of the links tends to decrease over time, and since the energy of the links grows due to the loss of connectivity, it is also demonstrated that the links are maintained. As expected, the cooperative behavior of quadrotors in a sub-swarm converges to the desired location. Then they spread around the target in a circle formation, as shown in Figure 3.2.

## 3.3. Victim Detection

### 3.3.1. Fundamental Concepts on Neural Networks

CNN is a particular class of ANN (Artificial Neural Network) proposed initially in [54] to process digital images in classification or identification tasks. These networks employ convolutional filters with linear rectifiers intended to extract multiple interest features from the image, such as borders, corners, or specific shapes. After the convolutional filtering, the resulting images are down-sampled in the so-called pooling process, reducing the image size but preserving the most relevant information. Those two steps (Convolution and pooling) are repeated several times, where every time, the result is a more significant number of images with smaller dimensions. Finally, the values of the resulting images are given as input to a traditional neural network with fully connected layers whose weights are adjusted in a supervised training process feed by a large number of images adequately labeled according to a human victim detection goal. For the case of victim detection, CNN has a single output in charge of determining if a victim was detected or not.

### 3.3.2. Distributed Victim Estimation Consensus

Sensors are essential when detecting victims successfully in a search and rescue scenario because they oversee acquiring all information from the environment. It is relevant to note that absolute sensors also capture noise in the measurements, which can come from electromagnetic noise, poor conditioning, or even external factors depending on the type of sensors. Sensors are usually modeled using different methods, but considering the methodology worked here using a quadrotor sub-swarm. The distributed linear least square approach seems to be the ideal way to model the dynamics of the sensors with presence noise, represented with additive zero-mean Gaussian noise as shown in Figure 3.4. The sensing of a Gaussian

signal without noise and in the presence of noise is respectively shown in Figure 3.4. Based on the work presented in [63], the underlying model involves the estimation of a linear function of a variable $\beta \in \mathbb{R}^q$ which is additively affected by noise $\varepsilon_i$ in each of the $n$ sensors on the network belonging to the quadrotor sub-swarm. Then, the model could be described as follows,

$$\sigma_i = \mathcal{H}_i\beta + \varepsilon_i, \tag{3.4}$$

where $\sigma_i$, $\varepsilon_i \in \mathbb{R}^{p_i \times 1}$ and $\mathcal{H}_i \in \mathbb{R}^{p_i xq}$ in which each value of the vector $\sigma_i$ is a measurement channel of the $i^{th}$ sensor. Additionally, $\mathcal{H}_i$ is assumed to be of rank $q$, assuring that the measurements are not entirely redundant.

Taking into account that the Least Squares aim to minimize the error function, we can see in the centralized case by isolating the error variable $\varepsilon = \sigma - \mathcal{H}\beta$, and then calculating the square error as,

$$\varepsilon^2 = \varepsilon\varepsilon^\top = (\sigma - \mathcal{H}\beta)^\top(\sigma - \mathcal{H}\beta),$$

thus,

$$\varepsilon^\top\varepsilon = \sigma^\top\sigma - 2\sigma^\top\mathcal{H}\beta + \beta^\top\mathcal{H}^\top\mathcal{H}\beta,$$

where the result is a function $f(\beta)$ which as can be seen depends only on $\beta$. To find the local or global minimums, it is necessary to calculate the gradient error and the Hessian matrix, which in this case are the following,

$$\bigtriangledown f(\beta) = \frac{\mathrm{d}f}{\mathrm{d}\beta} = -2\mathcal{H}^\top\sigma + 2\mathcal{H}^\top\mathcal{H}\beta,$$

by isolating $\beta$, it is found a minimum when,

$$\hat{\beta} = (\mathcal{H}^\top\mathcal{H})^{-1}\mathcal{H}^\top\sigma,$$

furthermore, to prove the minimum function, this must satisfy,

$$\frac{\mathrm{d}^2f}{\mathrm{d}^2\beta} = 2\mathcal{H}^\top\mathcal{H} > 0,$$

taking as $u$ a non-null vector by multiplying $u^\top\mathcal{H}^\top\mathcal{H}u > 0 \rightarrow (\mathcal{H}u)^\top(\mathcal{H}u) \rightarrow \|\mathcal{H}u\|^2 > 0$, it is possible to conclude that due to the Hessian matrix is always positive because the function is convex, the critical point is not just a minimum is optimum.
Considering now the distributed sensor network, this can be modeled as,

$$\hat{\beta} = \left(\sum_{i=1}^{n} {\mathcal{H}_i}^\top\mathcal{H}_i\right)^{-1}\left(\sum_{i=1}^{n} {\mathcal{H}_i}^\top\sigma_i\right)$$

by solving the following distributed optimization function,

$min \sum_{i=1}^{n} f_i(\beta)$
$\quad s.t.\ \beta \in \mathbb{R}^q$

as was shown previously each $f_i \colon \mathbb{R}^q \to \mathbb{R}$ are convex functions as a result, the target function is given by the average of the gradient functions of each node as $f^* = \frac{1}{n} \sum_{j=1}^{n} (f_i)$, taking the same form of the following

$$\hat{\beta} = \frac{1}{n} \sum_{j=1}^{n} \sigma_i, \tag{3.5}$$

this concludes that a consensus among the sensors can be achieved if some conditions are satisfied. Let's consider the iterated functioning of a sensor as

$$\hat{\beta}_i(k+1) = \hat{\beta}_i(k) + \Delta \sum_{j \in N_i} w_{ij} (\hat{\beta}_j(k) - \hat{\beta}_i(k))$$

in which $\hat{\beta}_i(k)$ illustrate the estimation of the $i^{th}$ sensor of variable $\beta$ at iteration $k$,

$$\lim_{k \to \infty} \hat{\beta}_i(k) = \left( \frac{1}{n} \sum_{i=1}^{n} \sigma_i \right) \mathbf{1}, \tag{3.6}$$

if and only if the sensor network is connected and $\rho(L_w(G)) < \frac{2}{\Delta}$, where $\rho(L_w(G))$ corresponds to the maximum eigenvalue of the graph in absolute values, then the agents on the network will converge to the average of its initial conditions as shown below in Figure 3.5.

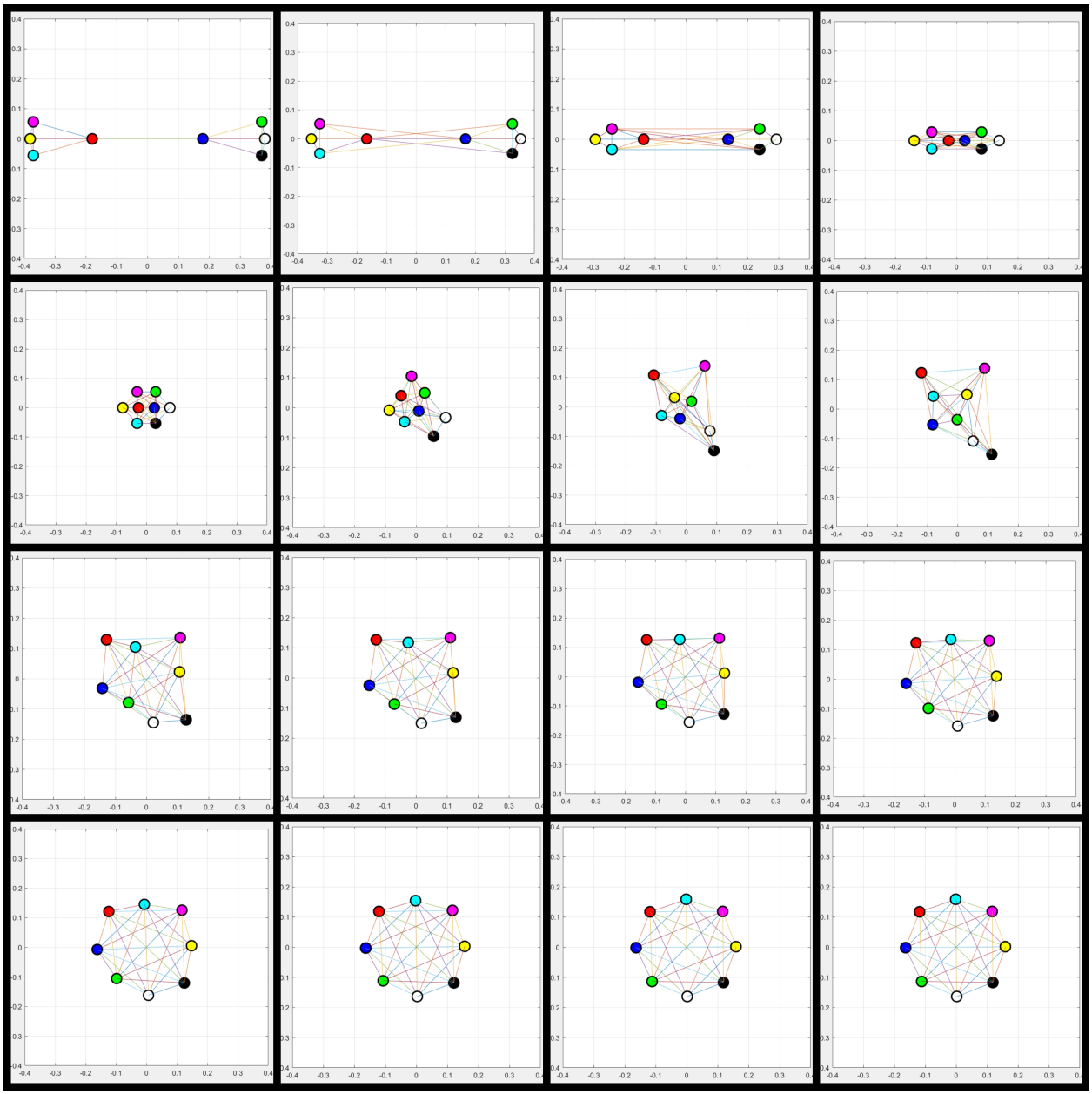


**Figure** 3.2: Consensus around of a circle.

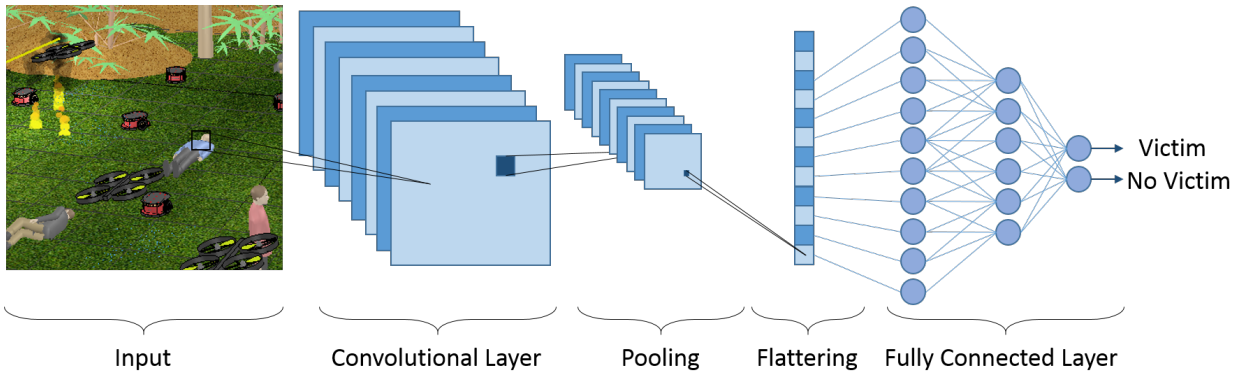
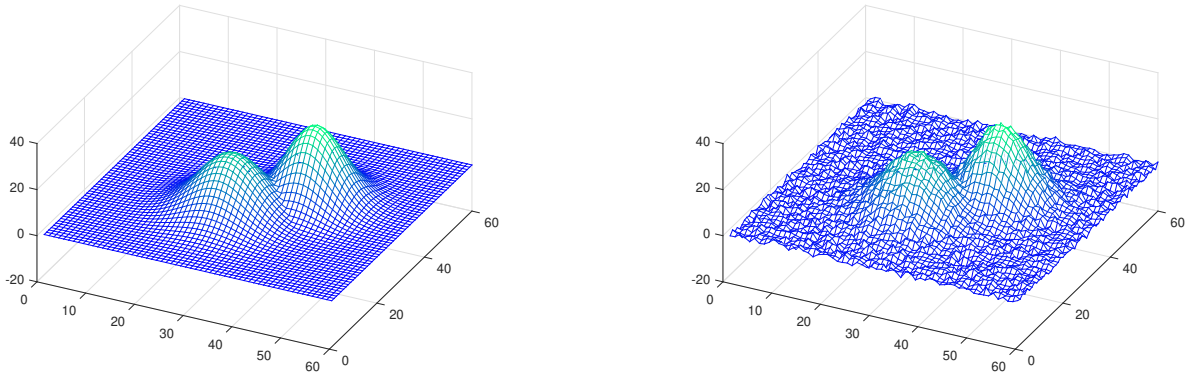


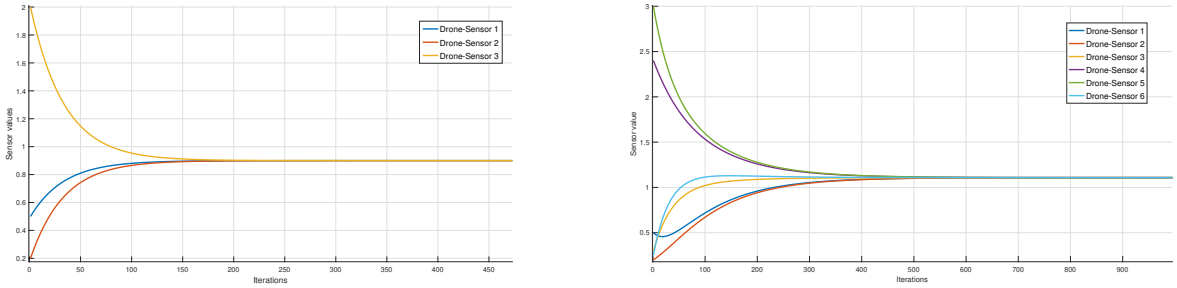

**Figure** 3.3: Digraph scheme.



(a) Sensing without noise in the signal.

(b) Sensing with noise in the signal.

**Figure** 3.4: Additive Gaussian noise applied to measurements in sensors.




(a) Sensor consensus with three agents.

(b) Sensor consensus with six agents.

**Figure** 3.5: Sensor network converging due to the consensus applied.

# 4 Cooperative Load Transportation

In recent years, accelerated technology development has made it possible to think about robotics optimizing search and rescue operations. Quadrotors have been demonstrated to be useful robots in many applications, such as surveillance, aerial construction, and agriculture. Its good maneuverability and mobility in a highly-constrained three-dimensional environment allow it to solve problems and tasks that neither humans nor other platforms do. Furthermore, using robots to do some tasks that rescuers normally do might avoid putting extra lives at risk, and also, in most cases, quadrotors can move faster than rescuers do, which possibly can improve the performance in the mission [7]. Tasks such as the delivery of first aid kits, extraction of survivors, and carrying rescuers' support devices are tasks that seem to be worthless. But, it might affect the chances that survivors in a disaster zone remain alive; time is precious when discussing search and rescue operations. All of these tasks involve at least one quadrotor lifting a load and carrying it to a determined place in the environment. This approach has been widely explored, considering the reduction of disturbances in the load, smooth maneuverability, optimal trajectories [29], using grippers, springs, and visual techniques [59]. Despite the great feasibility of using quadrotors to transport loads, quadrotors are usually limited in terms of their payload-carrying capacity. However, quadrotors can carry payloads beyond their limits by collaborating and performing cooperative transportation [72]. The same behavior is performed in ants when they cooperate in carrying food or elements to the nest that a single ant cannot handle. This is the reason why the objective of this chapter is to show how some ants express this behavior. And how inspired by this, it is possible to impose the desired behavior on a group of robots that cooperates to carry a suspended load which could be a victim, first aid kit, or devices needed in a search and rescue mission.

## 4.1. Ant Cooperative Load Transportation

Ants, renowned for their remarkable collective behaviors, have long fascinated researchers in the field of biology. Among their intriguing abilities is the cooperative transportation of heavy loads, where a group of ants collaboratively transports objects that would be impossible for an individual ant to carry alone. This unique behavior has captured the attention of scientists and engineers alike, inspiring investigations into the mechanisms underlying ant cooperative load transportation and its potential applications in various domains. Research in the field of ant cooperative load transportation has uncovered many intriguing findings.

In a study conducted by [61], researchers investigated the load-carrying strategies employed by leaf-cutter ants. By employing experimental setups that mimicked natural conditions, the researchers revealed how ants distribute the load among the group members, optimizing the transportation process. In addition, they found that ants exhibit a division of labor, with more giant ants taking on heavier loads and smaller ants responsible for clearing the pathway. This cooperative strategy enables the ants to overcome obstacles and transport loads efficiently.

Moreover, another research paper by [53] delved into the communication mechanisms employed by ants during load transportation. Through a series of experiments involving manipulations of the load and observation of ant behaviors, the researchers discovered that ants use a combination of tactile and chemical cues to coordinate their movements. They found that ants adjust their walking speed, direction, and force based on the interactions with their neighboring ants and the load, thus enabling the group to adapt to changing environmental conditions. The insights gained from these and other research papers on ant cooperative load transportation have significant implications beyond biology. One area that stands to benefit from the lessons learned from ant behavior is the field of multirobot load transportation, particularly in the context of aerial robots. Aerial robots, such as drones, are increasingly employed for various applications, including transportation tasks in urban environments, disaster response, and delivery services. However, the challenges associated with load transportation in aerial robotics, such as payload capacity, energy constraints, and coordination, necessitate innovative solutions. By leveraging the efficiency and adaptability observed in ant colonies, we can envision new strategies and algorithms that enhance the capabilities of aerial robots in load transportation tasks, ultimately leading to more efficient and reliable systems.

## 4.2. Geometric Control

Geometric control [49] is a field within control theory that applies geometric techniques to analyze and design control systems for various dynamical systems. It considers the system's state space and associated geometric structures to understand its behavior. Control-affine systems, which combine inherent dynamics and external inputs, are a key concept in geometric control. Researchers can analyze controllability, stability, and trajectory tracking using geometric methods like differential forms and Lie theory [44]. Geometric control theory also highlights the preservation of a system's geometric properties, allowing the design of control strategies that stabilize the system while respecting its intrinsic characteristics. In the case of quadrotors, we can utilize geometric control to develop effective control systems for their stabilization and maneuvering.

### 4.2.1. Agent Model

*Notation and Definitions:* $\{X_w, Y_w, Z_w\}$ unit vectors along the axis of $\{W\}$, $\{X_{B_i}, Y_{B_i}, Z_{B_i}\}$ unit vectors along the axis of the $i^{th}$ quadrotor $\{B_i\}$ with respect to $\{W\}$, $m_{q_i} \in \mathbb{R}_{>0}$ mass of the $i^{th}$ quadrotor, $J_i \in \mathbb{R}^{3\times3}$ inertia matrix of the $i^{th}$quadrotor with respect to $\{B_i\}$, $r_{q_i}$, $v_{q_i} \in \mathbb{R}^3$ position and velocity of the $i^{th}$ quadrotor center-of-mass in $\{W\}$ $R_i \in \mathbb{R}^3$ the rotation matrix from $\{B_i\}$ to$\{W\}$, $\Omega_i \in \mathbb{R}^3$ angular velocity of $i^{th}$ quadrotor in $\{B_i\}$, $F_i \in \mathbb{R}$ total thrust produced by the $i^{th}$ quadrotor $M_i \in \mathbb{R}^3$ moment produced by the $i^{th}$ quadrotor, $m_l \in \mathbb{R}_{>0}$ mass of the load, $r_l$, $v_l \in \mathbb{R}^3$ position and velocity of the load in $\{W\}$ $l_i \in \mathbb{R}_{>0}$ length of the cable from load to $i^{th}$ quadrotor, $T_i \in \mathbb{R}_{\geq0}$ tension on the cable produced by the $i^{th}$ quadrotor and the load.

#### Multi-Quadrotor Dynamics

The localization of each quadrotor UAV is defined by the position of its center of mass (CoM) and the attitude with respect to the $\{W\}$ inertial frame. The multi-robot UAV's motion can be modeled by applying the Newton-Euler approach to summate forces and torques concerning the system.

$$\begin{aligned} \dot{r}_{q_i} &= v_{q_i}, \\ m_{q_i}\dot{v}_{q_i} &= m_{q_i}g(-Z_w) + F_i(Z_{B_i}), \\ \dot{R}_i &= R_i\hat{\Omega}_i, \\ J_i\dot{\Omega}_i &= -\Omega_i \times J_i\Omega_i + M_i, \end{aligned} \tag{4.1}$$

where state variables are defined previously in Notation and Definition. Additionally, $Z_{B_i} = R_iZ_w$ in which $Z_w = u_3$ with $u_3 = [0\,0\,1]^\top$, $g$ is the gravitational acceleration constant applied in $Z_w$ and the hat map $\hat{\cdot}\colon \mathbb{R}^3 \to SO(3)$ is the skew-symmetric operator matrix as explained in [60] such that, $\hat{x}y = x \times y \;\; \forall x, y \in \mathbb{R}^3$, in which $w_i = [w_1, w_2, w_3]$ then,

$$\hat{W} = \begin{pmatrix} 0 & -w_3 & w_2 \\ w_3 & 0 & -w_1 \\ -w_2 & w_1 & 0 \end{pmatrix}, \tag{4.2}$$

here is assumed that each propeller of the quadrotors is directly controlled, avoiding the consideration of the motor and propeller dynamics. Also, the direction of the thrust is considered normal to the quadrotor plane. The total thrust produced by the quadrotors acting in $Z_w$ is the result of the summation of the thrust of each propeller in the quadrotor as $F_i = \sum_{k=1}^{4} -f_k$. Assuming the torque generated by each propeller proportionally to the thrust reached. Since if the quadrotor wants to gain altitude, it requires having a positive thrust, so, the first and third propellers must rotate clockwise, while the second and fourth propellers must rotate counterclockwise as shown in Figure 4.1. The thrust of each propeller $f_i$ makes the quadrotor move in $(-Z_w)$, the torque generated by the $k^{th}$ propeller of the $i^{th}$

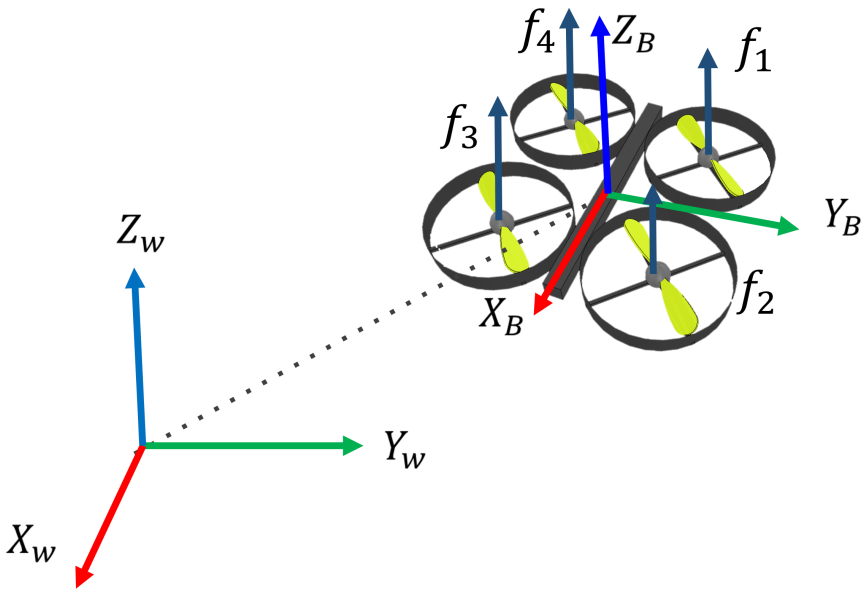



**Figure** 4.1: Quadrotor model: in the robot reference frame to the inertial reference frame.

quadrotor can be written as $\lambda_k = (-1)^k c_{\lambda f} f_i$, in which $c_{\lambda f}$ is a constant given by the design of the propellers. Here, the linear rotor dynamics can be included, leading to the result of the total thrust and the total moment of the quadrotor as,

$$\begin{bmatrix} F_i \\ M_{1_i} \\ M_{2_i} \\ M_{3_i} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -d & 0 & d \\ d & 0 & -d & 0 \\ -c_{\lambda f} & c_{\lambda f} & -c_{\lambda f} & c_{\lambda f} \end{bmatrix} \begin{bmatrix} f_{i_1} \\ f_{i_2} \\ f_{i_3} \\ f_{i_4} \end{bmatrix}.$$

**Multi-Quadrotor Dynamics Lifting a Load.**

Following the same procedure described previously, it is possible to identify the system's changes when the quadrotors start the lifting maneuver. We obtain the dynamics of the system as follows

$$\begin{aligned} \dot{r}_{q_i} &= v_{q_i}, \\ m_{q_i}\dot{v}_{q_i} &= m_{q_i} g(-Z_w) + F_i(Z_{B_i}), -T_i(\mu_i) \\ \dot{R}_i &= R_i \hat{\Omega}_i, \\ J_i \dot{\Omega}_i &= -\Omega_i \times J_i \Omega_i + M_i, \\ \dot{r}_l &= v_l, \\ m_l \dot{v}_l &= \sum T_i(\mu_i) - m_l g(Z_w) \end{aligned} \tag{4.3}$$

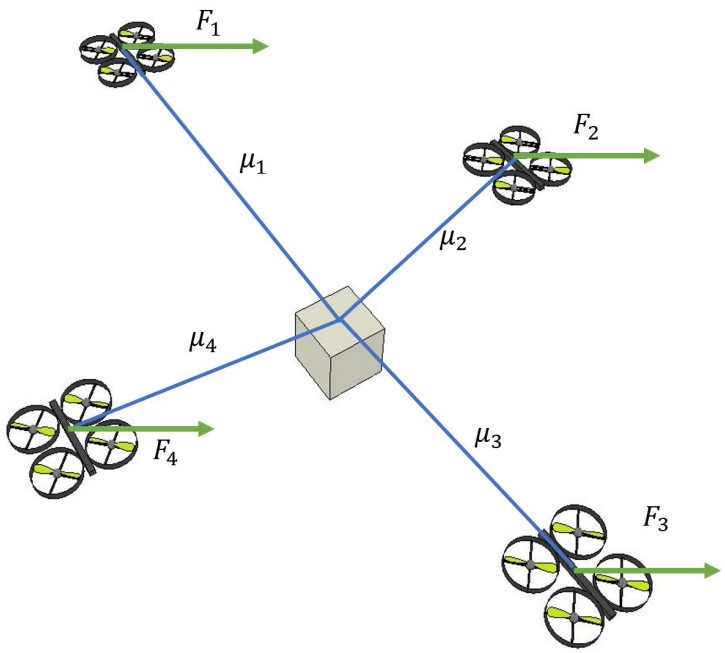



**Figure** 4.2: Cooperative transportation: carrying a load by cooperating with a group of quadrotors.

where $\mu_i$ is the unit vector that goes from the load towards the $i^{th}$ quadrotor, also considering that the position of the load $r_l$ can be determined by the following expression

$$r_l = r_{q_i} - l_i \mu_i.$$

**Cooperative Transport Approach.** Quadrotors $i$ and $j$ transporting an object exert forces $F_i$, $F_j$ (for robots attached at the $i^{th}$ and $j^{th}$ contact points, we always assume $i$ and $j$ positionally opposed to each other). The vectors to these contact positions (in the body fixed frame) are expressed as $\mu_i$, $\mu_j$ as shown in Fig. 4.2. These forces are summed (over $N$ robots) to give the net force on the object. Let $\gamma$ be the vector of net external forces on the body, then for the planar case (altitude remaining as constant), $\gamma_x = \sum_{i=1}^{N} F_{i,x}$, $\gamma_y = \sum_{i=1}^{N} F_{i,y}$. In this kind of problem is shown in [52] that if there is no interaction force then the following equation holds

$$(F_i - F_j) \cdot (\mu_i - \mu_j) = 0, \tag{4.4}$$

this is called the Zero Interaction Force Condition (ZIF), one of the control objectives.

## 4.2.2. Geometric Control Algorithm Design

### Position Based Passivity Controller

As the foregoing mentioned, this controller works to find the desired position at the end of the first sub-trajectory that places the robots in good positions to lift the load. First, a matrix that captures the interaction dynamics of the system is defined, called the incidence matrix, which is filled as follows

$$d_{lk}(G): = \begin{cases} +1 \text{ if } k \in L_i^+ \\ -1 \text{ if } k \in L_i^-, \\ 0 \text{ otherwise} \end{cases}$$

where $L_i^+$ and $L_i^-$ represents if the link $k^{th}$ points in or out of node $i^{th}$ respectively. The dynamic model employed for the target positions that will be delivered to the trajectory generator is a double integrator model. The double integrator dynamics takes the form

$$u_i = m_i \dot{v}_{q_i}, \;\; i = 1, ..., N, \tag{4.5}$$

where $u_i$ is the force input control, $m_i$ is the mass of agent $i^{th}$ and $v_{q_i}$ is the velocity of agent $i^{th}$. When the control signal is added, the dynamic equation can be expressed as

$$u_i = -k_i(v_{q_i} - v_r) + m_i \dot{v}_r + u_i^d, \; k_i \geq 0, \tag{4.6}$$

where $v_r$ is the common and reference velocity, $k_i$ is a proportional controller. We consider a group of agents modeled like the equation (4.5), the control law that is implemented to obtain the function in (4.6) seeks to guarantee that $(\|\dot{r}_{q_i} - v_r\|) \to 0$ and $\Delta_k \to \Delta_k^d$. According to Corollary 2.1 in [3], the global asymptotic stability of the system is guaranteed by

$$\dot{r}_{q_i} = H_i \left( -\sum_{i=1}^{k} d_{ik} \psi_k(\Delta_k) \right) + v_r, \; i = 1, ..., N,$$

where $H_i$ denotes the output at time $t$ of a static or dynamic block satisfying $|\dot{r}_q - v_r)| \to 0$ and $(r_{q_i} - r_{q_j}) \to 0$ as $t \to \infty$ for every pair of nodes $(v_i, v_j)$ which are connected by a path. $\Delta_k$ denotes the differences between $r_{q_i}$ and its neighbors $r_{q_j}$. $\psi_k : \mathbb{R}^p \to \mathbb{R}^p$ are non-linearities designed such that the target set where the dynamics evolve invariant and asymptotically stable. In this work the target set can be defined as $\Lambda = \left\{ \Delta_k \;\; | \; \Delta_k = \Delta_k^d \right\}$ where $\Delta_k^d$ dictates the desired relative configuration of the nodes. The control law, then, is

$$u_i^d = -\sum_{i=1}^{k} d_{ik} \psi_k(\Delta_k - \Delta_k^d),$$

giving as a result of replacing in (4.6) as

$$m_i(\ddot{r}_{q_i} - \dot{v}_r) + k_i(\dot{r}_{q_i} - v_r) + \sum_{i=1}^{k} d_{ik}\psi_k(\Delta_k - \Delta_k^d) = 0, \tag{4.7}$$

the consensus protocol starts from a potential quadratic function taking into account $\psi_k(\Delta_k) = \nabla P_k(\Delta_k)$ and is related to (4.7) as

$$P_k = \frac{\delta_k}{2}\left|\Delta_k - \Delta_k^d\right|^2,\ \delta_k \in \mathbb{R}_{>0}$$

$$\psi_k(\Delta_k) = \delta_k(\Delta_k - \Delta_k^d),$$

the $\delta_k$ constants are the system feedback gains that regulate the difference between the desired set and current system conditions. According to our requirements, $\Delta_k$ is chosen in the form that the desired shape is a rhombus. Where quadrotors 1 and 3 are aligned and positional opposed in relation to the load and quadrotors as quadrotors 2 and 4.

### Geometric Control $F_i$

As a result of (4.3), the control variables are two. First, $F_i$ corresponds to the total thrust produced by the four motors in the $i^{th}$ quadrotor. Second, the moments executed in each axis $M_i$ that will be shown in the next subsection. As mentioned in Section 1, geometric control is implemented as depicted in [55]. The procedure begins by defining the position and velocity errors as $e_{p_i} = r_{q_i} - r_i^{des}$ and $e_{v_i} = v_{q_i} - \dot{r}_i^{des}$ respectively, in this way the desired thrust takes the form as follows.

$$F_i^{des} = -K_p e_{p_i} - K_v e_{v_i} + m_{v_i}(gu_3 + \ddot{r}_i^{des}) - K_g((F_i - Fj)\cdot(\mu_i - \mu_j)), \tag{4.8}$$

where $m_{v_i} = m_{q_i} + m_l$, $K_p$ and $K_v$ are diagonal gain matrices and $\left\|F_i^{des}\right\| \neq 0$ at any time, but the motion of the quadrotor is controlled in the projection of $F_i^{des}$ on to the third body-axis giving, as a result,

$$F_i = F_i^{des} \cdot Z_{B_i}.$$

### Geometric Control $M_i$

On the other hand, the variable responsible to control the altitude error and the angular velocity is $M_i$. In the case of the altitude error that stabilizes the translational movement can be chosen as

$$Z_{B_i}^{des} = \frac{F_i^{des}}{\left\|F_i^{des}\right\|},$$

taking $F_i^{des}$ from (4.8). Here the yaw angle of the quadrotor remains at zero, allowing to choose $X_{B_i}^{des} = [1\,0\,0]^\top$, obtaining a desired pose as

$$R_i^{des} = \left[Y_{B_i}^{des} \times Z_{B_i}^{des}\ Y_{B_i}^{des}\ Z_{B_i}^{des}\right] \in SO(3),$$

where

$$Y_{B_i}^{des} = \frac{Z_{B_i}^{des} \times X_{B_i}^{des}}{\left\| Z_{B_i}^{des} \times X_{B_i}^{des} \right\|},$$

and finally, defining the pose error as

$$e_{R_i} = \frac{1}{2}\left((R_i^{des})^\top R_i - R_i^\top R_i^{des}\right)^\vee, \tag{4.9}$$

where $\vee$ represents the vee operator, which is the inverse of the skew-symmetric operator defined from (4.16) as

$$w^\vee = \begin{pmatrix} -W_{23} \\ W_{13} \\ -W_{12} \end{pmatrix}.$$

As mentioned, under the assumption to remain yaw angle constant, the angular velocity can be defined as was made in [62], $\Omega_i^{des} = \rho_i^{des} X_{B_i}^{des} + \varrho_i^{des} Y_{B_i}^{des}$, where $\rho_i^{des} = -h_\omega \cdot Y_{B_i}^{des}$ and $\varrho_i^{des} = h_\omega \cdot X_{B_i}^{des}$ where

$$h_\omega = \frac{m_v}{F_i}(\ddot{r}_i^{des} - (Z_{B_i}^{des} \cdot \ddot{r}_i^{des}) Z_{B_i}^{des}),$$

thus the angular velocity error is

$$e_{\Omega_i} = \Omega - \Omega_i^{des}, \tag{4.10}$$

then in the same way as in the geometric control of $F_i$, the control of $M_i$ taking (4.108) and (4.10) is as follows

$$M_i = -K_R e_{R_i} - K_\Omega e_{\Omega_i} + (\Omega_i \times J_i \Omega_i),$$

again, $K_R$ and $K_\Omega$ are diagonal gain matrices that guarantee the system's stability.

## 4.3. Adaptive Control

### 4.3.1. Agent Model

*Notation and Definitions:* The notation used for matrices and vectors is $X$ and $x$, respectively. We write $X^\top$ and $x^\top$ for the transpose of a matrix or a vector. The Euclidean norm is defined

as $\|X\|^2 = \sum_{i=1}^{n} |x_i|^2$. We denote $A^{-\top} = (A^{\top})^{-1}$ as the inverse of a transposed matrix. The trace of a matrix is $\mathrm{tr}(X)$, where $X$ is square. A directed graph is defined as the pair $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of graph nodes, and $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$ is the set of communication edges. The adjacency matrix is defined as $A = [a_{ij}]$ where $a_{ii} = 0$ and $a_{ij} = 1$ if $(j, i) \in \mathcal{E}$, where $i \neq j$. Regarding the tension $T$, it is modeled as the opposition force of the thrust generated by the quadrotor in each axis. This opposition is represented by the well-known spring and damper system, where the equation that governs the tension is $T_i(\mu_i) = -kx - b\dot{x} + \eta$, $x$ is a vector that contains the $x_W$, $y_W$, and $z_W$ axis.

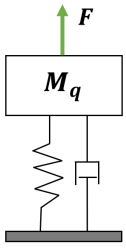



**Figure** 4.3: Tension modeled as a Spring-Damping system.

Getting as a result for the altitude dynamic of the form $\dot{x}_i = A_i x_i + (b_i + f_i(x_i))u_i + \eta_i$ for each quadrotor as

$$\dot{x}_i = \begin{bmatrix} 0 & 1 \\ -\frac{k_i}{m_{q_i}} & -\frac{b_i}{m_{q_i}} \end{bmatrix} x_i + \left( \begin{bmatrix} 0 \\ 1 \end{bmatrix} + f_i(x_i) \right) F_i + \eta_i, \tag{4.11}$$

where $f_i(x_i)$ is a bounded input uncertainty, $\eta_i$ is an unstructured bounded dynamic uncertainty, $x_i \in \mathbb{R}^2$ are the agent's states, $u_i \in \mathbb{R}^2$ are its thrust input, $A_i$ is an unknown matrix related to the agent's states, $b_i$ are known vectors with heterogeneous quadrotors ($A_i \neq A_j$ and $b_i \neq b_j$). This model works for the $x_W$ and $y_W$ axis. In the case of the $z_W$ axis, the term of gravity has to be added.

### 4.3.2. Adaptive $\sigma$-Modification for State Synchronization

Taking into account that we are working with a DMRAC, the model of the reference is described as

$$\dot{x}_m = A_m x_m + b_m r, \tag{4.12}$$

where $x_m \in \mathbb{R}^2$ is the state, $r \in \mathbb{R}$ is the reference, and $A_m$ and $b_m$ are matrices of the reference model. The graph used is shown in Figure 4.5, which conserves the disposition shown in Figure 4.2, but the links represent communication relations.

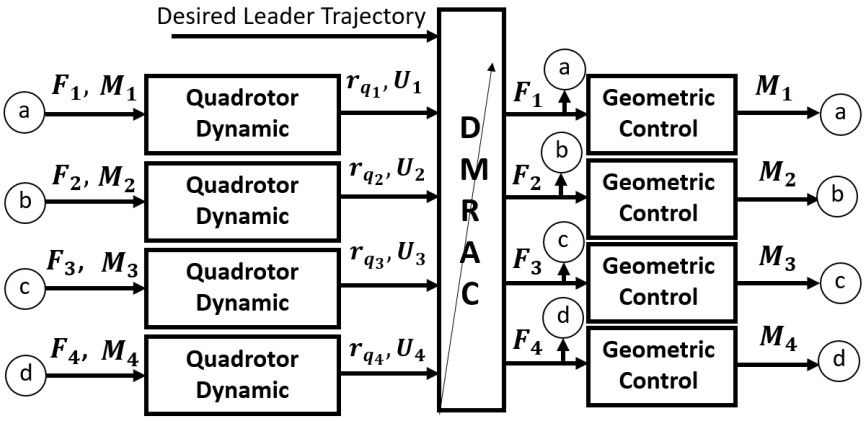



**Figure** 4.4: Control Scheme.

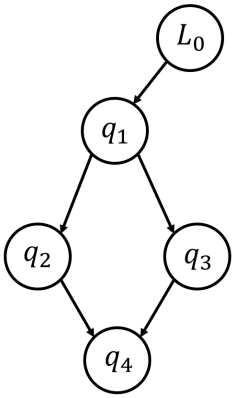



**Figure** 4.5: Leader-follower communication graph.

The following assumptions are made to accomplish a matching dynamic to guarantee a cooperative MRAC and replicate the behavior of the leader or a neighbor, depending on the case.

**Assumption 1.** The vector $k_{mi}^*$ and the scalar $k_{ri}^*$ exist and are defined as

$$A_m = A_i + b_i k_{mi}^{*\top}, \tag{4.13}$$

$$b_m = b_i k_{ri}^*. \tag{4.14}$$

Constants in (4.14) are known as feedback-matching conditions.

**Assumption 2.** The vector $k_{ij}^*$ and the scalar $l_{ij}^*$ exists and are defined such that

$$A_i = A_j + b_j k_{ij}^{*\top}, \tag{4.15}$$

$$b_i = b_j k_{rij}^*, \tag{4.16}$$

constants $k_{ij}^*$ and $k_{rij}^*$ in (4.16) are known as coupling matching conditions as in [4]. Focusing on the synchronization from quadrotor $q_1$ to the leader dynamics in the graph depicted in

Figure 4.5, the following control law is proposed

$$F_1 = k_m x_m + k_r r - u_{ad} + k_g \Delta_1, \tag{4.17}$$

where $k_g$ is a positive constant, $k_m$, and $k_r$ are adaptive gain vectors obtained from feedback matching conditions as

$$\dot{k}_m^\top = -\text{sgn}\left(k_r{}^*\right) \gamma \, b_m^\top P \left(x_1 - x_m\right) x_1^\top, \tag{4.18}$$

$$\dot{k}_r = -\text{sgn}\left(k_r{}^*\right) \gamma \, b_m^\top P \left(x_1 - x_m\right) r, \tag{4.19}$$

where the scalar $\gamma > 0$ is the adaptive gain, and $A_m$ is designed to guarantee that all of its eigenvalues have negative real parts, such that there exists a definite positive matrix $P \in \mathbb{R}^{2\times 2}$ and $Q \in \mathbb{R}^{2\times 2}$ in which the following condition is satisfied

$$P A_m + A_m^\top P = -Q, \quad Q > 0, \tag{4.20}$$

and $u_{ad} \in \mathbb{R}$ is a direct adaptive signal parameterized by linear-in-parameters input uncertainty

$$u_{ad} = \theta^\top \phi(x), \tag{4.21}$$

where $\phi : \mathbb{R}^2 \to \mathbb{R}^2$ is a known regressor vector, $\theta \in \mathbb{R}^{2\times 1}$ is the $\sigma$-modification based on [48] defined as

$$\dot{\theta} = -\gamma \left(\phi {e_1}^\top P b_1 + \sigma \theta\right). \tag{4.22}$$

where $\sigma$ is a constant satisfying $\sigma > 0$, and $e = x_m - x_1$ is the synchronization error.
Once the synchronization dynamic from quadrotor 1 to the leader is achieved, we consider the case with a third agent which does not have direct communication with the leader. Then, the proposed control laws for quadrotors 2 and 3 to achieve synchronization without having communication with the reference model are defined as

$$F_2 = k_{21}^\top x_1 + k_{m2}^\top \left(x_2 - x_1 - \delta_2\right) + k_{r21} u_1 - \theta_2^\top \phi_2 + k_g \Delta_2. \tag{4.23}$$

$$F_3 = k_{31}^\top x_1 + k_{m3}^\top \left(x_3 - x_1 - \delta_3\right) + k_{r31} u_1 - \theta_3^\top \phi_3 + k_g \Delta_3, \tag{4.24}$$

where $\delta_i$ are constant values adjusted to reach a desired distance between quadrotors in the $x_W$ and $y_W$ axis. For the case of the $z_W$ axis, the term is ignored due to the desired behavior to achieve a consensus. The adaptive laws for quadrotor 2 are

$$\dot{k}_{21}^\top = -\text{sgn}\left(k_{r2}^*\right) \gamma \, b_m^\top P \left(x_2 - x_1\right) x_2^\top, \tag{4.25}$$

$$\dot{k}_{m2}^\top = -\text{sgn}\left(k_{r2}^*\right) \gamma \, b_m^\top P \left(x_2 - x_1\right) \left(x_2 - x_1\right)^\top, \tag{4.26}$$

$$\dot{k}_{r21} = -\text{sgn}\left(k_{r2}^*\right) \gamma \, b_m^\top P \left(x_2 - x_1\right) u_1, \tag{4.27}$$

$$\dot{\theta}_2 = -\gamma \left(\phi_2 (x_2 - x_1)^\top P b_2 + \sigma \theta_2\right), \tag{4.28}$$

and for quadrotor 3

$$\dot{k}_{31}^{\top} = -\text{sgn}\left(k_{r3}^{*}\right)\gamma\, b_m^{\top} P\left(x_3 - x_1\right) x_3^{\top}, \tag{4.29}$$
$$\dot{k}_{m3}^{\top} = -\text{sgn}\left(k_{r3}^{*}\right)\gamma\, b_m^{\top} P\left(x_3 - x_1\right)\left(x_3 - x_1\right)^{\top}, \tag{4.30}$$
$$\dot{k}_{r31} = -\text{sgn}\left(k_{r3}^{*}\right)\gamma\, b_m^{\top} P\left(x_3 - x_1\right) u_1, \tag{4.31}$$
$$\dot{\theta}_3 = -\gamma\left(\phi_3 (x_3 - x_1)^{\top} P b_3 + \sigma\theta_3\right). \tag{4.32}$$

The last quadrotor represents a particular case where the dynamic has information from neither the leader nor the quadrotor 1. So errors $e_{42} = x_4 - x_2 - \delta_{42}$ and $e_{43} = x_4 - x_3 - \delta_{43}$ have the following dynamics

$$\dot{e}_{42} = A_m e_{42} + b_4\left(u_4 - k_{m4}^{*\top} x_2 - k_{42}^{*\top} e_{42} - k_{r4}^{*\top} u_2 + \theta_4^{\top}\phi_4\right), \tag{4.33}$$
$$\dot{e}_{43} = A_m e_{43} + b_4\left(u_4 - k_{m4}^{*\top} x_3 - k_{43}^{*\top} e_{43} - k_{r4}^{*\top} u_3 + \theta_4^{\top}\phi_4\right), \tag{4.34}$$

which allows us to obtain the following controller

$$F_4 = k_{42}^{\top}\frac{x_2}{2} + k_{43}^{\top}\frac{x_3}{2} + k_{m4}^{\top}\frac{e_{42} + e_{43}}{2} + k_{r42}^{\top}\frac{u_2}{2} + k_{r43}^{\top}\frac{u_3}{2} - \frac{\theta_4^{\top}\phi_4}{2} + k_g\Delta_4, \tag{4.35}$$

with adaptive laws

$$\dot{k}_{42}^{\top} = -\text{sgn}\left(k_{r4}^{*}\right)\gamma\, b_m^{\top} P\left(e_{42} + e_{43}\right) x_2^{\top}, \tag{4.36}$$
$$\dot{k}_{43}^{\top} = -\text{sgn}\left(k_{r4}^{*}\right)\gamma\, b_m^{\top} P\left(e_{42} + e_{43}\right) x_3^{\top}, \tag{4.37}$$
$$\dot{k}_{m4}^{\top} = -\text{sgn}\left(k_{r4}^{*}\right)\gamma\, b_m^{\top} P\left(e_{42} + e_{43}\right)\left(e_{42} + e_{43}\right)^{\top}, \tag{4.38}$$
$$\dot{k}_{r42} = -\text{sgn}\left(k_{r4}^{*}\right)\gamma\, b_m^{\top} P\left(e_{42} + e_{43}\right) u_2, \tag{4.39}$$
$$\dot{k}_{r43} = -\text{sgn}\left(k_{r4}^{*}\right)\gamma\, b_m^{\top} P\left(e_{42} + e_{43}\right) u_3, \tag{4.40}$$
$$\dot{\theta}_4 = -\gamma\left(\phi_4 (e_{42} + e_{43})^{\top} P b_4 + \sigma\theta_4\right). \tag{4.41}$$

These results allow the system to adapt to a reference given to a leader even when they do not have direct communication with it. The following theorem is the main contribution, and it captures the general form of any heterogeneous number of agents that synchronize with a leader to cooperate in the task. It is used as a DMRAC using $\sigma$-modification to overcome possible uncertainties in its dynamics and in the cable tensions and disturbances such as wind and minimize error synchronization.

**Theorem 1.** Consider $N$ agents with dynamics (4.3), where only the quadrotor one directly communicates with the reference. The other quadrotors employ the following control law.

$$F_i = \frac{\sum_{j=1}^{N} a_{ij} k_{ij}^{\top} x_i}{\sum_{j=1}^{N} a_{ij}} + \frac{k_{mi}\sum_{j=1}^{N} a_{ij}(x_i - x_j - \delta_i)}{\sum_{j=1}^{N} a_{ij}} + \frac{\sum_{j=1}^{N} a_{ij} k_{rij} u_i}{\sum_{j=1}^{N} a_{ij}} - \frac{\sum_{j=1}^{N} a_{ij}\theta_i^{\top}\phi_i}{\sum_{j=1}^{N} a_{ij}} + \frac{\sum_{j=1}^{N} a_{ij} k_g \Delta_i}{\sum_{j=1}^{N} a_{ij}}, \tag{4.42}$$

with adaptive laws

$$\dot{k}_{ij}^{\top} = -\operatorname{sgn}(k_{ri}^{*})\gamma\, b_m^{\top} P \left[\sum_{j=1}^{N} a_{ij}(x_i - x_j)\right] x_i^{\top}, \tag{4.43}$$

$$\dot{k}_{mi}^{\top} = -\operatorname{sgn}(k_{ri}^{*})\gamma\, b_m^{\top} P \left[\sum_{j=1}^{N} a_{ij}(x_i - x_j)\right] \left[\sum_{j=1}^{N} a_{ij}(x_j - x_i)\right]^{\top}, \tag{4.44}$$

$$\dot{k}_{rij} = -\operatorname{sgn}(k_{ri}^{*})\gamma\, b_m^{\top} P \left[\sum_{j=1}^{N} a_{ij}(x_i - x_j)\right] u_i. \tag{4.45}$$

The matched uncertainty parameter is taken from (4.21), with the $\sigma$-modification parameter defined as

$$\dot{\theta}_i = -\gamma \left(\phi_i (x_i - x_j)^{\top} P b_i + \sigma \theta_i\right), \tag{4.46}$$

then, the control law will synchronize all the quadrotors to the reference leader with a bounded error.

*Proof of Theorem 1:* We should guarantee that all agent synchronization errors are bounded, regardless of whether they are communicated to a reference model. For this, the following Lyapunov function is used

$$V(e_{ij}, \tilde{k}_{mi}, \tilde{k}_{rij}, \tilde{k}_{ij}, \tilde{\theta}_i, \tilde{\Delta}_i) = \sum_{i=1}^{N} \left[\sum_{j=0}^{N} a_{ij} e_{ij}\right]^{\top} P \left[\sum_{j=0}^{N} a_{ij} e_{ij}\right] + \sum_{j=1}^{N} \operatorname{tr}\left[\frac{\tilde{k}_{mi}^{\top} \tilde{k}_{mi}}{\gamma\, |k_{ri}^{*}|}\right] + \dots \tag{4.47}$$

$$\dots + \sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij}\, \operatorname{tr}\left[\frac{\tilde{k}_{ij}^{\top} \tilde{k}_{ij}}{\gamma\, |k_{ri}^{*}|}\right] + \sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij} \frac{\tilde{k}_{ri}^{2}}{\gamma\, |k_{r}^{*}|} + \sum_{i=1}^{N} \operatorname{tr}(\tilde{\theta}_i^{\top} \gamma^{-1} \tilde{\theta}_i) + \sum_{i=1}^{N} \operatorname{tr}(\Delta_i^{\top} \gamma^{-1} \Delta_i), \tag{4.48}$$

where $j = 0$ is used as a representation of the reference, $\tilde{k}_{mi} = k_{mi} - k_{mi}^{*}$, $\tilde{k}_{rij} = k_{rij} - k_{rij}^{*}$, $\tilde{k}_{ij} = k_{ij} - k_{ij}^{*}$, $\tilde{\theta}_i = \theta_i - \theta_i^{*}$, $\tilde{\Delta}_i = \Delta_i - \Delta_i^{*}$, $\eta_{ij} = \eta_i - \eta_j$, the error dynamics represented as $e_i = x_i - x_m$, and $e_{ij} = x_i - x_j$ in short notation, and in long notation as

$$\dot{e}_{ij} = A_m(x_i - x_j) + b_j[u_i - k_{ij}^{*\top} x_i - k_{mi}^{*\top} - k_{rij}^{*} u_i + \tilde{\theta}_j^{\top} \phi_j] - \eta_{ij}, \tag{4.49}$$

$$\dot{e}_{ij} = A_m e_{ij} + b_i[\tilde{k}_{ij}^{\top} x_i + \tilde{k}_{mi}^{\top} e_{ij} + \tilde{k}_{rij}^{\top} u_i - \tilde{\theta}_i^{\top} \phi_i - \tilde{\theta}_j^{\top} \phi_j] - \eta_{ij}. \tag{4.50}$$

The derivative of (4.48) can be obtained as

$$\dot{V} = \sum_{i=1}^{N}\left[\sum_{j=0}^{N} a_{ij}e_{ij}\right]^{\top}(PA_m + A_m^{\top}P)\left[\sum_{j=0}^{N} a_{ij}e_{ij}\right] + \ldots$$
$$\ldots + 2\left[\sum_{j=0}^{N} a_{ij}e_{ij}\right]^{\top} Pb_i\left[\sum_{i=1}^{N} a_{ij}\tilde{k}_{ij}^{\top}x_i + \tilde{k}_{mi}^{\top}\sum_{i=1}^{N} a_{ij}e_{ij} + \sum_{i=1}^{N} a_{ij}\tilde{k}_{rij}u_i - \theta_i^{\top}\phi_i\right] - \ldots$$
$$\ldots - 2\left[\sum_{j=0}^{N} a_{ij}e_{ij}\right]^{\top} P\eta_{ij} + \sum_{i=1}^{N}\operatorname{tr}\left(\frac{\tilde{k}_{mi}^{\top}\gamma^{-1}\dot{\tilde{k}}_{mi}}{|k_{ri}^{*}|}\right) + \sum_{i=1}^{N}\operatorname{tr}\left(\frac{\tilde{k}_{ij}^{\top}\gamma^{-1}\dot{\tilde{k}}_{ij}}{|k_{ri}^{*}|}\right) + \ldots$$
$$\ldots + \sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij}\frac{\tilde{k}_{rij}\gamma^{-1}\dot{\tilde{k}}_{rij}}{|k_{ri}^{*}|} - 2\sum_{i=1}^{N}\sum_{j=1}^{N}\operatorname{tr}\left(\tilde{\theta}_i^{\top}\phi_i[e_{ij}^{\top}Pb_i + \sigma\theta_i]\right) + \sum_{i=1}^{N}\operatorname{tr}\left(\frac{\tilde{\Delta}_i^{\top}\gamma^{-1}\dot{\tilde{\Delta}}_i}{|\Delta_i^{*}|}\right),$$

reducing the Lyapunov equation

$$\dot{V} = -\sum_{i=1}^{N}\left[\sum_{j=0}^{N} a_{ij}e_{ij}\right]^{\top} Q\left[\sum_{j=0}^{N} a_{ij}e_{ij}\right] - 2\left[\sum_{j=0}^{N} a_{ij}e_{ij}\right]^{\top} P\eta_{ij} - 2\sigma\sum_{i=1}^{N}\sum_{j=1}^{N}\operatorname{tr}\left(\tilde{\theta}_i^{\top}\theta_i\right). \quad (4.51)$$

hence

$$\dot{V} = -\sum_{i=1}^{N}\left[\sum_{j=0}^{N} a_{ij}e_{ij}\right]^{\top} Q\left[\sum_{j=0}^{N} a_{ij}e_{ij}\right] - 2\left[\sum_{j=0}^{N} a_{ij}e_{ij}\right]^{\top} P\eta_{ij} - 2\sigma\sum_{i=1}^{N}\sum_{j=1}^{N}\operatorname{tr}\left(\tilde{\theta}_i^{\top}\tilde{\theta}_i + \tilde{\theta}_i^{\top}\theta_i^{*}\right). \quad (4.52)$$

therefore $\dot{V}$ is bounded by

$$\dot{V} \leq -\sum_{i=1}^{N}\lambda_{\text{mín}}(Q)\sum_{j=1}^{N}\|e_{ij}\|^2 + 2\sum_{i=1}^{N}\sum_{j=1}^{N}\|e_{ij}\|\lambda_{\text{máx}}(P)\eta_{0ij} - 2\sum_{i=1}^{N}\sigma\|\tilde{\theta}_i\|^2 + 2\sum_{i=1}^{N}\sigma\|\tilde{\theta}_i\|\theta_{0i}, \quad (4.53)$$

where $\eta_{0ij} = \|\sup|\eta|\|$, $\theta_{0i} = \|\theta_i^{*}\|$.
Completing square in (4.53) it is possible to see that $\|e_{ij}\|$, and $\|\tilde{\theta}_i\|$ are bounded by $\alpha_1 \leq \|e_{ij}\| \leq \alpha_2$, and $\beta_1 \leq \|\tilde{\theta}_i\| \leq \beta_2$. Thus, $\dot{V} > 0$ inside a compact set, but $\dot{V} \leq 0$ outside it. Then $\|e_{ij}\|$ has lower bound. ■
Theorem 1 guarantees that all synchronization errors of quadrotors are bounded.
Finally, this procedure is performed for each axis to fulfill the following control approach requirements. Bearing in mind the special considerations for the $z_W$ axis, such as adding the gravitational $g$ term and avoiding the use of $\delta_i$, since the desired behavior is to achieve consensus in altitude. Moreover, the vector $F_i^{des}$ is as follows

$$F_i^{des} = \begin{bmatrix} F_i^x \\ F_i^y \\ F_i^z \end{bmatrix}. \quad (4.54)$$

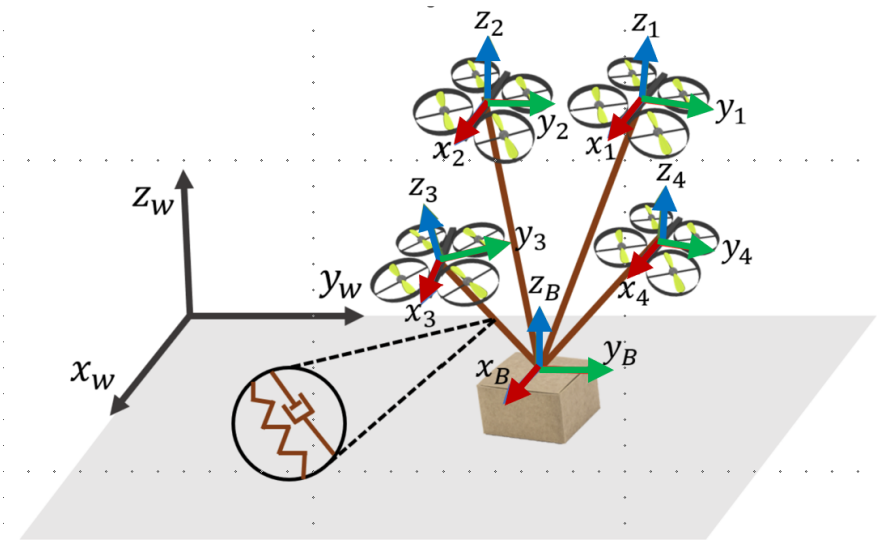



**Figure** 4.6: Multiple quadrotors transporting a cable-suspended load.

## 4.4. Bilevel Adaptive Control

### 4.4.1. Problem Statement

We propose the employment of heterogeneous quadrotors (in the sense that they can have different masses, rotors, and inertial tensors) transporting a cable-suspended load to a desired position. Figure 4.6 shows the configuration we are using where four quadrotors are attached to a point-mass load through stretchable cables.

**Definition 1 (Cable attached quadrotor)** *A* Cable attached quadrotor *is composed of a quadrotor attached from its center of mass with a cable to another object in the environment, the maneuverability of a quadrotor considering the restrictions that a cable impose is considered*

We refer to the position of a quadrotor as $\mathbf{r}_i \in \mathbf{q}$, where $\mathbf{q} = \{1, ..., n\}$ is the set of quadrotors. The mass and inertia tensor of each quadrotor is denoted by $m_i \in \mathbb{R}_{>0}$ and $\mathbf{J}_i \in \mathbb{R}^{3\times3}$, respectively. Figure 4.6 illustrates the inertial reference frame, each quadrotor body frame, and the box frame, which are defined as follows $\{\mathcal{W}\} = \{\mathbf{x}_w, \mathbf{y}_w, \mathbf{z}_w\}$, $\{\mathcal{Q}_i\} = \{\mathbf{x}_{q_i}, \mathbf{y}_{q_i}, \mathbf{z}_{q_i}\}$, and $\{\mathcal{B}\} = \{\mathbf{x}_B, \mathbf{y}_B, \mathbf{z}_B\}$ respectively. Note that each frame can be transformed to the inertial reference frame by using the rotational matrix as $\mathbf{x}_{q_i} = \mathbf{R}_i^W\mathbf{e}_1$, $\mathbf{y}_{q_i} = \mathbf{R}_i^W\mathbf{e}_2$, $\mathbf{z}_{q_i} = \mathbf{R}_i^W\mathbf{e}_3$, $\mathbf{x}_B = \mathbf{R}_B^W\mathbf{e}_1$, $\mathbf{y}_B = \mathbf{R}_B^W\mathbf{e}_2$, and $\mathbf{z}_B = \mathbf{R}_B^W\mathbf{e}_3$, where $\mathbf{e}_1 = [1\,0\,0]$, $\mathbf{e}_2 = [0\,1\,0]$, and $\mathbf{e}_3 = [0\,0\,1]$ are unitarian vectors. We refer to the velocity and acceleration of a quadrotor as $\mathbf{v}_i \in \mathbb{R}^3$ and $\dot{\mathbf{v}}_i \in \mathbb{R}^3$ in $\{\mathcal{W}\}$, $\mathbf{R}_i^w \in SO(3)$ a rotation matrix from each quadrotor frame to the inertial reference frame, $\boldsymbol{\omega}_i \in \mathbb{R}^3$ angular velocity of the $i$-th quadrotor, $f_i \in \mathbb{R}$ total thrust produced by the $i$-th quadrotor, $\boldsymbol{\tau}_i \in \mathbb{R}^3$ moment produced by the $i$-th quadrotor, the length of the $i^{\text{th}}$ cable going from the $i$-th quadrotor to the load is $l_i \in \mathbb{R}_{>0}$.

**Definition 2 (Box)** *A* box *is a cuboid with the following properties, side length*: $a \in \mathbb{R}_{>0}$, *width*: $b \in \mathbb{R}_{>0}$, *height*: $c \in \mathbb{R}_{>0}$, *mass* $m_b \in \mathbb{R}_{>0}$, *and inertia tensor* $\mathbf{J}_b \in \mathbb{R}^{3\times3}$, *the position, velocity and acceleration of the box* $\mathbf{r}_b \in \mathbb{R}^3$, $\mathbf{v}_b \in \mathbb{R}^3$, *and* $\dot{\mathbf{v}}_b \in \mathbb{R}^3$ *in* $\{\mathcal{W}\}$, *angular velocity* $\boldsymbol{\omega}_b$, *respectively,* $\mathbf{f}_{Ti} \in \mathbb{R}_{\geq 0}$ *tension on the cable produced by the box to the* $i-th$ *quadrotor.*

#### Configuration Space

We next consider the next two scenarios. *Quadrotors Moving Free:* Here the cable is not being taut so $\forall i, \quad \|\mathbf{r}_i - \mathbf{r}_b\| < l_i$, then the configuration space is $SE(3)^n$ where $n$ is the number of quadrotors.
*Quadrotors Transporting a Cable-suspended Load:* Once that all of the quadrotors are lifting the load, then $\|\mathbf{r}_i - \mathbf{r}_b\| = l_i + \epsilon_i$ where $\epsilon_i$ is a small cable deformation which depends on the tension applied in the cable. Thus the configuration spaces are led to $SE(3) \times \mathbb{R}^n \times SO(3)^n \times S^{2n}$ where n is the number of quadrotors, for this particular problem $n = 4$.

### 4.4.2. Load Transporting Problem

Here, we propose the employment of four quadrotors to lift a load and carry it to a desired location. Fig. 4.6 illustrates the inertial reference frame $\{\mathcal{W}\}$ and each quadrotor body frame which coincides with the center of mass of each quadrotor.

**Problem 1** *Given a desired trajectory for a load to follow, with desired positions, velocities, and accelerations* $\mathbf{r}_b^d$, $\mathbf{v}_b^d$, *and* $\dot{\mathbf{v}}_b^d$, *respectively, we want to find the control inputs* $f_i, \boldsymbol{\tau}_i$ *for each quadrotor which makes the load to follow the desired trajectory, given physical interactions between the quadrotors due to that they are attached to the same point to the load.*

**Remark 1.** As each quadrotor is attached through a stretchable cable to a point-mass load, the desired position of each quadrotor when the cable has tension can be computed as

$$\mathbf{r}_i^d = \mathbf{r}_b^d + l_i \left( \frac{\mathbf{f}_{T_i}}{\|\mathbf{f}_{T_i}\|} \right).$$

Note that the second term captures the direction of the quadrotor from the attaching point to each quadrotor.

### 4.4.3. Model Dynamics

As mentioned in the configuration space, we consider two scenarios, one with the quadrotors moving free and the second with the quadrotors tightening the cables attached to the load.

### Quadrotor Dynamics Moving Free

By using the Newton-Euler approach to a quadrotor the system model is the following [55]

$$\begin{aligned} \dot{\mathbf{r}}_i &= \mathbf{v}_i \\ m_i\dot{\mathbf{v}}_i &= -m_i g\mathbf{e}_3 + f_i\mathbf{R}_i^W\mathbf{e}_3 \\ \dot{\mathbf{R}}_i^W &= \mathbf{R}_i^W\hat{\boldsymbol{\omega}}_i, \\ \mathbf{J}_i\dot{\boldsymbol{\omega}}_i &= -\boldsymbol{\omega}_i \times \mathbf{J}_i\boldsymbol{\omega}_i + \boldsymbol{\tau}_i, \end{aligned} \tag{4.55}$$

where $g$ is the gravitational constant acceleration applied in $\mathbf{e}_3$ and the hat map $\hat{\cdot} : \mathbb{R}^3 \to SO(3)$ defined by the condition that $\hat{\mathbf{a}}\mathbf{b} = \mathbf{a} \times \mathbf{b}$, $\forall\, \mathbf{a}, \mathbf{b} \in \mathbb{R}^3$ [60].

### Quadrotors Transporting the Load

We follow the same procedure to model the dynamics of the quadrotors attached to a cable-suspended load using the Newton-Euler approach

$$\begin{aligned} \dot{\mathbf{r}}_i &= \mathbf{v}_i \\ m_i\dot{\mathbf{v}}_i &= -m_i g\mathbf{e}_3 + f_i\mathbf{R}_i^W\mathbf{e}_3 + \mathbf{R}_b^W\mathbf{f}_{Ti}, \\ \dot{\mathbf{R}}_i^W &= \mathbf{R}_i^W\hat{\boldsymbol{\omega}}_i, \\ \mathbf{J}_i\dot{\boldsymbol{\omega}}_i &= -\boldsymbol{\omega}_i \times \mathbf{J}_i\boldsymbol{\omega}_i + \boldsymbol{\tau}_i, \\ \dot{\mathbf{r}}_b &= \mathbf{v}_b \\ m_b\ddot{\mathbf{r}}_b &= -m_b g\mathbf{e}_3 + \sum_i^N (\mathbf{R}_b^W\mathbf{f}_{Ti}). \end{aligned} \tag{4.56}$$

Note that the difference here lies in adding the tension force $\mathbf{f}_{Ti}$ and the loaded dynamic. Considering that we consider the cable as a spring-damping system [51], the variation in the position of one quadrotor generates a tension force on the other cables. In the same way, one quadrotor experiences an opposing force coming from the other cables when it moves. Let's define $k_i$ and $v_i$ as the constant factors coming from the spring and damping effects when referring to the translational movement in (4.56). Considering $\dot{\mathbf{x}} = [\dot{\mathbf{r}}_i^x, \dot{\mathbf{v}}_i^x]^\top$, $\dot{\mathbf{y}} = [\dot{\mathbf{r}}_i^y, \dot{\mathbf{v}}_i^y]^\top$, and $\dot{\mathbf{z}} = [\dot{\mathbf{r}}_i^z, \dot{\mathbf{v}}_i^z]^\top$ as the dynamic of the system in each axis of $\{\mathcal{W}\}$ can be transformed as

$$\dot{\mathbf{x}}_i = \begin{bmatrix} 0 & 1 \\ -\frac{k_i}{m_i} & -\frac{b_i}{m_i} \end{bmatrix} \mathbf{x}_i + \begin{bmatrix} 0 \\ f_{x_i}^d \end{bmatrix} + \sum_{j \in \mathcal{N}_i} \begin{bmatrix} 0 & 1 \\ -\frac{k_j}{m_j} & -\frac{b_j}{m_j} \end{bmatrix} \mathbf{x}_j + \eta_{ij}, \tag{4.57}$$

where $\mathcal{N}_i$ refers to the neighborhood of the $i$-th quadrotor, $k_j$, $b_j$, $m_j$ are the parameters of the other quadrotors, and $\eta_{ij}$ is an unknown parameter collecting the uncertainties and perturbations. Due to the limited space, we show the procedure to the $\mathbf{x}_w$-axis only, even though the procedure to the $\mathbf{y}_w$-axis and $\mathbf{z}_w$-axis is the same just remembering that for the

$\mathbf{z}_i$-axis we add the gravity effects. Note that $f_{x_i}^d$ is considered in each of the three axes, but the thrust of a quadrotor can just be generated in the $\mathbf{z}_{q_i}$- axis. Transforming (4.57) in a compact form, considering $u_i = [0, f_{x_i}^d]^\top$, then

$$\dot{\mathbf{x}}_i = \mathbf{A}_i\mathbf{x}_i + \mathbf{b}_i u_i + \sum_{j\in\mathcal{N}_i} \mathbf{A}_{ij}\mathbf{x}_j + \eta_{ij}, \tag{4.58}$$

this is the dynamic model we work with, and in the next section, where the control design is explained.

### 4.4.4. Control Design

We split this problem into a translational and rotational control design for the control theory analysis. First, the translational control depends on the communication that a drone has with the predefined virtual reference model and the physical interaction. Second, the rotational control takes the desired thrust controller and translates it to the rotation needed.

#### Desired Force on the $\mathbf{x}_w$-axis Design

The following proposition shows the case where a drone directly communicates with the reference. A linear reference model is defined as

$$\dot{\mathbf{x}}_m = \mathbf{A}_m\mathbf{x}_m + \mathbf{b}_m r(t)^x, \tag{4.59}$$

where $\mathbf{x}_m \in \mathbb{R}^n$ is the reference state, and $r(t)$ is the reference trajectory that we want the mass to follow. The matrix and vector $\mathbf{A}_m \in \mathbb{R}^{n\times n}$, $\mathbf{b}_m \in \mathbb{R}^n \times p$ are known and consistent with linear dynamics.

**Proposition 1.** Consider a drone $\mathbf{x}_1$ with dynamics (4.58) communicated directly with the reference (4.59). Using the following control law

$$u_1 = \mathbf{k}_m\mathbf{x}_m + \mathbf{k}_r r(t)^x - \boldsymbol{\theta}^\top\boldsymbol{\phi}, \tag{4.60}$$

where $\mathbf{k}_m \in \mathbb{R}^n$ and $\mathbf{k}_r \in \mathbb{R}^n$ are the result of the adaptive laws

$$\dot{\mathbf{k}}_m^\top = -\text{sgn}\left(\mathbf{k}_r{}^*\right)\gamma\,\mathbf{b}_m^\top\mathbf{P}\left(\mathbf{x}_1 - \mathbf{x}_m\right)\mathbf{x}_1^\top, \tag{4.61}$$

$$\dot{\mathbf{k}}_r = -\text{sgn}\left(\mathbf{k}_r{}^*\right)\gamma\,\mathbf{b}_m^\top\mathbf{P}\left(\mathbf{x}_1 - \mathbf{x}_m\right)r(t)^x, \tag{4.62}$$

with the adaptive gain $\gamma > 0$ and $\mathbf{P} \in \mathbb{R}^{n\times n}$ the solution of the linear Lyapunov equation

$$\mathbf{P}\mathbf{A}_m + \mathbf{A}_m^\top\mathbf{P} = -\mathbf{Q}, \quad \mathbf{Q} \succ 0. \tag{4.63}$$

The auxiliary control parameter $\boldsymbol{\theta}^\top \boldsymbol{\phi}$ allows suppression of uncertainty parameters with $\boldsymbol{\theta} \in \mathbb{R}^{n\times p}$ defined as

$$\dot{\boldsymbol{\theta}} = -\gamma \left( \boldsymbol{\phi} \left( \mathbf{x}_1 - \mathbf{x}_m \right)^\top \mathbf{P}\mathbf{b}_1 + \sigma \boldsymbol{\theta} \right), \tag{4.64}$$

where $\boldsymbol{\phi}\colon \mathbb{R}^n \rightarrow \mathbb{R}^p$ is a known bounded basis function and $\sigma$ is an adaptive gain. By synthesizing the information through the controller and the adaptive laws, it is possible to achieve synchronization with the reference model.

*Proof:* It follows by [66].

On the other hand, the proposed control methodology is defined in the following proposition for drones that are not directly communicated with the reference but have a physical interconnection.

**Proposition 2:** Consider a second agent $\mathbf{x}_2$ with dynamics (4.58) interconnected and communicated with the drone $\mathbf{x}_1$, using the following control law

$$u_2 = \mathbf{k}_{m21}^\top \mathbf{x}_1 + \mathbf{k}_{m2}^\top \mathbf{e}_{21} + \mathbf{k}_{r21} u_1 + \mathbf{k}_{21}\mathbf{x}_1 - \boldsymbol{\theta}_2^\top \boldsymbol{\phi}_2, \tag{4.65}$$

with $\mathbf{e}_{21} = (\mathbf{x}_2 - \mathbf{x}_1)$ and the adaptive laws

$$\dot{\mathbf{k}}_{m21}^\top = -\operatorname{sgn}\left(\mathbf{k}_{r2}^*\right) \gamma\, \mathbf{b}_m^\top \mathbf{P}\mathbf{e}_{21}\mathbf{x}_2^\top, \tag{4.66}$$

$$\dot{\mathbf{k}}_{m2}^\top = -\operatorname{sgn}\left(\mathbf{k}_{r2}^*\right) \gamma\, \mathbf{b}_m^\top \mathbf{P}\mathbf{e}_{21}\mathbf{e}_{21}^\top, \tag{4.67}$$

$$\dot{\mathbf{k}}_{r21} = -\operatorname{sgn}\left(\mathbf{k}_{r2}^*\right) \gamma\, \mathbf{b}_m^\top \mathbf{P}\mathbf{e}_{21} u_1, \tag{4.68}$$

$$\dot{\mathbf{k}}_{21}^\top = -\operatorname{sgn}(\mathbf{k}_{r2}^*) \gamma\, \mathbf{b}_m^\top \mathbf{P}\mathbf{e}_{21}\mathbf{x}_1^\top, \tag{4.69}$$

$$\dot{\boldsymbol{\theta}}_2 = -\gamma \left( \boldsymbol{\phi}\mathbf{e}_{21}^\top \mathbf{P}\mathbf{b}_1 + \sigma\boldsymbol{\theta} \right). \tag{4.70}$$

Then, the drone is synchronized with the reference model.

*Proof:* To validate the synchronization of a drone not communicated to the reference model, an analysis is done using Lyapunov's theory. The following Lyapunov function is defined

$$\mathcal{V}_{21} = \mathbf{e}_{21}^\top \mathbf{P}\mathbf{e}_{21} + \operatorname{tr}\left( \frac{\tilde{\mathbf{k}}_{m21}^\top \tilde{\mathbf{k}}_{m21}}{\gamma |\mathbf{k}_{r2}^*|} \right) + \operatorname{tr}\left( \frac{\tilde{\mathbf{k}}_{m2}^\top \tilde{\mathbf{k}}_{m2}}{\gamma |\mathbf{k}_{r2}^*|} \right) \tag{4.71}$$

$$+ \operatorname{tr}\left( \frac{\tilde{\mathbf{k}}_{21}^\top \tilde{\mathbf{k}}_{21}}{\gamma |\mathbf{k}_{r2}^*|} \right) + \frac{\tilde{\mathbf{k}}_{r21}^2}{\gamma |\mathbf{k}_{r2}^*|} + \operatorname{tr}\left( \tilde{\boldsymbol{\theta}}_2^\top \gamma^{-1} \tilde{\boldsymbol{\theta}}_2 \right), \tag{4.72}$$

where $\tilde{\mathbf{k}}_{m21} = \mathbf{k}_{m21} - \mathbf{k}_{m21}^*$; $\tilde{\mathbf{k}}_{m2} = \mathbf{k}_{m2} - \mathbf{k}_{m2}^*$; $\tilde{\mathbf{k}}_{21} = \mathbf{k}_{21} - \mathbf{k}_{21}^*$; $\tilde{\mathbf{k}}_{r21} = \mathbf{k}_{r21} - \mathbf{k}_{r21}^*$; $\tilde{\boldsymbol{\theta}}_2 = \boldsymbol{\theta}_2 - \boldsymbol{\theta}_2^*$.

Extending the dynamics of the error $\mathbf{e}_{21}$, the equation (4.72) is derived obtaining

$$\dot{\mathcal{V}}_{21} = \mathbf{e}_{21}^{\top}\left(\mathbf{P}\mathbf{A}_m + \mathbf{A}_m^{\top}\mathbf{P}\right)\mathbf{e}_{21} \tag{4.73}$$

$$+ 2\mathbf{e}_{21}^{\top}\mathbf{P}\mathbf{b}_2\left(\tilde{\mathbf{k}}_{m21}^{\top}\mathbf{x}_1 + \tilde{\mathbf{k}}_{m2}^{\top}\mathbf{e}_{21} + \tilde{\mathbf{k}}_{r21}^{\top}u_1 + \tilde{\mathbf{k}}_{21}^{\top}\mathbf{x}_1 - \tilde{\boldsymbol{\theta}}_2^{\top}\boldsymbol{\phi}_2\right) \tag{4.74}$$

$$+ 2\mathrm{tr}\left(\frac{\tilde{\mathbf{k}}_{21}^{\top}\gamma^{-1}\dot{\tilde{\mathbf{k}}}_{21}}{|\mathbf{k}_{r2}^{*}|}\right) + 2\mathrm{tr}\left(\frac{\tilde{\mathbf{k}}_{m2}^{\top}\gamma^{-1}\dot{\tilde{\mathbf{k}}}_{m2}}{|\mathbf{k}_{r2}^{*}|}\right) \tag{4.75}$$

$$+ 2\frac{\tilde{\mathbf{k}}_{r21}^{\top}\gamma^{-1}\dot{\tilde{\mathbf{k}}}_{r21}}{|\mathbf{k}_{r2}^{*}|} - 2\mathrm{tr}\left(\tilde{\boldsymbol{\theta}}_2^{\top}\boldsymbol{\phi}_2[\mathbf{e}_{21}^{\top}\mathbf{P} - v\boldsymbol{\phi}_2^{\top}\boldsymbol{\theta}_2\mathbf{b}_2^{\top}\mathbf{P}\mathbf{A}_m^{-1}]\mathbf{b}_2\right), \tag{4.76}$$

and with mathematical reduction

$$\dot{\mathcal{V}}_{21} = -\mathbf{e}_{21}^{\top}\mathbf{Q}\mathbf{e}_{21} + 2\left(\mathrm{sgn}\left(\mathbf{k}_{r2}^{*}\right)\mathbf{b}_m^{\top}\mathbf{P}\mathbf{e}_{21}\mathbf{x}_1^{\top} + \gamma^{-1}\tilde{\mathbf{k}}_{m21}^{\top}\right)\frac{\tilde{\mathbf{k}}_{m21}^{\top}}{|\mathbf{k}_{r2}^{*}|} \tag{4.77}$$

$$+ 2\left(\mathrm{sgn}\left(\mathbf{k}_{r2}^{*}\right)\mathbf{b}_m^{\top}\mathbf{P}\mathbf{e}_{21}\mathbf{e}_{21}^{\top} + \gamma^{-1}\tilde{\mathbf{k}}_{m2}^{\top}\right)\frac{\tilde{\mathbf{k}}_{m2}^{\top}}{|\mathbf{k}_{r2}^{*}|} \tag{4.78}$$

$$+ 2\left(\mathrm{sgn}\left(\mathbf{k}_{r2}^{*}\right)\mathbf{b}_m^{\top}\mathbf{P}\mathbf{e}_{21}u_1 + \gamma^{-1}\tilde{\mathbf{k}}_{r21}^{\top}\right)\frac{\tilde{\mathbf{k}}_{r21}^{\top}}{|\mathbf{k}_{r2}^{*}|} \tag{4.79}$$

$$+ 2\left(\mathrm{sgn}\left(\mathbf{k}_{r2}^{*}\right)\mathbf{b}_m^{\top}\mathbf{P}\mathbf{e}_{21}u_1 + \gamma^{-1}\tilde{\mathbf{k}}_{21}^{\top}\right)\frac{\tilde{\mathbf{k}}_{21}^{\top}}{|\mathbf{k}_{r2}^{*}|} \tag{4.80}$$

$$+ 2v\boldsymbol{\phi}_2^{\top}\tilde{\boldsymbol{\theta}}_2\mathbf{b}_2^{\top}\mathbf{P}\mathbf{A}_m^{-1}\mathbf{b}_2\tilde{\boldsymbol{\theta}}_2{}^{\top}\boldsymbol{\phi}_2 \tag{4.81}$$

$$+ 2v\boldsymbol{\phi}_2^{\top}\left(\boldsymbol{\theta}_2^{\top}\boldsymbol{\phi}_2 + \epsilon_2\right)\mathbf{b}_2^{\top}\mathbf{P}\mathbf{A}_m^{-1}\mathbf{b}_2\tilde{\boldsymbol{\theta}}_2{}^{\top}\boldsymbol{\phi}_2. \tag{4.82}$$

Considering $\mathbf{b}_2^{\top}\mathbf{P}\mathbf{A}_m^{-1}\mathbf{b}_2 < 0$ and defining the boundary parameters $\beta_1 = \lambda_{\text{mín}}(\mathbf{Q})$, $\beta_2 = \lambda_{\text{mín}}(\mathbf{b}_2\mathbf{A}_m^{-\top}\mathbf{Q}\mathbf{A}_m^{-1}\mathbf{b}_2)$ and $\beta_3 = \frac{\|\mathbf{b}_2^{\top}\mathbf{P}\mathbf{A}_m^{-1}\mathbf{b}_2\|\boldsymbol{\theta}_{02}}{\beta_2}$, with $\boldsymbol{\theta}_{02} = \|\boldsymbol{\theta}_2^{*}\|$ an defining the bound

$$\|\mathbf{e}_{21}\| \geq \sqrt{\frac{v\beta_2\beta_3\|\boldsymbol{\phi}_2\|^2}{\beta_1}} = \psi_2. \tag{4.83}$$

Therefore, equation (4.72) is bounded by

$$\dot{\mathcal{V}}_{21} \leq -\lambda_{\text{mín}}\left(\mathbf{Q}\right)\|\mathbf{e}_{21}\|^2 \tag{4.84}$$

$$+ 2\lambda_{\text{máx}}(\mathbf{P})\|\mathbf{b}_2\|\left(\|\boldsymbol{\theta}_2^{*\top}\boldsymbol{\phi}_2\| + \|\delta_{\epsilon 2}\|\right)\|\mathbf{e}_{21}\| \tag{4.85}$$

$$- \mathbf{v}\lambda_{min}(\mathbf{Q})\|\mathbf{A}_m^{-1}\mathbf{b}_2\boldsymbol{\theta}_2^{\top}\boldsymbol{\phi}_2\|^2, \tag{4.86}$$

with $\|\boldsymbol{\theta}_2^{*\top}\boldsymbol{\phi}_2\| = \|\sup_t|\boldsymbol{\theta}_2^{*\top}\boldsymbol{\phi}_2|\|$, synchronization error $\|\mathbf{e}_{21}\|$ has as lower bound $\psi_2$, what leads to $\dot{\mathcal{V}}_{21} \leq 0$, this guarantees the boundary of the synchronization error. ■

To generalize the solution to this control problem, the following distributed control law is defined as

$$\bar{a}u_i = \bar{a}\mathbf{k}_{mij}^{\top}\mathbf{x}_i + \mathbf{k}_{mi}\boldsymbol{\Xi}_{ij} + \bar{a}\mathbf{k}_{rij}u_i + \bar{a}\mathbf{k}_{ij}\mathbf{x}_j - \boldsymbol{\theta}_i^{\top}\boldsymbol{\phi}_i, \tag{4.87}$$

where $\mathbf{\Xi}_{ij} = \sum_{j\in\mathcal{N}_i} a_{ij}(\mathbf{x}_i - \mathbf{x}_j)$ and $\bar{a} = \sum_{j\in\mathcal{N}_i} a_{ij}$. The following adaptive laws are defined as well

$$\dot{\mathbf{k}}_{mij}^{\top} = -\operatorname{sgn}(\mathbf{k}_{ri}^{*})\gamma\, \mathbf{b}_m^{\top}\mathbf{P}\mathbf{\Xi}_{ij}\mathbf{x}_i^{\top} \tag{4.88}$$

$$\dot{\mathbf{k}}_{mi}^{\top} = -\operatorname{sgn}(\mathbf{k}_{ri}^{*})\gamma\, \mathbf{b}_m^{\top}\mathbf{P}\mathbf{\Xi}_{ij}\mathbf{\Xi}_{ij}^{\top} \tag{4.89}$$

$$\dot{\mathbf{k}}_{rij} = -\operatorname{sgn}(\mathbf{k}_{ri}^{*})\gamma\, \mathbf{b}_m^{\top}\mathbf{P}\mathbf{\Xi}_{ij}u_i \tag{4.90}$$

$$\dot{\mathbf{k}}_{ij}^{\top} = -\operatorname{sgn}(\mathbf{k}_{ri}^{*})\gamma\, \mathbf{b}_m^{\top}\mathbf{P}\mathbf{\Xi}_{ij}\mathbf{x}_j^{\top} \tag{4.91}$$

$$\dot{\boldsymbol{\theta}}_i = -\gamma(\boldsymbol{\phi}_i\mathbf{e}_{ij})\mathbf{P}\mathbf{b}_i + \sigma\boldsymbol{\theta}_i \tag{4.92}$$

where $\gamma$, $v$ and $\sigma$ are positive adaptive gains.

**Theorem 1.** Consider a network of drones with dynamics (4.55), interconnected to each other and unlinked from the reference model (4.59), then a synchronization is achieved using the control law (4.87) and the adaptive laws (4.92).

*Proof:* To validate the boundary of the drone distributed synchronization error, we use Lyapunov's theory, where we define the following equation

$$\mathcal{V} = \sum_{i\in\mathcal{N}_i} \mathbf{\Xi}_{ij}^{\top}\mathbf{P}\mathbf{\Xi}_{ij} + \sum_{j\in\mathcal{N}_i} \operatorname{tr}\left[\frac{\tilde{\mathbf{k}}_{mi}^{\top}\tilde{\mathbf{k}}_{mi}}{\gamma\ |\mathbf{k}_{ri}^{*}|}\right] + \sum_{i\in\mathcal{N}_i} \bar{a}\ \operatorname{tr}\left[\frac{\tilde{\mathbf{k}}_{mij}^{\top}\tilde{\mathbf{k}}_{mij}}{\gamma\ |\mathbf{k}_{ri}^{*}|}\right] \tag{4.93}$$

$$+ \sum_{i\in\mathcal{N}_i} \bar{a}\ \operatorname{tr}\left[\frac{\tilde{\mathbf{k}}_{ij}^{\top}\tilde{\mathbf{k}}_{ij}}{\gamma\ |\mathbf{k}_{ri}^{*}|}\right] + \sum_{i\in\mathcal{N}_i} \bar{a}\frac{\tilde{\mathbf{k}}_{ri}^{2}}{\gamma\ |\mathbf{k}_{r}^{*}|} + \sum_{i\in\mathcal{N}_i} \operatorname{tr}(\tilde{\boldsymbol{\theta}}_i^{\top}\gamma^{-1}\tilde{\boldsymbol{\theta}}_i), \tag{4.94}$$

derived along the dynamics of the synchronization error

$$\dot{\mathbf{e}}_{ij} = \mathbf{A}_m\mathbf{e}_{ij} + \mathbf{b}_i[\tilde{\mathbf{k}}_{ij}^{\top}\mathbf{x}_i + \tilde{\mathbf{k}}_{mij}^{\top}\mathbf{e}_{ij} + \tilde{\mathbf{k}}_{rij}^{\top}u_i + \tilde{\mathbf{k}}_{ij}^{\top}\mathbf{x}_j - \tilde{\boldsymbol{\theta}}_i^{\top}\boldsymbol{\phi}_i - \tilde{\boldsymbol{\theta}}_j^{\top}\boldsymbol{\phi}_j]. \tag{4.95}$$

obtaining

$$\dot{\mathcal{V}} = \sum_{i\in\mathcal{N}_i} \mathbf{\Xi}_{ij}^{\top}(\mathbf{P}\mathbf{A}_m + \mathbf{A}_m^{\top}\mathbf{P})\mathbf{\Xi}_{ij} \tag{4.96}$$

$$+ 2\mathbf{\Xi}_{ij}^{\top}\mathbf{P}\mathbf{b}_i\left[\bar{a}\tilde{\mathbf{k}}_{ij}^{\top}\mathbf{x}_i + \tilde{\mathbf{k}}_{mi}^{\top}\bar{a}\mathbf{e}_{ij} + \bar{a}\tilde{\mathbf{k}}_{rij}u_i - \boldsymbol{\theta}_j^{\top}\boldsymbol{\phi}_j\right] \tag{4.97}$$

$$+ \sum_{i\in\mathcal{N}_i} \operatorname{tr}\left[\frac{\tilde{\mathbf{k}}_{mi}^{\top}\gamma^{-1}\dot{\tilde{\mathbf{k}}}_{mi}}{|\mathbf{k}_{ri}^{*}|}\right] + \sum_{i\in\mathcal{N}_i} \operatorname{tr}\left[\frac{\tilde{\mathbf{k}}_{mij}^{\top}\gamma^{-1}\dot{\tilde{\mathbf{k}}}_{mij}}{|\mathbf{k}_{ri}^{*}|}\right] \tag{4.98}$$

$$+ \sum_{i\in\mathcal{N}_i} \operatorname{tr}\left[\frac{\tilde{\mathbf{k}}_{ij}^{\top}\gamma^{-1}\dot{\tilde{\mathbf{k}}}_{ij}}{|\mathbf{k}_{ri}^{*}|}\right] + \sum_{i\in\mathcal{N}_i} \bar{a}\frac{\tilde{\mathbf{k}}_{rij}\gamma^{-1}\dot{\tilde{\mathbf{k}}}_{rij}}{|\mathbf{k}_{ri}^{*}|} \tag{4.99}$$

$$- 2\sum_{i\in\mathcal{N}_i}\sum_{j\in\mathcal{N}_i} \operatorname{tr}\left(\tilde{\boldsymbol{\theta}}_i^{\top}\boldsymbol{\phi}_i[\mathbf{e}_{ij}^{\top}\mathbf{P} - v\boldsymbol{\phi}_i^{\top}\boldsymbol{\theta}_i\mathbf{b}_i^{\top}\mathbf{P}\mathbf{A}_m^{-1}]\mathbf{b}_i\right), \tag{4.100}$$

where making mathematical reduction

$$\dot{\mathcal{V}} = -\sum_{i\mathcal{N}_i} \mathbf{\Xi}_{ij}^{\top}\mathbf{Q}\mathbf{\Xi}_{ij} + \sum_{i\in\mathcal{N}_i} 2v\boldsymbol{\phi}_i^{\top}\tilde{\theta}_i\mathbf{b}_i^{\top}\mathbf{P}\mathbf{A}_m^{-1}\mathbf{b}_i\tilde{\boldsymbol{\theta}}_i{}^{\top}\boldsymbol{\phi}_i \tag{4.101}$$

$$+\sum_{i\in\mathcal{N}_i} 2v\boldsymbol{\phi}_i^{\top}\left(\boldsymbol{\theta}_i^{\top}\boldsymbol{\phi}_i + \epsilon_i\right)\mathbf{b}_i^{\top}\mathbf{P}\mathbf{A}_m^{-1}\mathbf{b}_i\tilde{\boldsymbol{\theta}}_i{}^{\top}\boldsymbol{\phi}_i. \tag{4.102}$$

Similarly, $\mathbf{b}_i^{\top}\mathbf{P}\mathbf{A}_m^{-1}\mathbf{b}_i < 0$ thus

$$2v\boldsymbol{\phi}_i^{\top}\tilde{\boldsymbol{\theta}}_i\mathbf{b}_i^{\top}\mathbf{P}A_m^{-1}\mathbf{b}_i\tilde{\boldsymbol{\theta}}_i{}^{\top}\boldsymbol{\phi}_i = -v\boldsymbol{\phi}_i^{\top}\tilde{\boldsymbol{\theta}}_i\mathbf{b}_i^{\top}\mathbf{A}_m^{-\top}\mathbf{Q}\mathbf{A}_m^{-1}\mathbf{b}_i\tilde{\boldsymbol{\theta}}_i{}^{\top}\boldsymbol{\phi}_i. \tag{4.103}$$

With the definition of the parameters, $\beta_2 = \lambda_{\text{mín}}(\mathbf{b}_i\mathbf{A}_m^{-\top}\mathbf{Q}\mathbf{A}_m^{-1}\mathbf{b}_i)$, $\beta_3 = \frac{\|\mathbf{b}_i^{\top}\mathbf{P}\mathbf{A}_m^{-1}\mathbf{b}_i\|\boldsymbol{\theta}_{0i}}{\beta_2}$, $\boldsymbol{\theta}_{0i} = \|\boldsymbol{\theta}_i^*\|$ and $\delta_{\epsilon i} = \sup|\epsilon_i|$, allowing to define the bound

$$\sum_{i\in\mathcal{N}_i}\sum_{j\in\mathcal{N}_i}\|\mathbf{e}_{ij}\| \geq \sqrt{\frac{v\beta_2\beta_3\|\boldsymbol{\phi}_i\|^2}{\beta_1}} = \psi_i, \tag{4.104}$$

this allows defining the bound of (4.100) as

$$\dot{\mathcal{V}} \leq -\sum_{i\in\mathcal{N}_i}\lambda_{\text{mín}}\left(\mathbf{Q}\right)\sum_{j\in\mathcal{N}_i}\|\mathbf{e}_{ij}\|^2 \tag{4.105}$$

$$+2\sum_{i\in\mathcal{N}_i}\sum_{j\in\mathcal{N}_i}\lambda_{\text{máx}}(\mathbf{P})\|\mathbf{b}_i\|\left(\|\boldsymbol{\theta}_i^{*\top}\boldsymbol{\phi}_i\| + \|\delta_{\epsilon i}\|\right)\|\mathbf{e}_{ij}\| \tag{4.106}$$

$$-\sum_{i\in\mathcal{N}_i} v\lambda_{min}(\mathbf{Q})\|\mathbf{A}_m^{-1}\mathbf{b}_i\boldsymbol{\theta}_i^{\top}\boldsymbol{\phi}_i\|^2, \tag{4.107}$$

with $\|\boldsymbol{\theta}_i^{*\top}\boldsymbol{\phi}_i\| = \|\sup_t|\boldsymbol{\theta}_i^{*\top}\boldsymbol{\phi}_i|\|$, therefore as well $\|\mathbf{e}_{ij}\|$ has as lower bound $\psi_i$, so $\dot{\mathcal{V}} \leq 0$, then we can guarantee that all the synchronization errors $\mathbf{e}_{ij}$ are bounded. ■

#### Real Thrust Generated on Each Quadrotor $f_i$

Notice that $u_i$ contains the desired force on the $\mathbf{x}_w$-axis $f_{x_i}^d$, in the same way, we can obtain the desired force for the $\mathbf{y}_w$-axis and $\mathbf{z}_w$-axis as $f_{y_i}^d$ and $f_{z_i}^d$ respectively. Lets define the desired force vector for each quadrotor as $\mathbf{f}_i^d = [f_{x_i}^d, f_{y_i}^d, f_{z_i}^d]^{\top} \neq 0$. However, the thrust that each quadrotor can produce is just in the $\mathbf{z}_{q_i}$-axis. Thus the force generated must be $f_i = \mathbf{f}_i^d \cdot \mathbf{z}_{q_i}$.

#### Rotational Control Design $\boldsymbol{\tau}_i$

The input responsible for the attitude control is $\boldsymbol{\tau}_i \in \mathbb{R}^3$. To make the attitude error tend to zero, the translational movement can be expressed as $\mathbf{z}_{q_i}^d = \mathbf{f}_i^d/\left\|\mathbf{f}_i^d\right\|$, taking $\mathbf{f}_i^d$ from the DMRAC. Hence $\mathbf{x}_{q_i}^d = [1\,0\,0]^{\top}$, obtaining a desired attitude as, $\mathbf{R}_i^{w,d} = \left[\mathbf{y}_{q_i}^d \times \mathbf{z}_{q_i}^d\ \mathbf{y}_{q_i}^d\ \mathbf{z}_{q_i}^d\right] \in SO(3)$, where

$$\mathbf{y}_{q_i}^d = \frac{\mathbf{z}_{q_i}^d \times \mathbf{x}_{q_i}^d}{\left\|\mathbf{z}_{q_i}^d \times \mathbf{x}_{q_i}^d\right\|},$$

and finally, defining the attitude error as

$$\mathbf{e}_{R_i} = \frac{1}{2}\left((\mathbf{R}_i^{w,d})^\top \mathbf{R}_i^w - \mathbf{R}_i^{w\top}\mathbf{R}_i^{w,d}\right)^\vee, \tag{4.108}$$

where $\vee$ represents the $vee-operator$ which is the inverse of the skew-symmetric operator as in [55]. Angular velocity can be defined as, $\boldsymbol{\omega}_i^d = \rho_i^d \mathbf{x}_{q_i}^d + \varrho_i^d \mathbf{y}_{q_i}^d$, where $\rho_i^d = -\mathbf{h}_{\omega_i} \cdot \mathbf{y}_{q_i}^d$ and $\varrho_i^d = \mathbf{h}_{\omega_i} \cdot \mathbf{x}_{q_i}^d$ where

$$\mathbf{h}_{\omega_i} = \frac{m_v}{f_i}(\dot{\mathbf{v}}_i^d - (\mathbf{z}_{q_i}^d \cdot \dot{\mathbf{v}}_i^d)\mathbf{z}_{q_i}^d),$$

thus the angular velocity error is

$$\mathbf{e}_{\boldsymbol{\omega}_i} = \boldsymbol{\omega}_i - \boldsymbol{\omega}_i^d, \tag{4.109}$$

thus $\boldsymbol{\tau}_i$ control input taking (4.109) and (4.108) is as follows

$$\begin{aligned}\boldsymbol{\tau}_i = &- \mathbf{K}_R \mathbf{e}_{R_i} - \mathbf{K}_\omega \mathbf{e}_{\omega_i} + (\boldsymbol{\omega}_i \times \mathbf{J}_i \boldsymbol{\omega}_i) \\ &- \mathbf{J}_i(\hat{\boldsymbol{\omega}}_i \mathbf{R}_i^{w\top}\mathbf{R}_i^{w,d}\boldsymbol{\omega}_i^d - \mathbf{R}_i^{w\top}\mathbf{R}^{w,d}\dot{\boldsymbol{\omega}}),\end{aligned}$$

where $\mathbf{K}_R \in \mathbb{R}^{3\times3}$ and $\mathbf{K}_\omega \in \mathbb{R}^{3\times3}$ are positive diagonal gain matrices that guarantee the stability of the system.

# 5 Simulation and Results

## 5.1. Exploration and Navigation

### 5.1.1. Reinforcement Learning

The results shown here are part of the following work [11], in which the proposed reinforcement learning system is evaluated through a virtual robot simulator V-Rep in the Edu version. This experiment is used to verify that the agent learns to navigate without colliding. Besides, the performance of some of the important parameters of the proposed system will be reviewed. The experiment is based on the start-up of a terrestrial robot with differential traction in an unknown environment. The robot used in this experiment is the virtual model of a Pioneer $3-DX$, and the unknown virtual environment is an office, considering that in a disaster zone, robots will have to navigate through non-convex indoor spaces. The robot simulation system used in the experiment is V-Rep (Virtual Robot Experimentation Platform). This system allows the use of virtual robot models and the building of various scenarios. The scenario and robot used in this experiment are shown in Figure 5.1.

#### Robot

The robot used in this experiment is a Pioneer $3-DX$. This robot has differential traction and three distance sensors attached. These sensors were located on the front and the other two on the sides. The robot is shown in Figure. 5.2. The robot is controlled by setting up the speed to the left and right wheels, and its kinematics corresponds to the model shown in Section 2.2.1.

#### Scenario

The scenario is an office with a size of 10x10 meters. The scenario has sofas, meeting tables, computers, shelves, sliding doors, walls, plants, and people. All the objects in the scenario keep a real proportion between them and the robot and have been put as collidable and detectable to the robot.

#### Experiment setup

Some important parameters are set to carry out the simulation, which is mentioned below. The maximum robot speed maxVel is $0{,}5m/s$. The robot has 3 distance sensors (one on

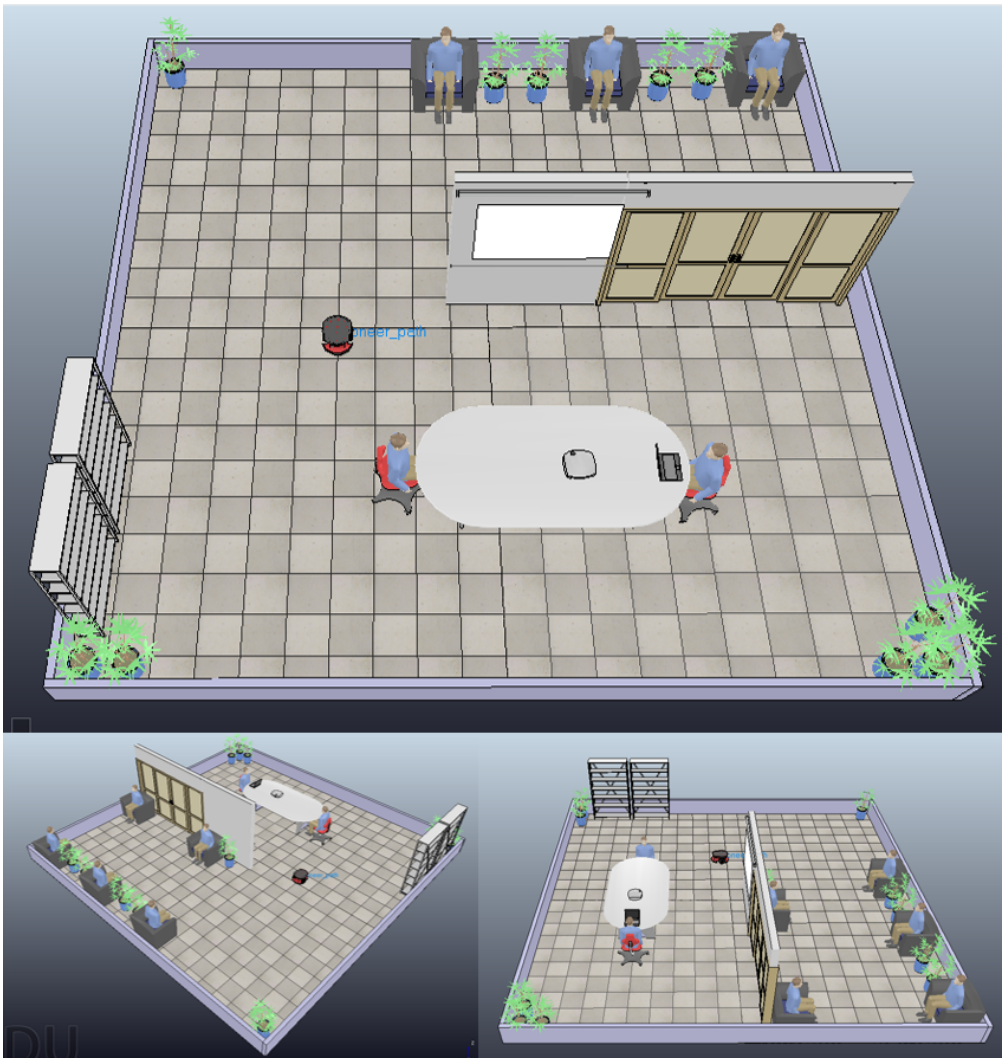


**Figure** 5.1: The unknown simulation environment for the robot.

the front and two on the sides). The discretization of the sensors is of 5 levels, where the maximum level is $L4$ for distances greater than 4 meters, and the minimum is $L0$ for distances less than 1 meter. Since the robot has 3 sensors and the discretization of the sensors is 5 levels, according to (2.6), the number of states of the system is 125. The movements of the robot were limited to three. Where the robot can move forward, turn left, or turn right. Given the number of actions, the levels of discretization of the sensors, and the number of sensors, the dimension of Q is a 125x3 matrix. The coefficients of the reward policy were defined according to the position of the sensor, $a_{1,3} = 50, a_2 = 100, b_{1,3} = 5, b_2 = 20$, where sensors 1 and 3 are located at robot sides and sensor 2 is on the front. The Learning rate factors were defined as: $\alpha_\infty = 0{,}05, \alpha_s = 1$, and $\alpha_{50} = 8$. Discount factor $(\gamma) = 0{,}5$, and finally, the initial value for the exploration policy $k_a = 800$.

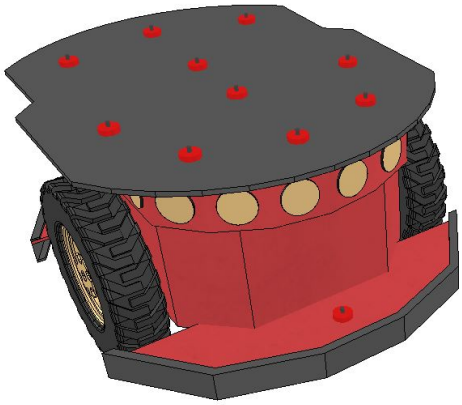


**Figure** 5.2: Pioneer 3-DX Robot.

### Experimental results

The simulation is to make the robot navigate freely through the non-convex scenario. As the robot interacts with the environment, the Q-leaning algorithm is executed. The algorithm adjusts the values of the Q matrix using the acquired experience. The interaction of the algorithm and the robot is described in four steps as follows:

**Step 1** The robot takes the information from the sensors and identifies the current status.

**Step 2** Based on the exploration policy, the robot decides the action to be performed and executes it during an established time.

**Step 3** The robot takes information from the sensors and identifies the new state.

**Step 4** The values of the matrix Q are updated based on the reward policy, the current state, and the old state. The states are updated, and it returns to step 2.

The execution of these 4 simple steps is called an episode. For the purpose of this experiment, about 10,000 episodes are executed. The time elapsed through the 10.000 episodes allowed the robot to adjust its parameters and learn to navigate without crashing. Within the robot learning process, some important factors were obtained by observing and analyzing it, such as the number of crashes, the path followed by the robot during the learning process, the convergence of the $Q(s, a)$ values, and the effect of the exploration policy will be analyzed below.

### Path

During the learning process, the robot navigates, exploring different places in the environment. At the same time, the robot maintains a prudential distance from objects, avoiding collisions. The path described by the robot is evidence of the exploration and obstacle skills developed by the robot.
Figure 5.3 shows the robot navigation path through the 10,000 episodes.

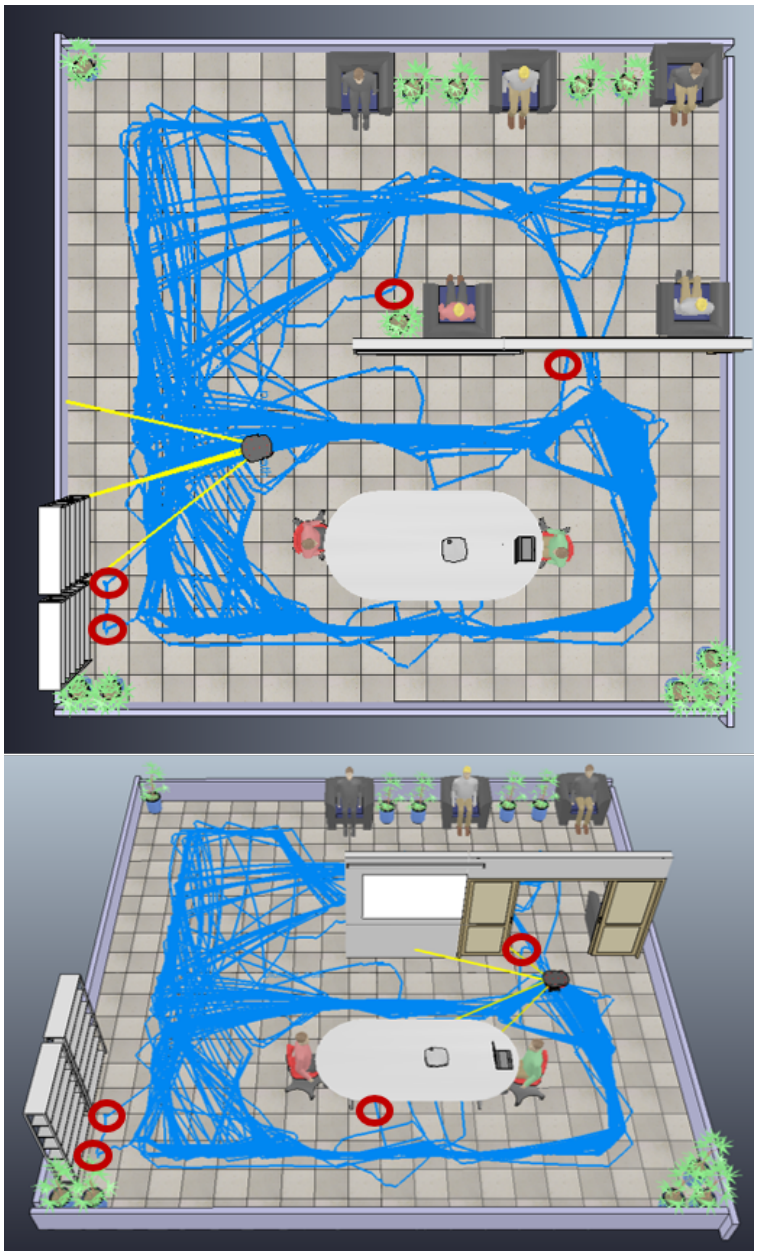


**Figure** 5.3: Generated path by a robot over time in the learning process.

The path followed by the robot in the experiment is depicted in Fig.5.3. The blue line is the

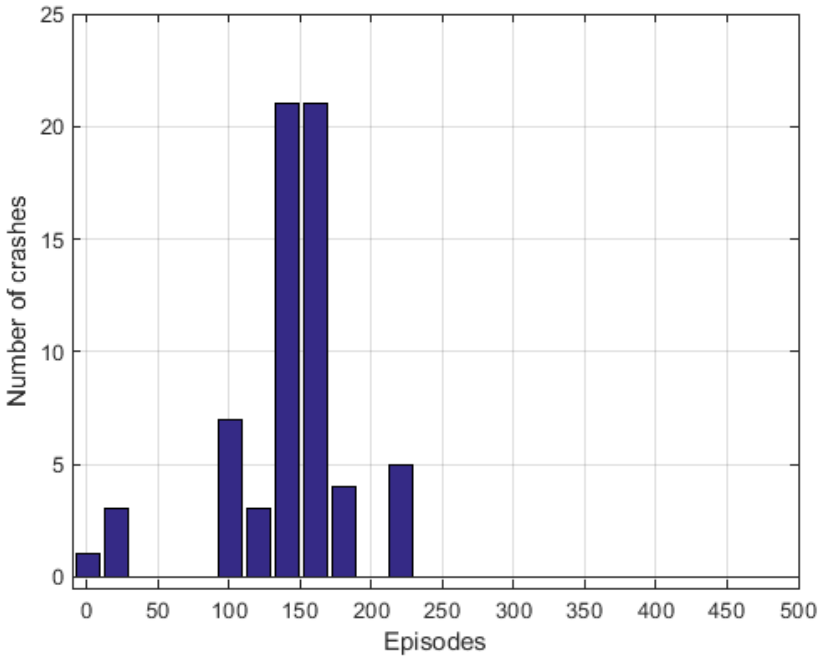



**Figure** 5.4: Robot crashes through the learning episodes.

path, and the red circles are the places where the robot crashes more frequently.

### Crashes

The crash incident is defined as when the robot collides with a scene object or passes too close to the objects or walls, as shown by Figure 5.3. The quantification of the crashes is shown in Fig. 5.4.
Figure 5.4 shows the number of crashes divided by slices of 20 episodes through the first 500 episodes. The system does not register more crash incidents after the $230^{th}$ episode. This result could be interpreted as the robot learn to navigate without crashing, and it used the rest of the episodes to adjust the new $Q$ values to the optimal navigation behavior.

### $Q$-values convergence

The learning process is given by updating $Q$ matrix values over time. After the $230^{th}$ episode, the robot learned to avoid obstacles. However, the robot continued its process of learning and searching for the best actions. The values of the $Q$ matrix reached a stable state where the updates do not generate major changes. As shown in Figure 5.5.
Figure 5.5 shows that after episode 700, the $Q(s = 120)$ values remain practically constant. This means that the system converged, and the robot found the best group of actions to navigate and avoid obstacles. The blue line represents $Q(s = 120, a = 1)$, orange line is

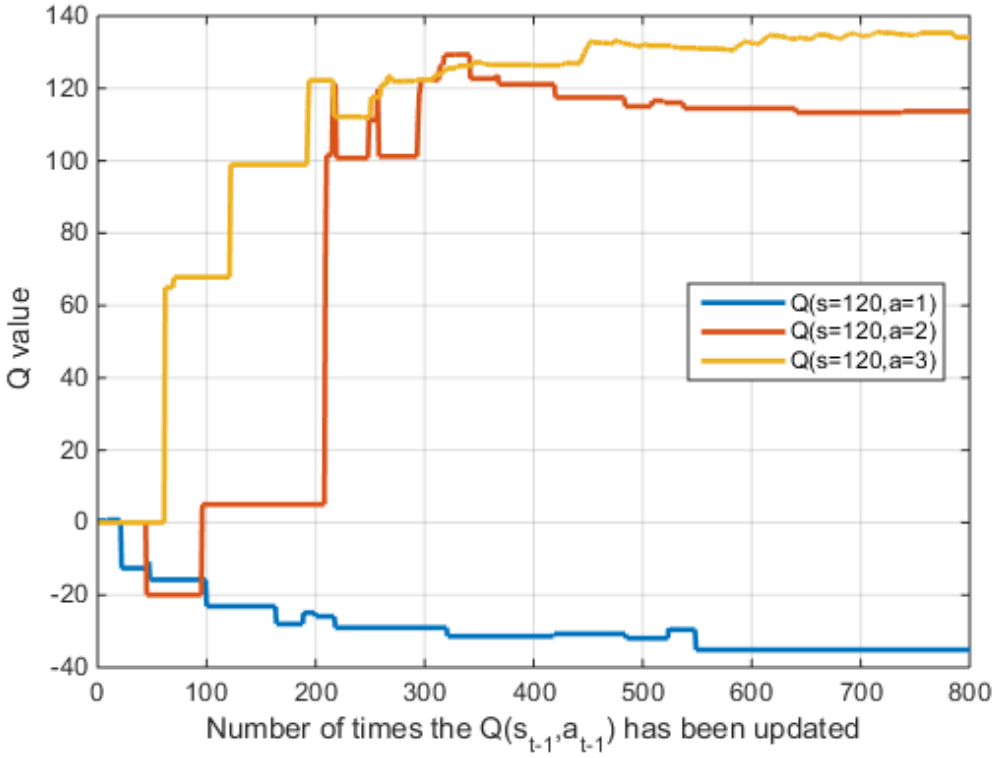



**Figure** 5.5: Q values evolution through the learning process.

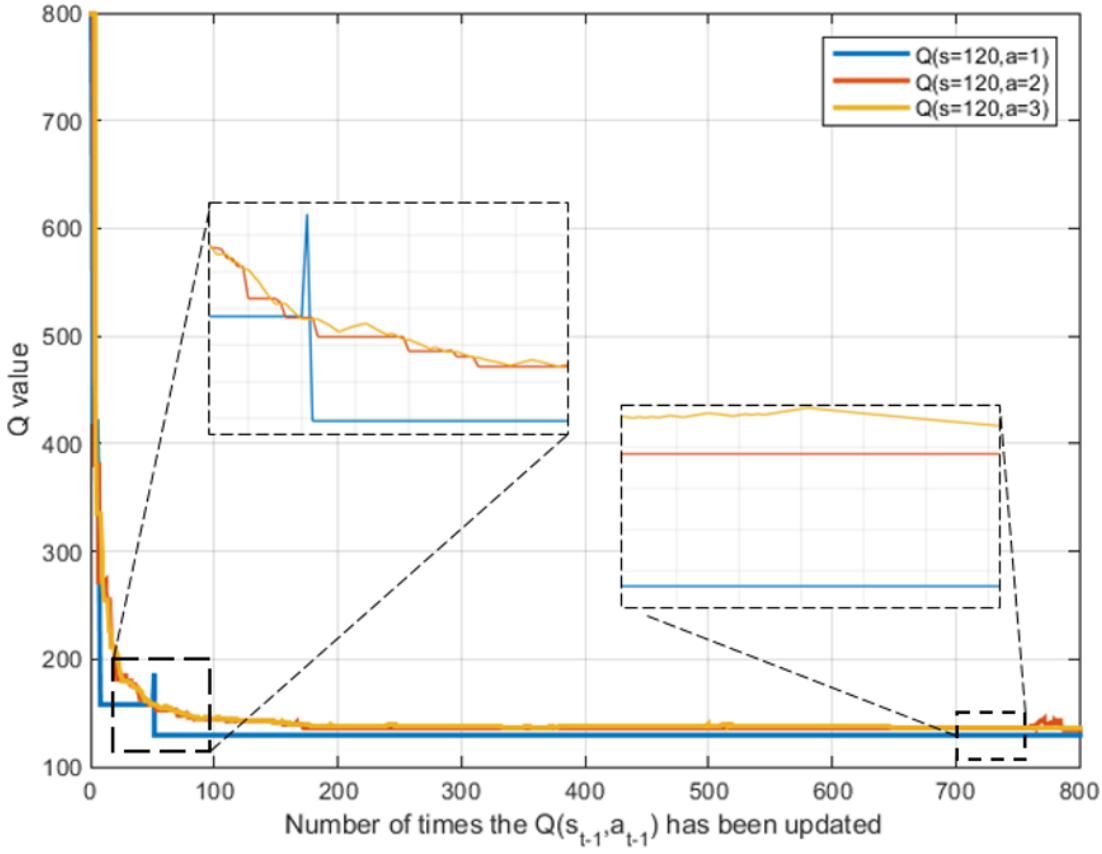



**Figure** 5.6: Explore and exploit policy through the learning process.

$Q(s = 120, a = 2)$, and yellow line is $Q(s = 120, a = 3)$.

### Exploration Policy

This policy allows the robot to alternate between two types of behaviors (explore or exploit). (2.11) shows how the selection of the action that will carry out the behavior selection is made. During the experiment, the robot used the (2.11) to choose the actions, allowing exploring or exploiting. Figure 5.6 shows the three possible selection values for the state's actions.

As described above, the selection values decrease in time, allowing alternating actions. Figure 5.6 shows how the selection values around episode 50 alternate. This values variation allows the system to alternate the action selection and execute an explore behavior. In contrast, around episode 720, the values remain constant, always making the same action selection. This generates an exploit behavior and indicates that the system converged and the robot found the optimal action.

### 5.1.2. Voronoi Tessellation

To verify algorithm functioning, different simulations are performed. In all the figures, the red filled dots correspond to the robots, the red non-filled dots correspond to the markers of the interested robot (in a general view, there is no interest robot), the green dots correspond to the target position of each robot in each iteration, and the blue non-filled dots correspond to the markers of the non-interest robots. Regarding to the background colors employed, there are two complementary views of the situation.

The first one uses white color for the known clear areas, black for the obstacles, and gray for the unknown areas, and the second one uses different colors to identify to which robot is assigned each area (light red for robot 1, light blue for the robot 2, light green for the robot 3, cyan for robot 4 and, light yellow for robot 5) and pink for areas that have been unassigned because there is a marker. Note that the perspective of unconnected robots may be very different if they are not connected, but when they are, the information sharing increases the similarity but the markers assignation.

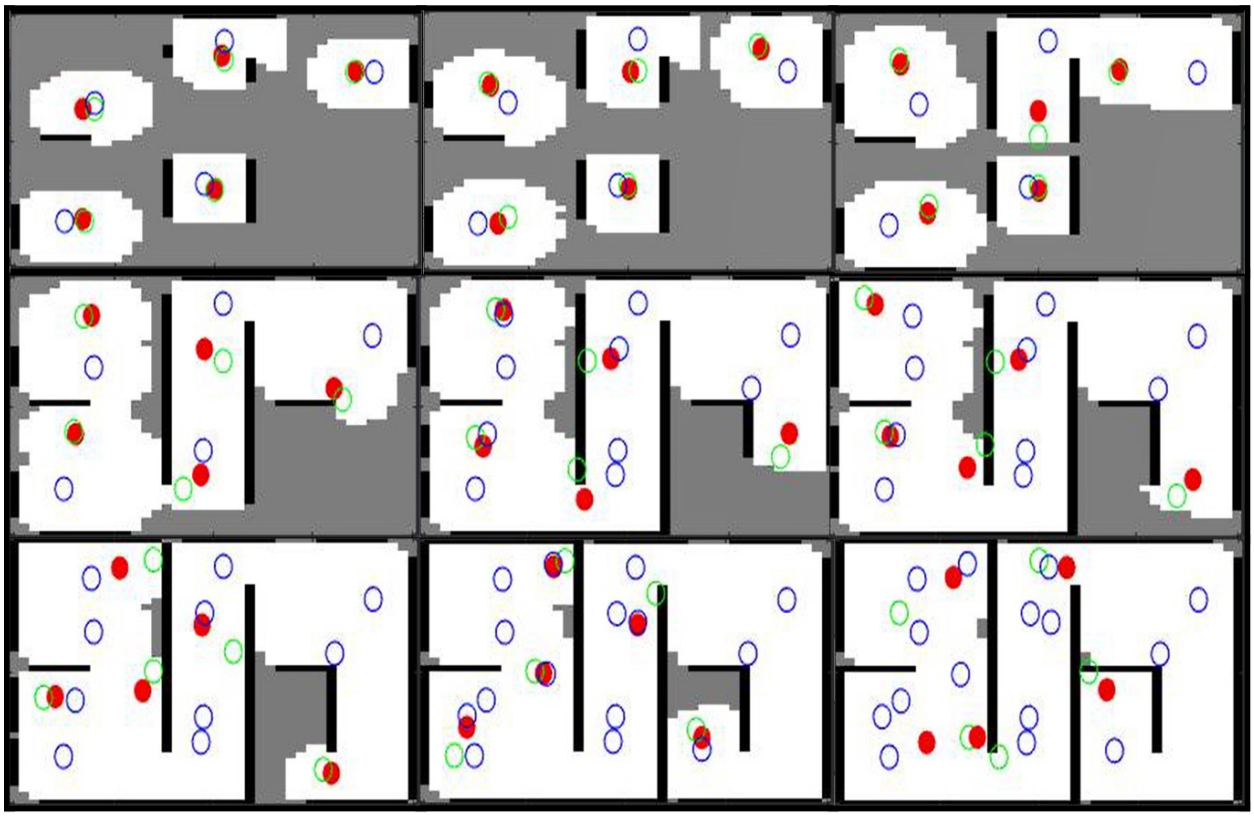


**Figure** 5.7: Global Map sequence, Gotten by the 5 robots.

In Fig. 5.7, it is possible to observe that each of the 5 robots was randomly deployed within the mission environment; each agent starts navigating through the environment looking for their own frontier, realizing a local exploration. Also, as time passes, the number of markers

increases as expected, preventing the robots from exploring areas that were already explored even if they do not know them yet. In this simulation, the end condition is when at least one of the robots determines that the entire map has been explored.

Considering that this determination can be made considering that the unknown area is covered by markers, even if a robot does not yet know the entire area, it can notice when the exploration process has ended. This approach was taken since, in this particular case, it is not important that all the robots know the entire environment. However, if this was required, then it would be enough to establish the full connectivity of the network to guarantee that condition, so setting and tracking a meeting point previously known by the robots may be a possible option.

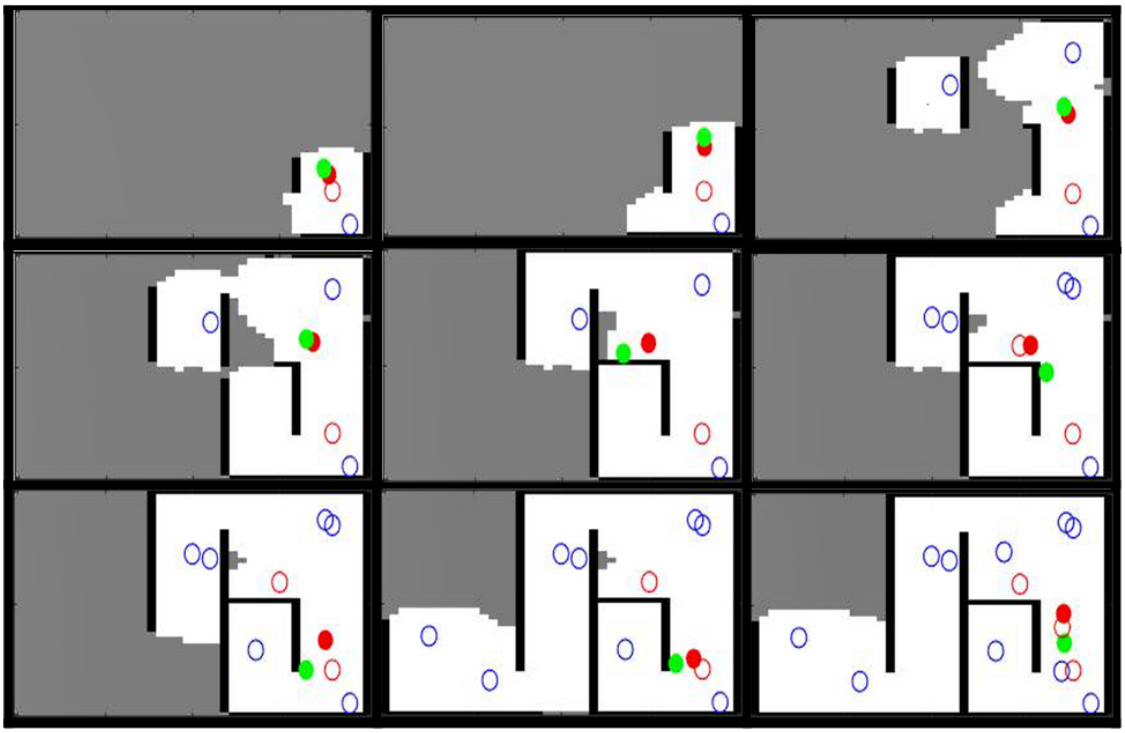


**Figure** 5.8: Exploration Sequence - Robot 1.

On the other hand, in Figure 5.8, Figure. 5.10, Figure. 5.12, Figure. 5.14, and Figure. 5.16, all the robots' individual exploration sequences can be appreciated. It is worth highlighting that, as mentioned earlier, not all the robots know the entire map at the end of the mission. However, it is possible to note that the robots communicating with others during their exploration have more complete versions of the map. Regarding the markers deployment and sense, it is possible to note that the robots never approach the markers of other robots (presented as blue non-filled dots) as expected and that the good markers of each robot also serve as indicators of the path covered by a single robot.

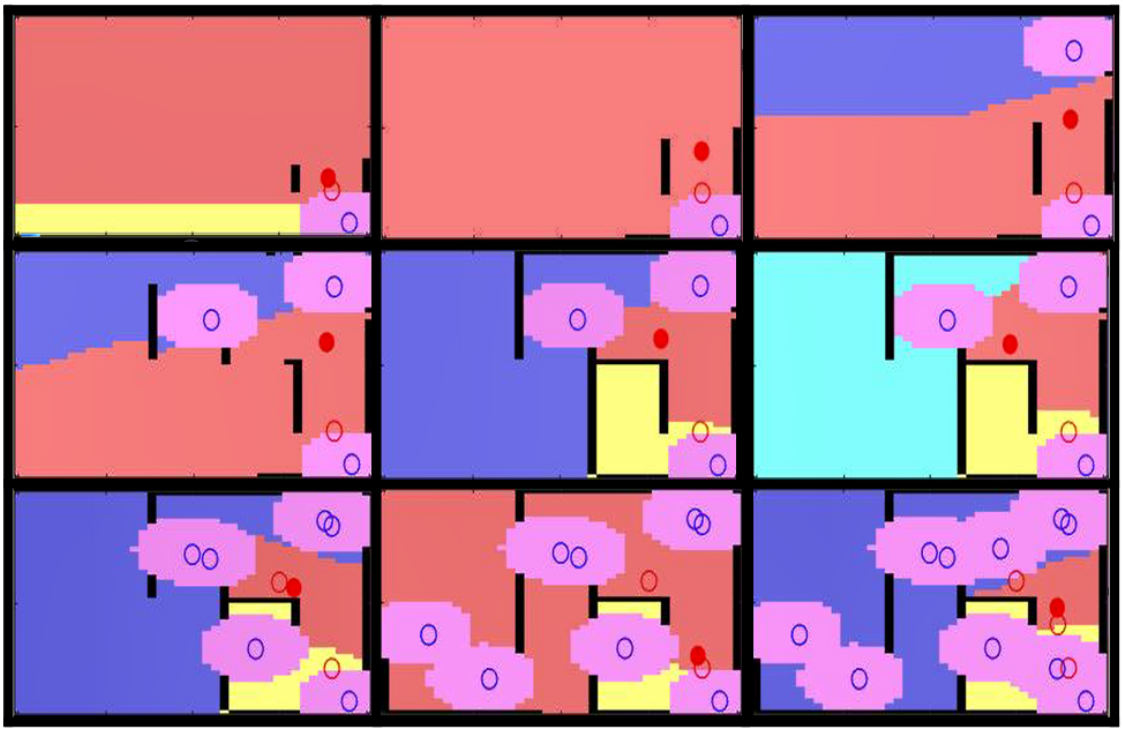


**Figure** 5.9: Voronoi Cells Evolution - Robot 1.

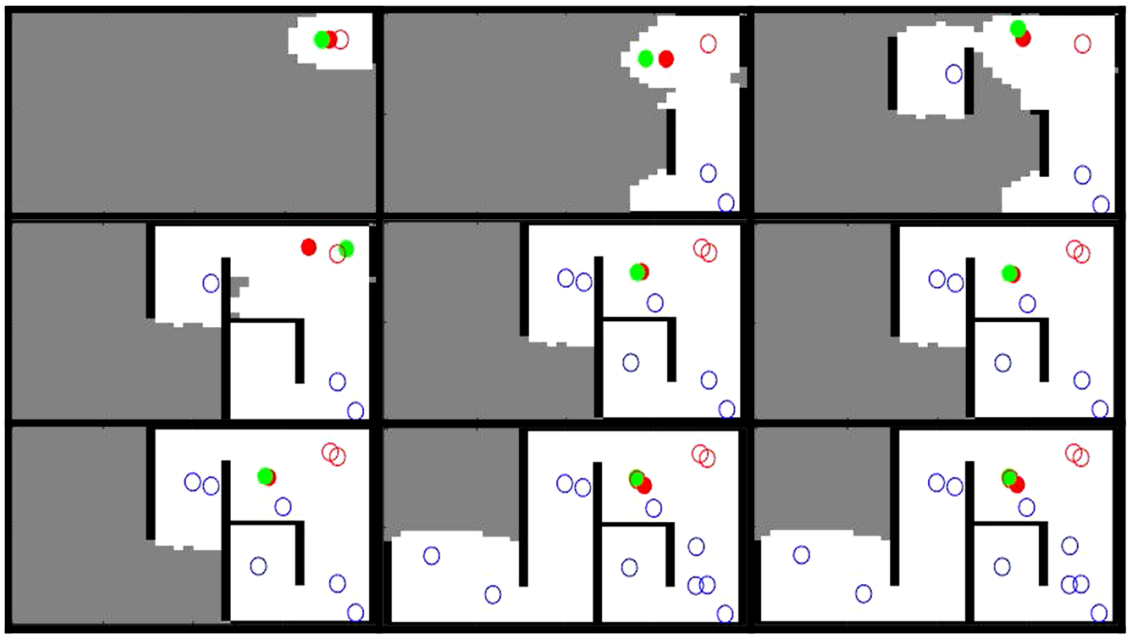


**Figure** 5.10: Exploration Sequence - Robot 2.

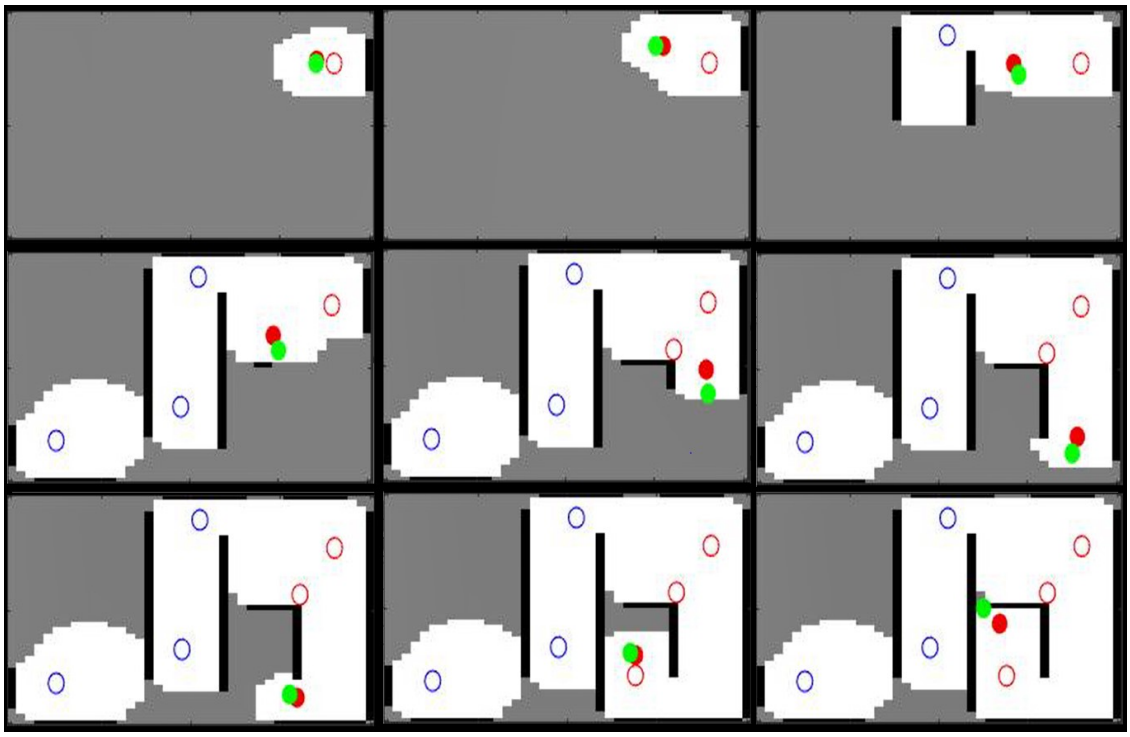


**Figure** 5.14: Exploration Sequence - Robot 4.

Additionally, in Figures 5.9, 5.11, 5.13, 5.15, and 5.17, we present the evolution in time of the areas owned by each one of the robots in the mission. Notably, in those figures, the communication (or its absence) between robots is evidenced when a robot does not have any

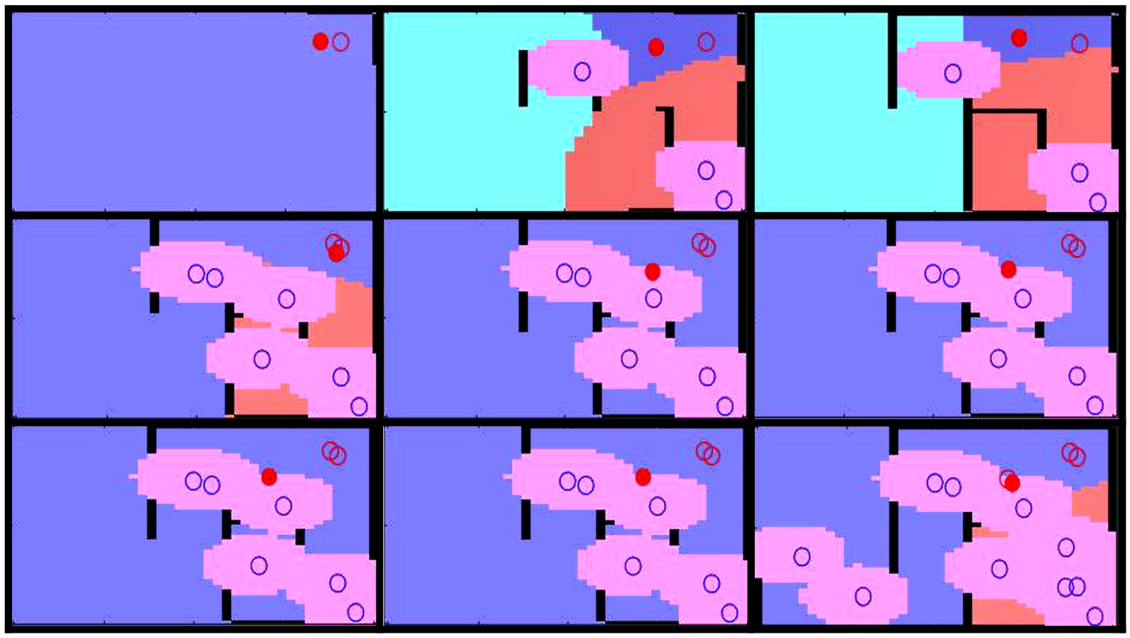


**Figure** 5.11: Voronoi Cells Evolution - Robot 2.

communication link. It considers the whole map as its exploration zone, excluding the areas covered by markers. An interesting fact to note is that as time passes, the markers number increases and the robots lose interest in many areas of the map and focus on exploring their local environment.

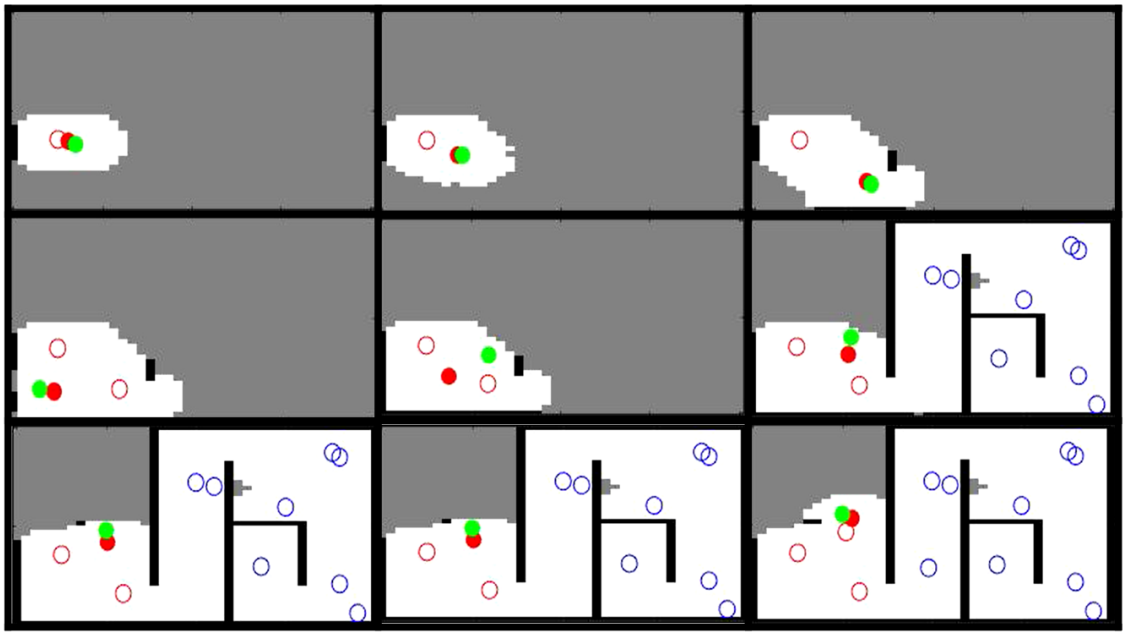


**Figure** 5.12: Exploration Sequence - Robot 3.

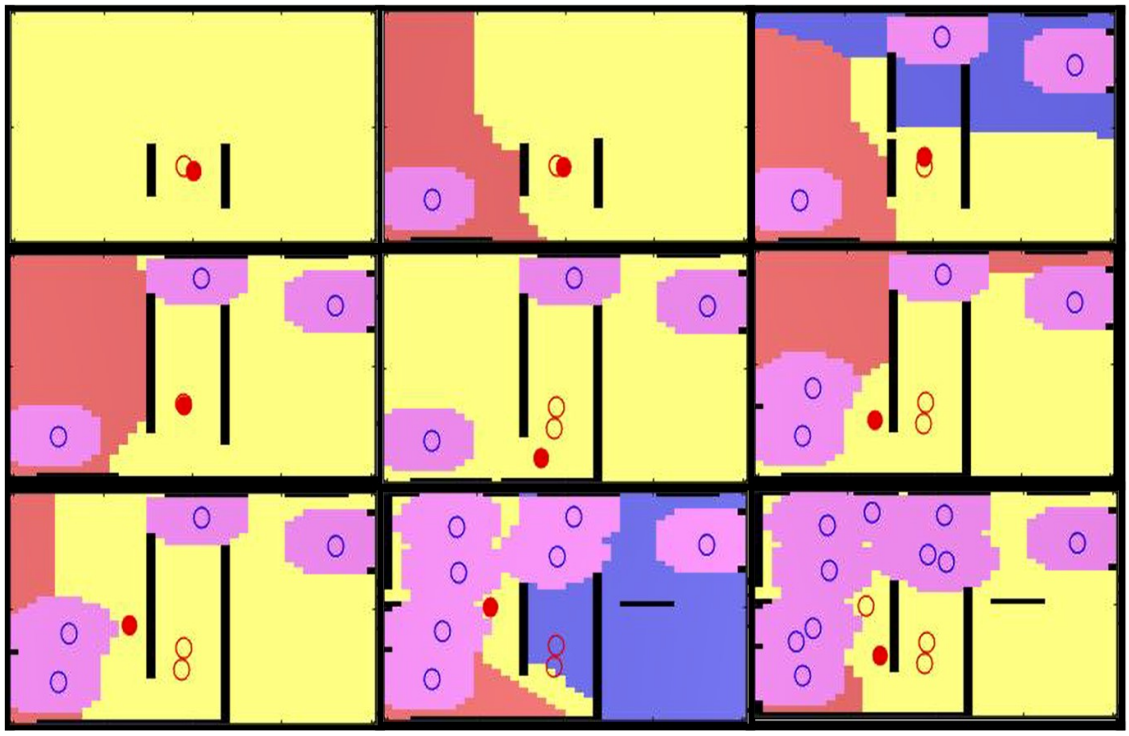


**Figure** 5.17: Voronoi Cells Evolution - Robot 5.

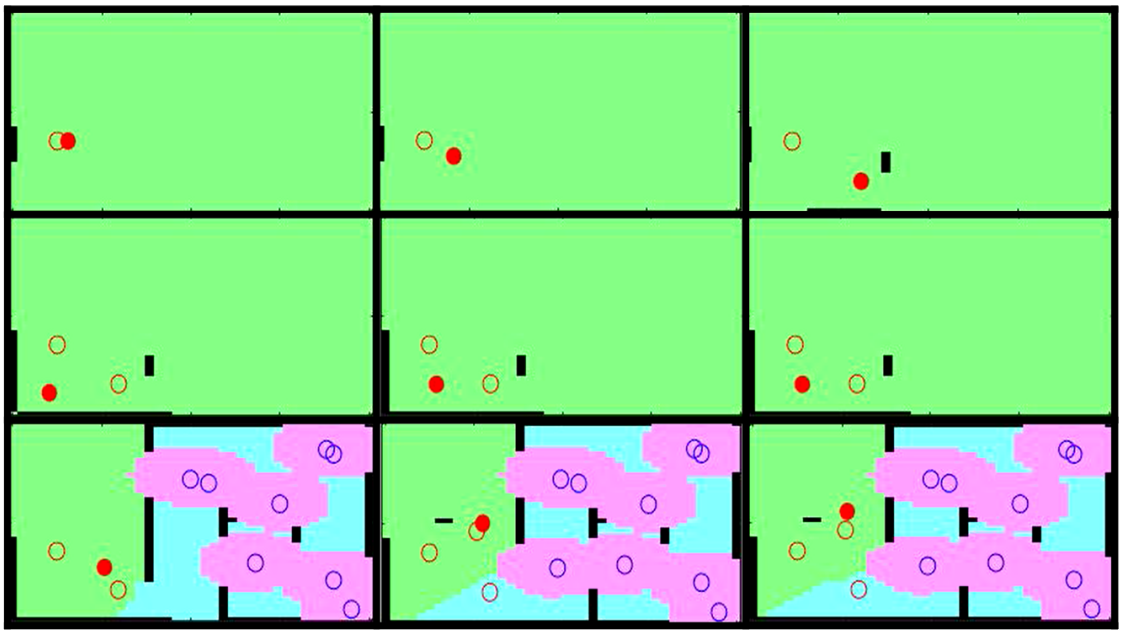


**Figure** 5.13: Voronoi Cells Evolution - Robot 3.

## 5.2. Victim Detection Algorithm

### 5.2.1. Swarm Navigation

To perform experiments for swarm navigation, sub-swarm generation, and victim detection, the multi-agent model previously explained was implemented using Matlab and V-Rep. The Matlab implementation evaluated the proposed mathematical model in different scenarios and behavioral cases as previously named. V-Rep was used to perform simulations of the proposed swarm behavior model over 3D environments with a virtual model of real robots like Drones. The principal objective of the V-Rep simulations is to validate the proposed system using a more realistic environment, as shown in Figure 5.18.

Six different experiments were developed to verify the proposed model in several navigations, sub-swarm generation, and victim detection situations. Every experiment is shown in both simulation environments, as named previously (Matlab and V-Rep). Both types of simulations were performed using 23 agents or Drones, respectively. According to the case, the trace left by the agent is shown as a solid line behind every agent or Drone. Those lines show the path every swarm robot follows and depict the collective behavior generated by the swarm computational model. In Matlab simulations, the green circles represent the initial locations of the swarm's agents, the red-filled circle is the agent's current position, and the empty red circles are a sample location in the agent path through the simulation time. Next, every experiment is explained in detail, and the results are analyzed.

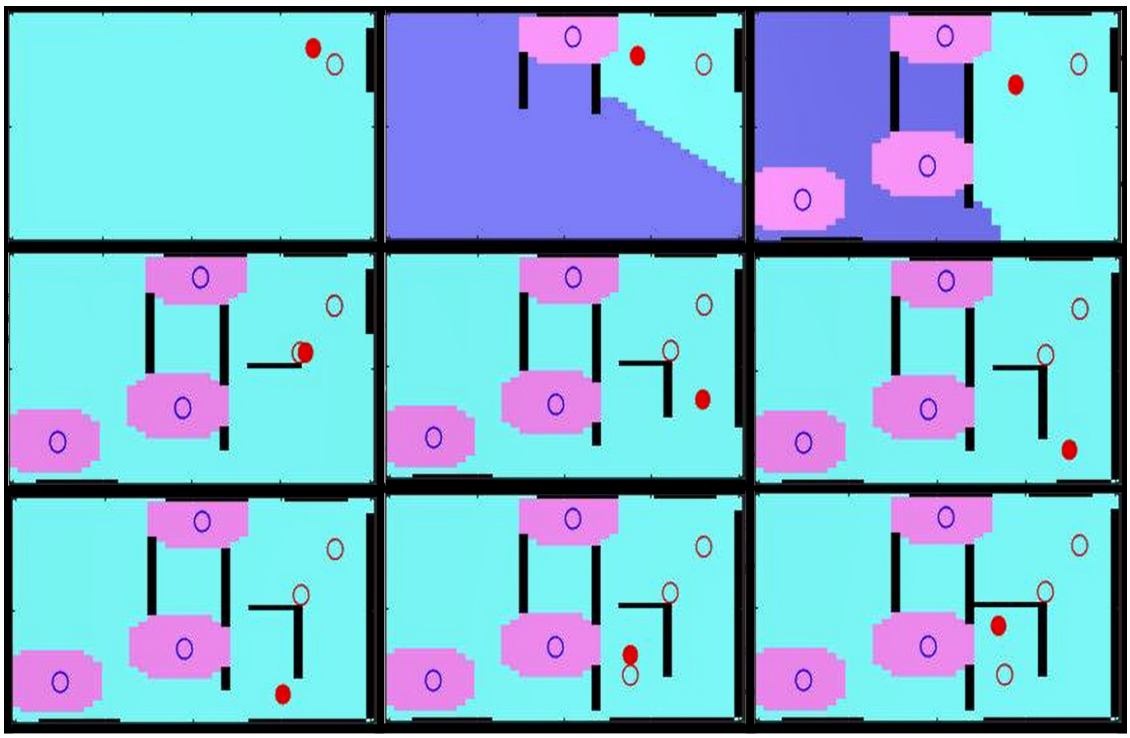


**Figure** 5.15: Voronoi Cells Evolution - Robot 4.

### Obstacle avoidance:

The first version of this experiment is performed using 23 agents, and two cases of single obstacle avoidance are performed. The first case is a small obstacle, as Figure 5.19 shows. This part of the experiment depicts how the swarm goes around the obstacle to avoid it. In the V-Rep simulation, the small obstacle is represented by a bunch of trees. This simulation shows how the obstacle is avoided by the swarm's drones flying around the trees. This is possible because the obstacle is relatively small, and the agents can tolerate the obstacle between the attraction forces. The second version uses a big obstacle, as depicted in Figure 5.20. In the second case, the agents avoid the obstacle by taking a side path. The big obstacle is represented by a building in V-Rep-simulation, where the swarm of drones navigates, describing a side path to the building. This avoiding obstacle strategy occurs because there is insufficient space between the attraction forces to allow broad obstacles to stay between the agents.

### Multiple Obstacles:

This part of the experiment depicts how the swarm goes around several obstacles to avoid them. This is possible because the obstacles are relatively small, and the agents can tolerate obstacles between the attraction forces, as depicted by Figure 5.21. This case uses nine obstacles distributed throughout the area between the start and goal points. For the virtual

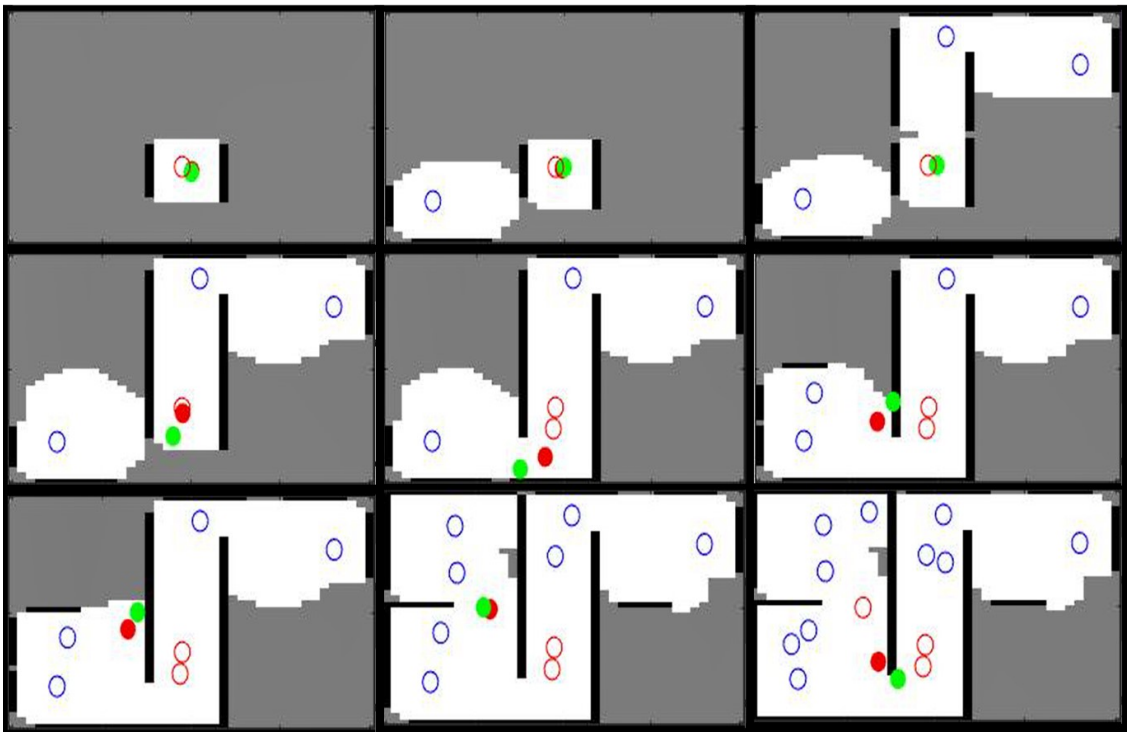


**Figure** 5.16: Exploration Sequence - Robot 5.

environment, obstacles are represented by nine trees distributed along the search area. It forces the swarm to navigate through obstacles while avoiding them and moving toward the goal point. Figure 5.21 shows the agents navigating and exploring the zone. At the same time, the path described by every agent is drawn, depicting the explored area.

### Victim Localization:

This test uses a flat terrain to show how the swarm localizes victims. This process is accomplished with the use of the sub-swarm generation process. Once an agent has localized a victim, it stops near the potential victim. At the same time, the swarm stops because of the attraction force between the agents. At that moment, the "Sub-Swarm Generation"process detaches the agents that found victims. The detached agents create a new swarm surrounding the victims. The original swarm restarts its movement toward the target point and leaves behind the swarm agents that are in charge of the victims. Figure 5.22 shows both simulation styles, where the agents stop near the victims and surround them while the swarm leaves them behind.

### Navigation and Victim Localization:

This is a complete case where the swarm navigates through an area full of obstacles and some places with potential victims. The experiment uses 23 agents and six victims distributed in

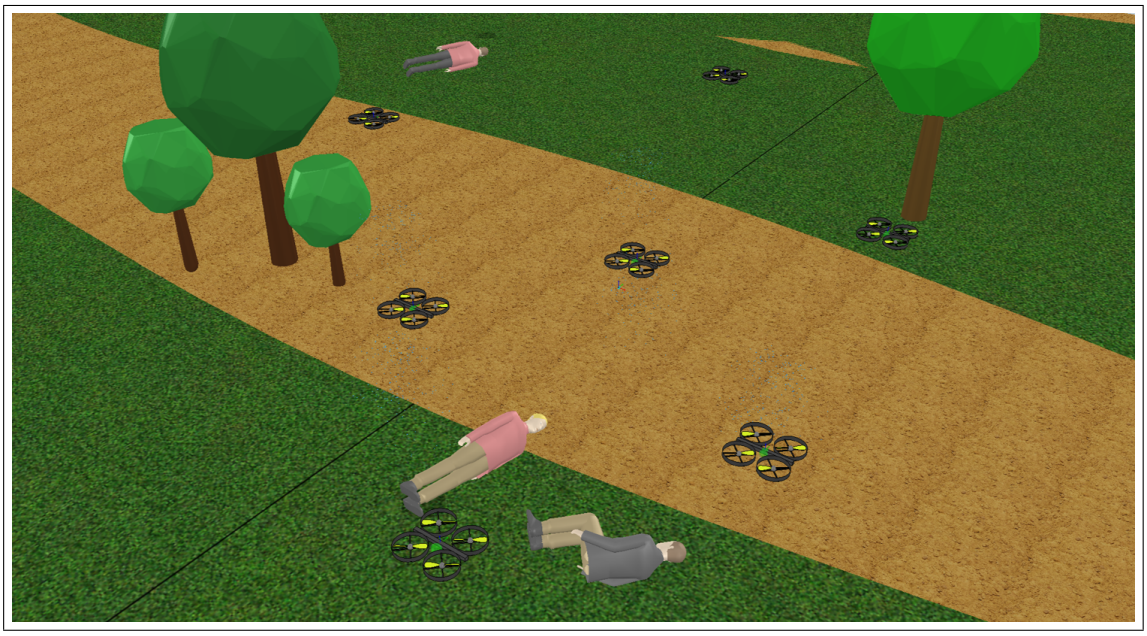
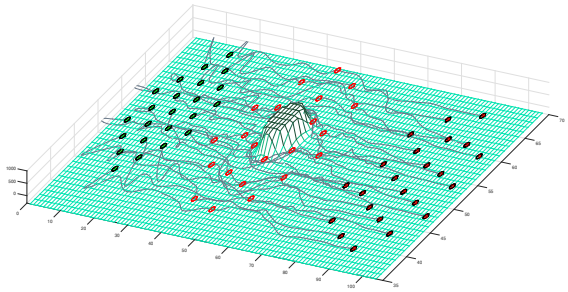


**Figure** 5.18: Simulation in a realistic environment and robot platforms using V-Rep.

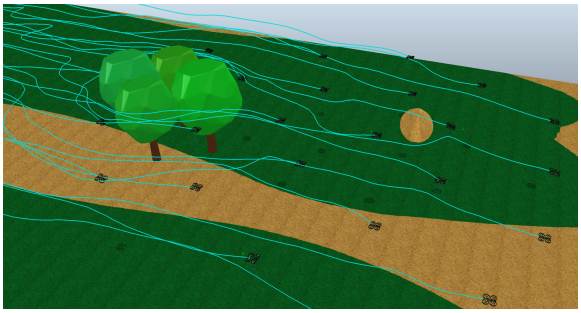
5 victim places, given there are two victims in the same place as shown by Figure 5.23. This case depicts how the swarm covered the area navigating through obstacles and localizing potential victims simultaneously. The simulation shows how the proposed model generates sub-swarms with more drones in places with more victims or where the probability of finding victims is greater. This case is represented in the virtual environment simulation, where there is a place with two victims, and the model assigns a sub-swarm with six drones. This victim's place has more assigned drones than the other places.

#### Convergence analysis of the distributed estimation consensus:

To perform a convergence analysis, it is worth mentioning that the estimation algorithm was carried out in a computer with the following specs: 7-th generation core-$i7$ microprocessor, 32 GB installed memory (RAM), and NVIDIA GEFORCE $940MX$. Several simulations were performed to know the algorithm performance under different parameter values. The principal parameter considered is the number of agents able to take a measurement, which in this case varies from 3 to 10. Additionally, in the interest of proving the robustness of the estimation algorithm, the measurement values are settled randomly. In this way, the algorithm's convergence depending on the number of agents is shown in figure 5.24. It is appreciable how the convergence is affected by the number of agents. The more agents give measurements, the more time the algorithm converges.
As mentioned, several simulation scenarios were performed to know how the variation of parameters can affect the algorithm convergence. In fact, 30 simulations were carried out for each variation in the number of agents (3, 5, 7, and 10). The results are summarized

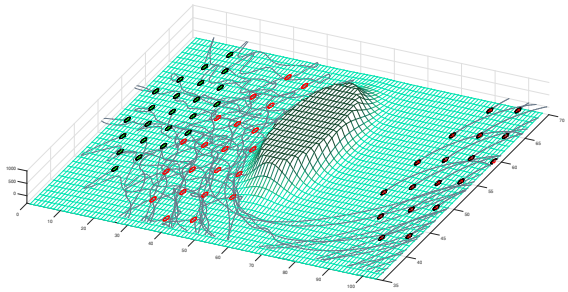


(a) Small-obstacle avoidance in Matlab.

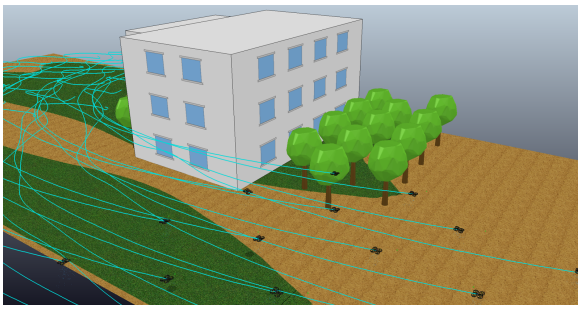


(b) Small-obstacle avoidance in a virtual environment.

**Figure** 5.19: Simulation of robots avoiding small obstacles.



(a) Large-obstacle avoidance in Matlab.



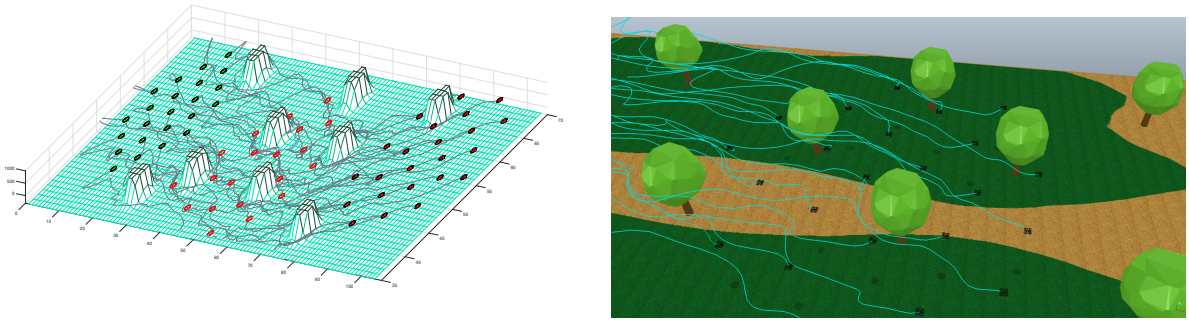
(b) Large-obstacle avoidance in a virtual environment.

**Figure** 5.20: Simulation of robots avoiding large obstacles.

in Table 5.1, where it is appreciable that increasing the number of agents also increases the iterations number to converge as the convergence time does. Considering this, the mean value $\mu$ and standard deviation $\sigma$ for each case was calculated. In which the largest number of experiments converge between the mean value and the standard deviation, this corresponds to approximately 68 percent of the experiments converging within the interval $[\mu-\sigma, \mu+\sigma]$. The fact that the standard deviation is not large numbers shows that the algorithm is robust under parameter changes.

**Table** 5.1: Convergence data depending on the number of agents.

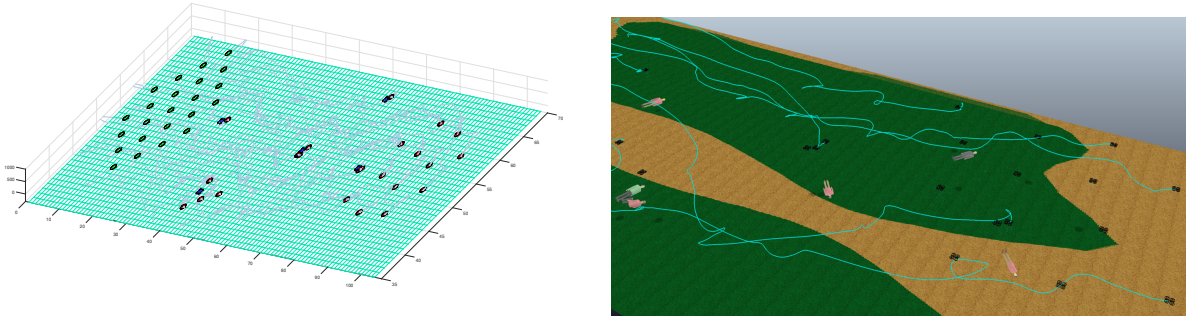
| Agents Number | Average Convergence Iteration | Standard Deviation | Average Convergence Time |
|---|---|---|---|
| 3 | 176.6 | 8.3 | 90.6(mS) |
| 5 | 404.9 | 22.4 | 103.9(mS) |
| 7 | 602.6 | 39.4 | 144.5(mS) |
| 10 | 1162.4 | 102.3 | 226.6(mS) |



(a) Multi-obstacle avoidance in Matlab.

(b) Multi-obstacle avoidance in a virtual environment.

**Figure** 5.21: Simulation of robots avoiding multi-obstacle scenario.



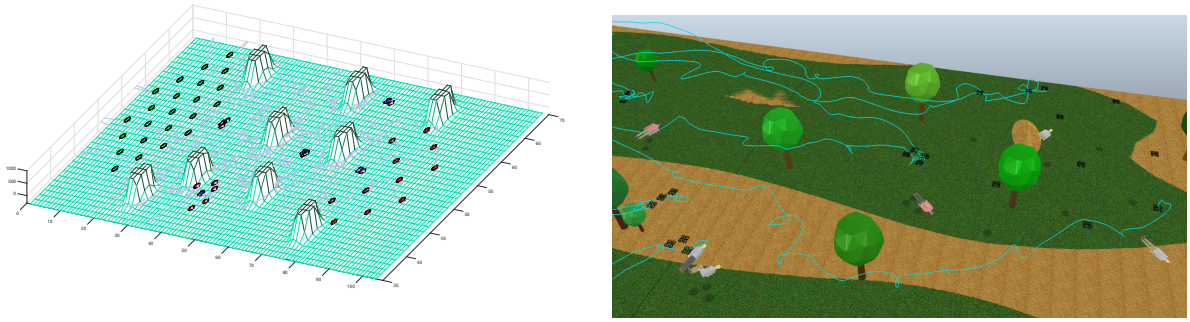
(a) Victim localization in Matlab.

(b) Victim localization in a virtual environment.

**Figure** 5.22: Victim localization in convex space.

### Effective Coverage Area

An important parameter that should be considered is the effective coverage area. This parameter is critical because developed tasks are related to the search and localization of human beings in a disaster zone. The idea of swarm drone exploration is supported by the aim to inspect every tiny space in the disaster zone at least once by at least one drone. The most insignificant failure in the exploration can be translated into the loss of human life, so it is preferable that more than one drone inspect the same area within the disaster zone. Generating in this way more robustness to the task of search and localization. This search and rescue approach greatly emphasizes using a drone swarm, which seeks to increase the probability of finding victims. The effective coverage area is defined as a corridor through which the swarm travels. This corridor is as long as the area to explore, and the width depends on the number of drones since the width of the area depends on the number of drones that are located across the width of the swarm. The width of the coverage area is directly related to factors such as the number of drones, comfortable distance between drones ($d$), and the radius of sensors coverage ($r$) used to perform the inspection and search tasks. To analyze



(a) Navigation and victim localization in Matlab.

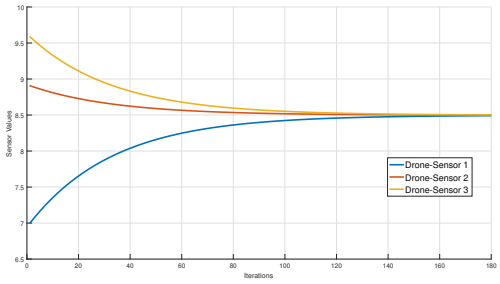
(b) Navigation and victim localization in a virtual environment.

**Figure** 5.23: Navigation and victims localization in an environment with the presence of obstacles.

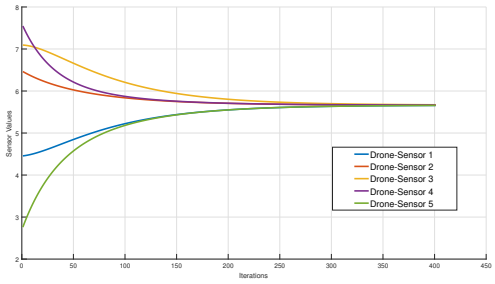
the exploration degree of the covered zone, several experiments were carried out in which the parameter $r$ was varied about $d$ as shown in table 5.2.

The experiment's aim is the percentage calculation of the unexplored and explored area by at least one, two, three, or more drones. Figure 5.25 shows the simulation of the experiment and the representation of different degrees of exploration for the areas covered by the drone in its trajectory toward the endpoint.

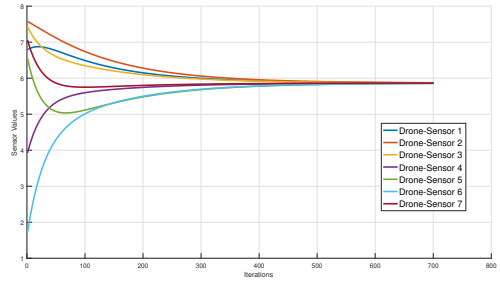
Figure 5.26 shows different experimental cases where several gray color levels depict the covered area by the swarm through the path to the endpoint. Figure 5.26 shows how explored area changes according to the relationship between $r$ and $d$. In Figure 5.26 from $(a)$ to $(d)$ is showed the covered area using a swarm of 23 agents. The covered area increases according to $r$ value approaches $d$. Figure 5.26 $(e)$ and $(f)$ are shown how the explored area is affected by the swarm agent number. The covered area percentage in relation to the values of $r$ and $d$ is depicted by Table 5.2. There are described seven experiments where the rate between $r$ and $d$ is studied from $r = d/15$ to $r = d$. From Table5.2, it is concluded that explored area increases significantly through the increase of the $r$ factor. The equilibrium point is reached when $r = d/4$ because at this point, where at least two drones explore 100 % and 80 % of the entire area.



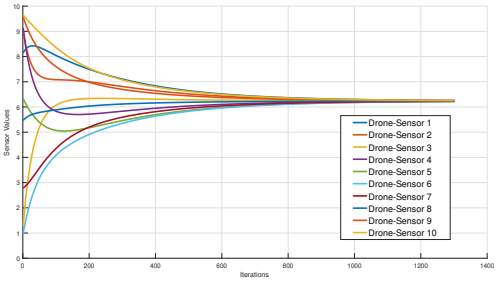

(a) Measurements convergence with 3 agents.




(b) Measurements convergence with 5 agents.




(c) Measurements convergence with 7 agents.




(d) Measurements convergence with 10 agents.

**Figure** 5.24: Convergence analysis with variation in the number of agents and random initial values.

**Table** 5.2: Percentage of the covered area according to different rate values between $r$ and $d$.

| r | Uncovered | Covered | +2 Drones | +3 Drones | +4 Drones | +5 Drones |
|---|---|---|---|---|---|---|
| d/15 | 43.10 % | 56.89 % | 9.55 % | 0.0014 % | 0.0 % | 0.0 % |
| d/8 | 13.51 % | 86.48 % | 42.01 % | 4.35 % | 0.0069 % | 0.0 % |
| d/5 | 1.73 % | 98.26 % | 73.28 % | 24.57 % | 3.06 % | 0.0 % |
| d/4 | 0.0 % | 100 % | 87.77 % | 54.55 % | 20.53 % | 0.20 % |
| d/3 | 0.0 % | 100 % | 94.34 % | 80.14 % | 52.72 % | 14.56 % |
| d/2 | 0.0 % | 100 % | 98.17 % | 94.04 % | 85.42 % | 71.26 % |
| d | 0.0 % | 100 % | 100 % | 100 % | 97.57 % | 96.11 % |

Table 5.3 shows how the areas explored by more than one drone increase as the number of drones increases. This result is quite logical, considering that increasing the number of drones also increases the number of rows in the swarm. In this case, more drones will follow roads similar to the path taken by the drones of the first rows, and they will explore similar disaster zones. The relation between $r$ and $d$ was selected as $r = d/4$ to ensure the covered area is close to 100 % and focus the results on the percentage of areas explored by more than one drone.

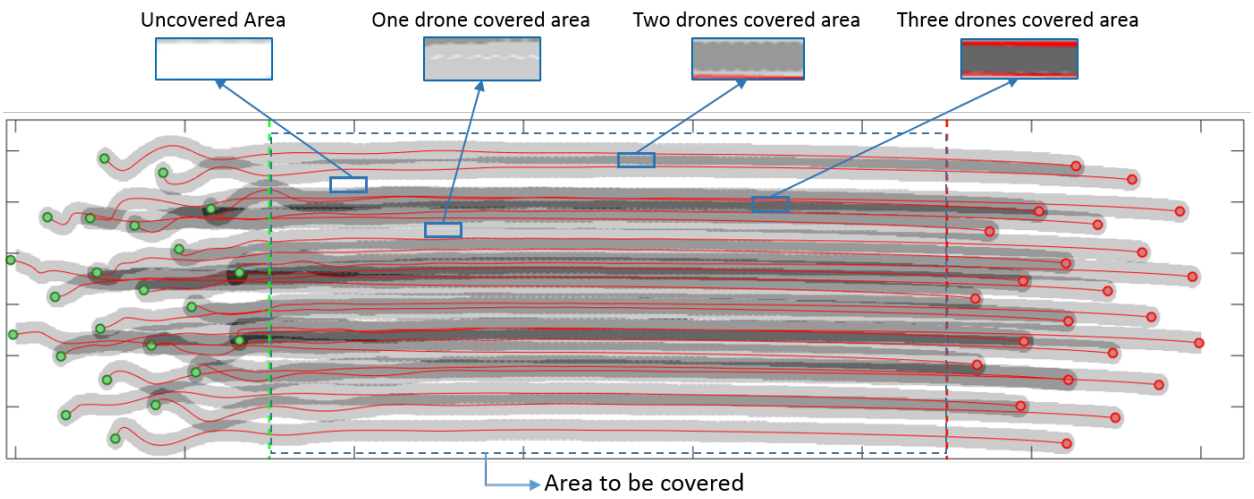


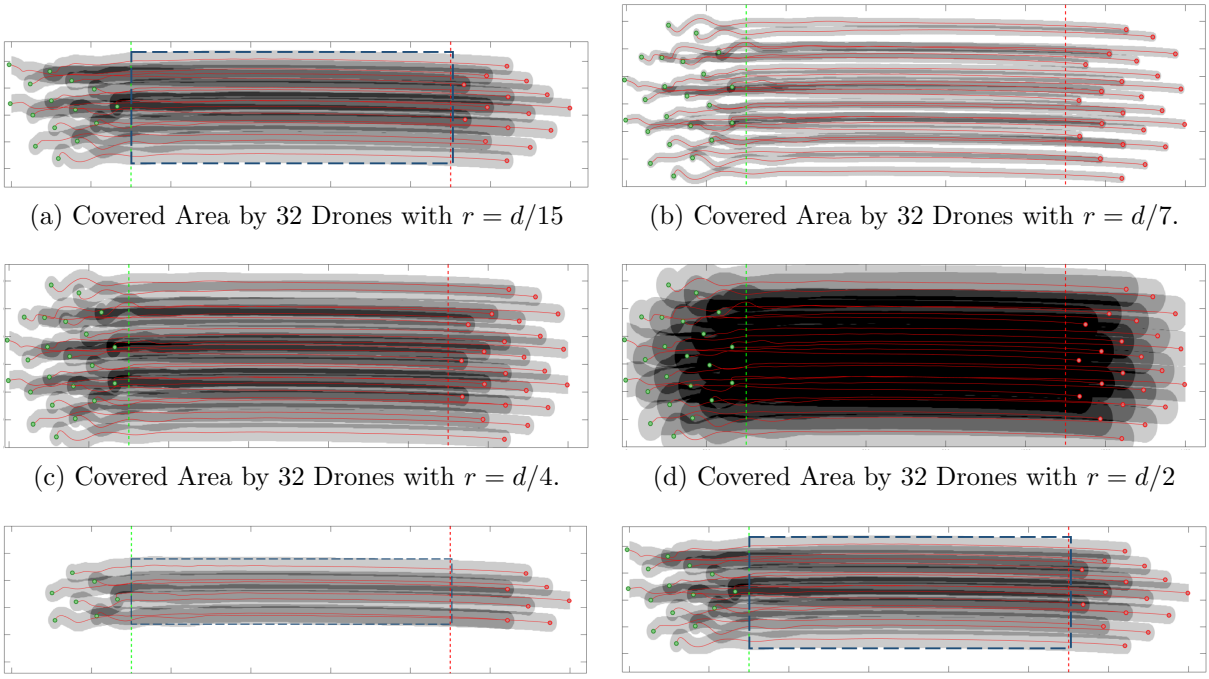

**Figure** 5.25: Formation control applied to the three agents sub-swarm generated by localization of a victim.

**Table** 5.3: Percentage of the covered area according to different swarm sizes and $r = d/4$.

| Drone Number | Uncovered | Covered | +2 Drones | +3 Drones | +4 Drones | +5 Drones |
|---|---|---|---|---|---|---|
| 7 | 0.0 % | 100 % | 93.14 % | 16.63 % | 0.01 % | 0.0 % |
| 12 | 1.02 % | 98.87 % | 83.88 % | 29.8 % | 5.03 % | 0.0 % |
| 17 | 0.0 % | 100 % | 89.10 % | 48.48 % | 11.87 % | 0.0 % |
| 23 | 0.0 % | 100 % | 87.77 % | 54.55 % | 20.53 % | 0.20 % |

### 5.2.2. Sub-Swarm Formation Control

This experiment shows two cases of possible victims with different detection probabilities. The first case presents two victims in an open field with a high probability of being detected by the drone sensors, as shown in Figure 5.27. Given that victim detection is easy in this case, the swarm is made up of 6 agents located in the vicinity of the victims. After Sub-swarm creation, the consensus and control formation algorithm is performed as shown in Figure 5.27. As explained previously, the consensus looks to approach every drone to the victim, reducing the uncertainty of the sensing factor over the possible victim. Additionally, the formation control redistributes the drones in a circle around the victim to better assist the victims and balance the sensor measurements. The second case presents a victim with the body partially covered and a sub-swarm of three agents. Given the difficulty of victim detection, the number of agents for this case is lower than in the first case. Once the victim is detected, the sub-swarm executes the consensus and control formation processes, as shown by Figure 5.28.



(a) Covered Area by 32 Drones with $r = d/15$

(b) Covered Area by 32 Drones with $r = d/7$.

(c) Covered Area by 32 Drones with $r = d/4$.

(d) Covered Area by 32 Drones with $r = d/2$

(e) Covered Area by 7 Drones with $r = d/4$.

(f) Covered Area by 15 Drones with $r = d/4$.

**Figure** 5.26: The covered area by a different number of drones with several relation values between $r$ and $d$.

### 5.2.3. Victim Detection Using Neuronal Networks

To perform experiments where visual victim detection algorithms can be evaluated was performed a virtual scenario with trees, human victims, fire, and Quadrotors in uneven terrain. The scenario was developed using a combination of Matlab, Python, and V-Rep. Matlab was used to implement the mathematical model for navigation and consensus algorithms. Python performed the CNN model and was responsible for detecting human victims. Finally, V-Rep is a Virtual Robotics Environment used to develop a virtual disaster scenario where Quadrotor models can be used in SAR operations. The principal reason to perform these experiments in a virtual environment simulation is because of the difficulty of having a real disaster scenario where it was possible to achieve this kind of experiment, as depicted previously by [33].

Each Quadrotor has cameras pointing to the front and the other to the floor as a sensing system whose principal aim is human victim detection. The image processing task is performed by a CNN in charge of image analysis, focusing on identifying potential human victims. The topology of CNN consists of 3 convolutional neural layers and two fully connected layers. The convolutional layers have 256 filters, each one and $5 \times 5$, $3 \times 3$, and $3 \times 3$ as kernel sizes, respectively. The fully connected layers have 512 and 256 neurons, respectively, with a Linear Rectifier as the activation function. Finally, the output layer has two neurons in charge of providing the certainty level related to the human victim detection or absence of

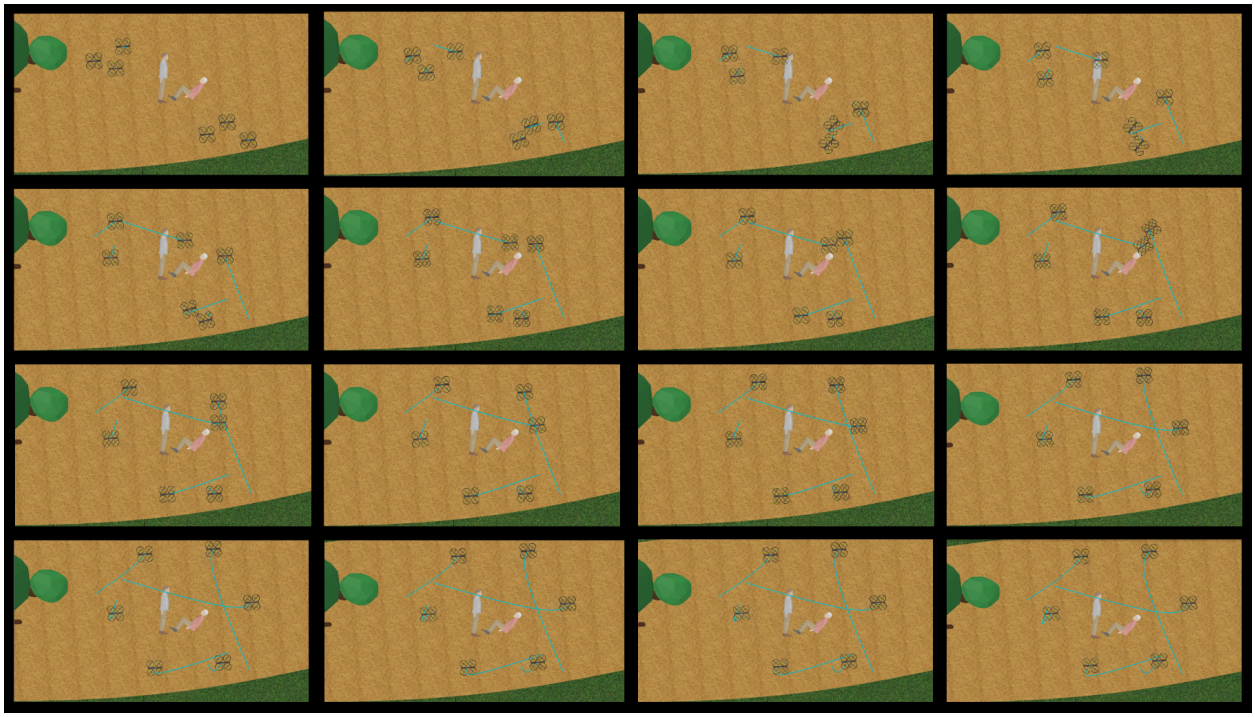


**Figure** 5.27: Formation control applied to 6 agents sub-swarm generated by localization of a victim.

it. As depicted in figure 5.29, three cases of victim detection are shown with their respective Victim Detection Levels (VDL).

Once the Quadrotors are determined as a part of the sub-swarm, they all form a consensus where everyone navigates through the area near the potential victim and covers the more extensive possible area around it. While formation consensus is performed, the victim detection certainty level changes according to the visibility and proximity of the Quadrotor to the potential victim, as depicted by figure 5.29.

Figure 5.30 shows the path followed by six Quadrotors that conform to sub-swarm and the victim detection level of three of these Quadrotors during the formation consensus.

As shown by figure 5.30, the measurements provided by different quadrotors about victims' existence in the immediate area can be confusing and dissimilar. For instance, figure 5.30 depicts three different quadrotor cases where D1 is a Quadrotor that starts its navigation with a total lack of evidence of victim detection. However, while the formation consensus is performed, victim detection improves considerably. D4 depicts a low and constant detection level all the time. Finally, D5 is a Quadrotor that loses visual contact with victims but, over time, recovers some certain detection level.

This possible lack of agreement between all the sub-swarm Quadrotors demonstrates the need for an estimation consensus to have a concerted victim detection level.

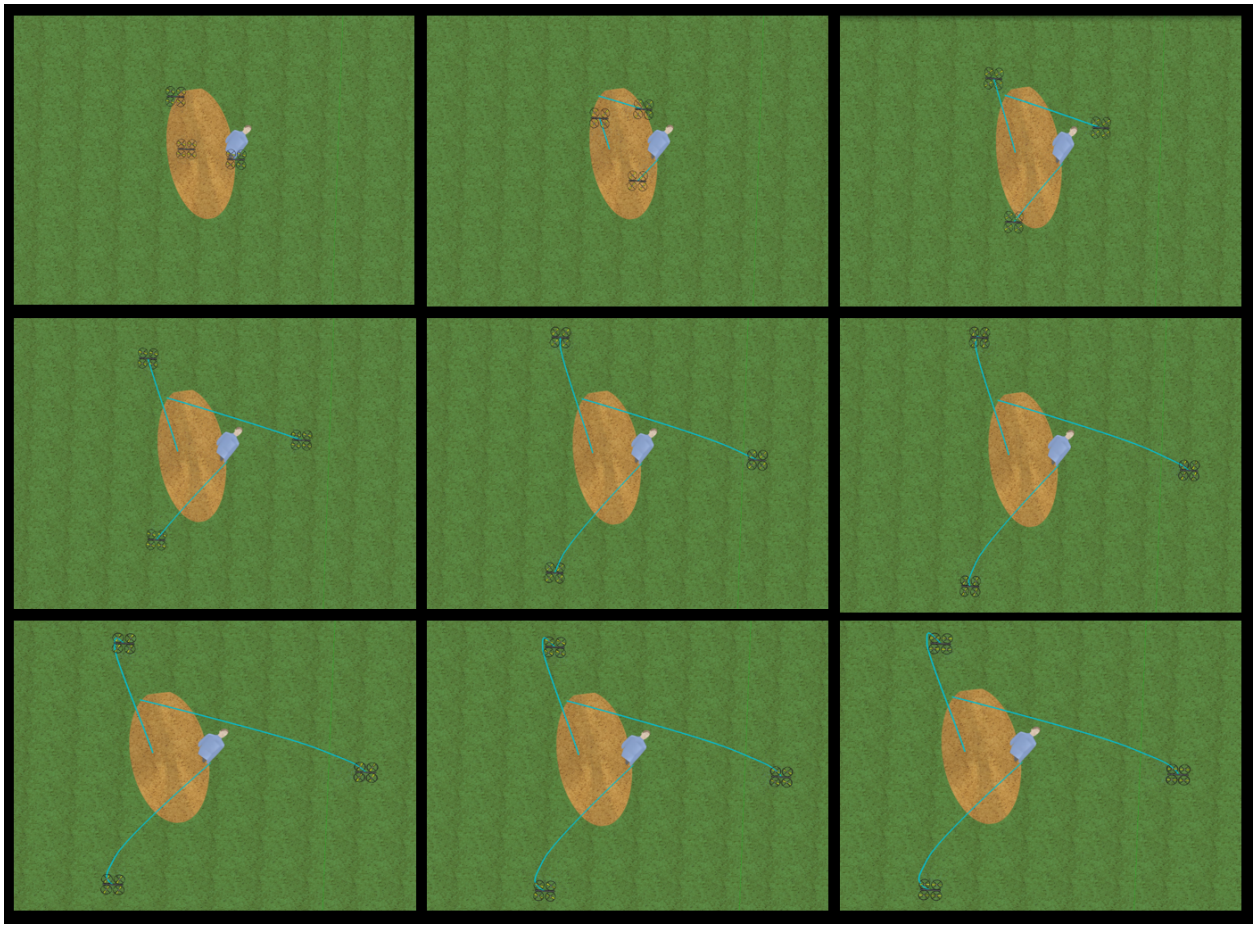


**Figure** 5.28: Formation control applied to the three agents sub-swarm generated by localization of a victim.

### 5.2.4. Estimation Consensus

As depicted in a previous section, the single measurement provided by a single quadrotor has a considerable discrepancy compared with the rest of the sub-swarm agents. This discrepancy is logical if it is considered that a disaster place is a chaotic environment where a measurement can be affected by fire, debris, interference, and landslides. Following the experiment, the estimation consensus finds an agreement point among different sub-swarm agents.

Figure 5.31 shows the victim detection value of every Quadrotor and the estimation consensus of those values. Figure 5.31 depicts a massive discrepancy among victim detection values, e.g., D1 and D2 are totally sure about the existence of a human victim, providing a victim detection level of 94.38 % and 89.4 %, respectively. In contrast, D4, D5, and D6 have moderately low victim detection levels, close to 43.05 %, 43.33 %, and 46.95 %. These discrepancies make clear the variability of quadrotors' sensing and the need to use consensus estimation with the intent to determine the victim's existence in the disaster area.

As explained previously, Figure 5.34 shows the variability of the quadrotors' measurements


(a) Total occlusion of the victim (0 % VDL)

(b) Partial occlusion of the victim (50 % VDL)

(c) Victim totally detected (100 % VDL)

**Figure** 5.29: Victim Detection.

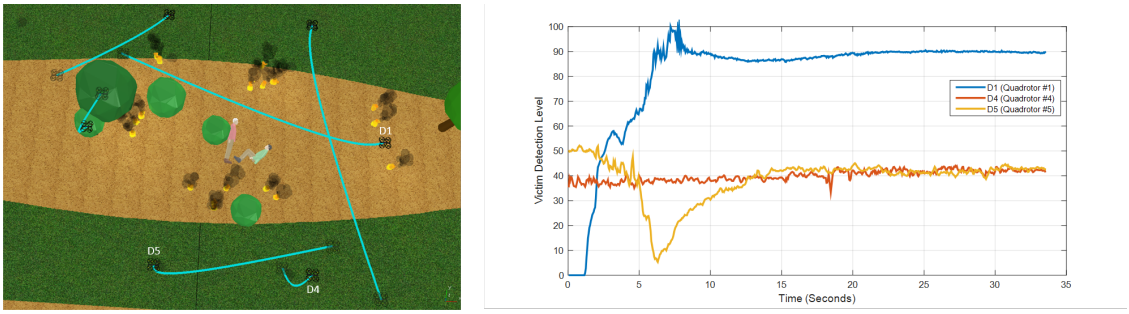



**Figure** 5.30: Digraph scheme.

(red area) versus the Consensus Estimation value (red line). This Figure shows how the consensus algorithm evaluates the different values thrown by the quadrotors, and it finds a consensus value among the other agents to determine the final level of victim detection. It should be taken into account that this victim detection value is achieved through partial information based on communication between some agents and their neighbors within the sub-swarm.

## 5.3. Cooperative Load Transportation

### 5.3.1. Geometric Control

Two simulations are carried out to demonstrate the improvements implemented, allowing us to compare the position errors in the quadrotors. First, we contemplate the switching dynamics but without the fourth term in (4.8), which is the geometric control developed in [55]. This algorithm needs predefined waypoints as inputs for the geometric control that takes the quadrotor to the desired position. Second, we perform the simulation that gathers all the concepts mentioned in the previous section. Finally, we show the improvements in

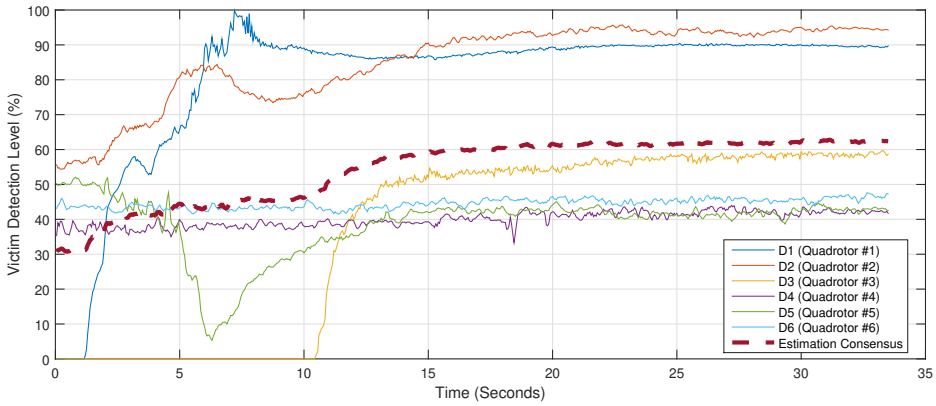



**Figure** 5.31: Consensus Estimation based on single Quadrotor victim detection.

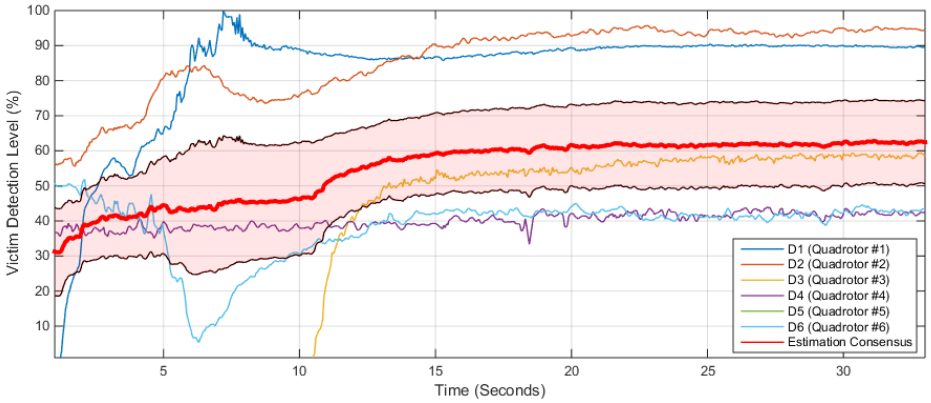



**Figure** 5.32: Consensus Estimation vs Quadrotors' sensing variability.

the position error compared to the one obtained in [55]. The algorithm can take random initial quadrotors' positions as long as they are physically possible. This means that robots cannot be far from the load since they are attached to the load. This is possible by applying the position-based passivity controller that takes the initial position of each quadrotor and gives the first group of waypoints. Those waypoints can minimize lifting efforts satisfying (4.4). In this particular case, the formation is a rhombus when using four quadrotors. Due to this shape, quadrotors minimize the efforts. Once the first desired position is achieved, the dynamics of the system change to the one that contemplates the mass in the model. In the same way, to lift the load to the desired altitude, the control must be modified considering this time the fourth term in (4.8) and bearing in mind that the body's mass is changing due to the load weight. Table 1 defines the simulation parameters needed.

To start analyzing the results, the two simulations previously mentioned are shown in a 3-D plot in Fig. 5.35, and Figure. 5.36, where dashed lines correspond to the trajectory performed

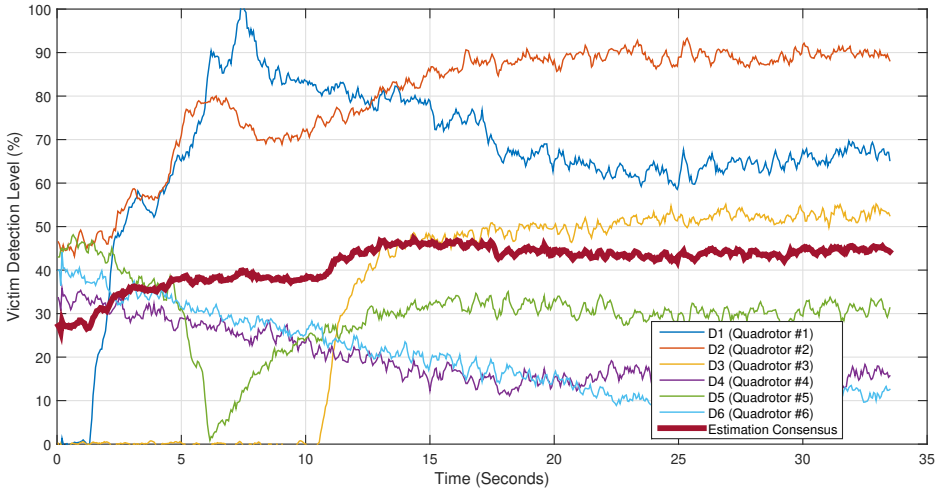



**Figure** 5.33: Drones estimation and Consensus Estimation in a fire environment.

by each quadrotor. Asterisks represent the quadrotors, and the solid dark gray lines are the cables attached to the brown square load. As expected, regardless of the controller implemented, both can take the load to the desired altitude. However, there are some changes, first simulation, which does not contemplate the improvements, presents translation changes in the x and y-axis suddenly and abruptly. On the other hand, the second simulation also presents changes in the x and y-axis, but they are shorter and smoother. The changes in the x and y-axis are due to the new force added when the cable is stretched to the limit, and the load starts to influence it. Furthermore, the position error is reduced by the improvements applied in the second simulation. It is worth mentioning that in the single quadrotor, lifting the load, the tension can be found by summation of forces in the entire system, giving a result

$$T = \frac{m_l}{m_q + m_l} F,$$

the forces are all on the z-axis because the quadrotor is over the load. In contrast, in the multi-quadrotor system, the effects of the forces depend on unit vectors $\mu_i$, and also the summation of forces changes to

$$\frac{T_1 + T_2 + T_3 + T_4}{m_l} = \frac{F_1 - T_1}{m_{q_1}} + \frac{F_2 - T_2}{m_{q_2}} + \frac{F_3 - T_3}{m_{q_3}} + \frac{F_4 - T_4}{m_{q_4}},$$

this is something expected, considering that the mass dynamic depends on the summation of all the tensions generated in each quadrotor. This new term in control shows that placing the robots aligned with its positional opposed quadrotor improves the performance of the task.

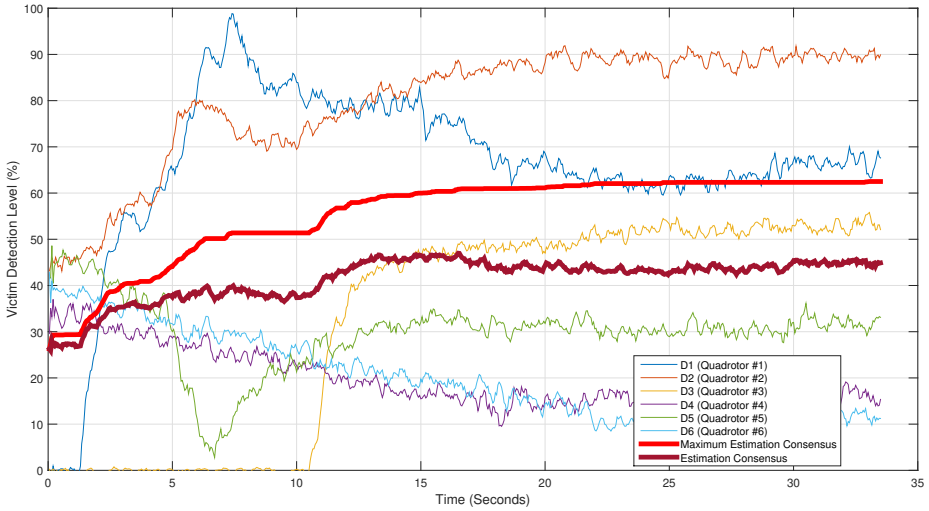



**Figure** 5.34: Drones estimation, Consensus Estimation, and Max Consensus estimation in a fire environment.

**Table** 5.4: Simulation Parameters

| **Parameter** | **Value** | **Parameter** | **Value** |
|---|---|---|---|
| $r_{q_1}(0)$ | [-0.75, 0.5, 0] | $r_{q_2}(0)$ | [0.75, 0.5, 0] |
| $r_{q_3}(0)$ | [0.75, -0.5, 0]$r_{q_4}(0)$ | [0.75, -0.5, 0] | |
| $v_{q_1}(0)$ | [0, 0, 0] | $v_{q_2}(0)$ | [0, 0, 0] |
| $v_{q_3}(0)$ | [0, 0, 0] | $r_{q_4}(0)$ | [0, 0, 0] |
| $J_1$ | $I_3$[11.7, 11.7, 23.4] | $m_{q_1}$ | 0.505 |
| $J_2$ | $I_3$[11.7, 11.7, 23.4] | $m_{q_2}$ | 0.505 |
| $J_3$ | $I_3$[11.7, 11.7, 23.4] | $m_{q_3}$ | 0.505 |
| $J_4$ | $I_3$[11.7, 11.7, 23.4] | $m_{q_4}$ | 0.505 |
| $l_1$ | 1.5 | $l_2$ | 1.5 |
| $l_3$ | 1.5 | $l_4$ | 1.5 |
| $k_p$ | $I_3$[1.58, 1.56, 1.73] | $t_f$ | 10 |
| $k_v$ | $I_3$[0.79, 0.79, 0.71] | $\alpha$ | 10 |
| $k_R$ | $I_3$[11, 11.7, 7.3] | $\beta$ | 10 |
| $k_\Omega$ | $I_3$[1.9, 1.7, 0.29] | $m_l$ | 0.1 |
| $k_g$ | $I_3$[0.1, 0.1, 0] | $\delta_i$ | $I_3$ |
| $v(t)$ | 0 | $k_i$ | 5 |

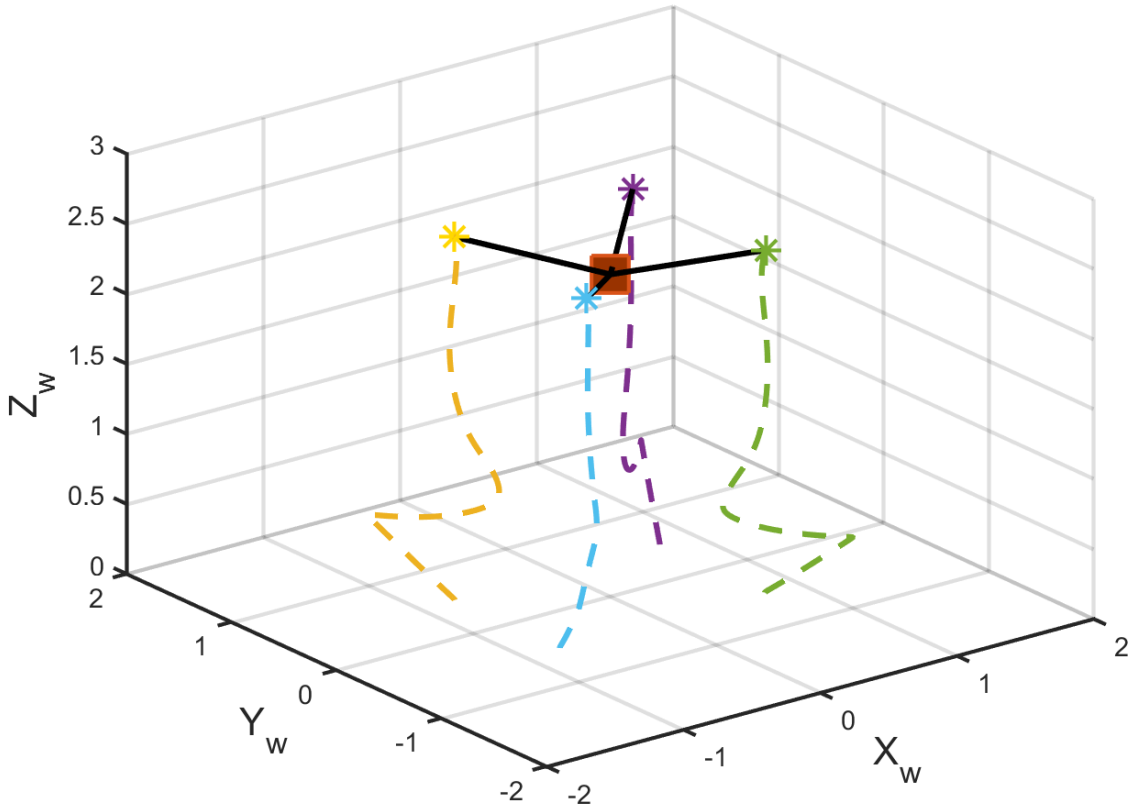



**Figure** 5.35: Cooperative transportation: Simulation using conventional geometric control.

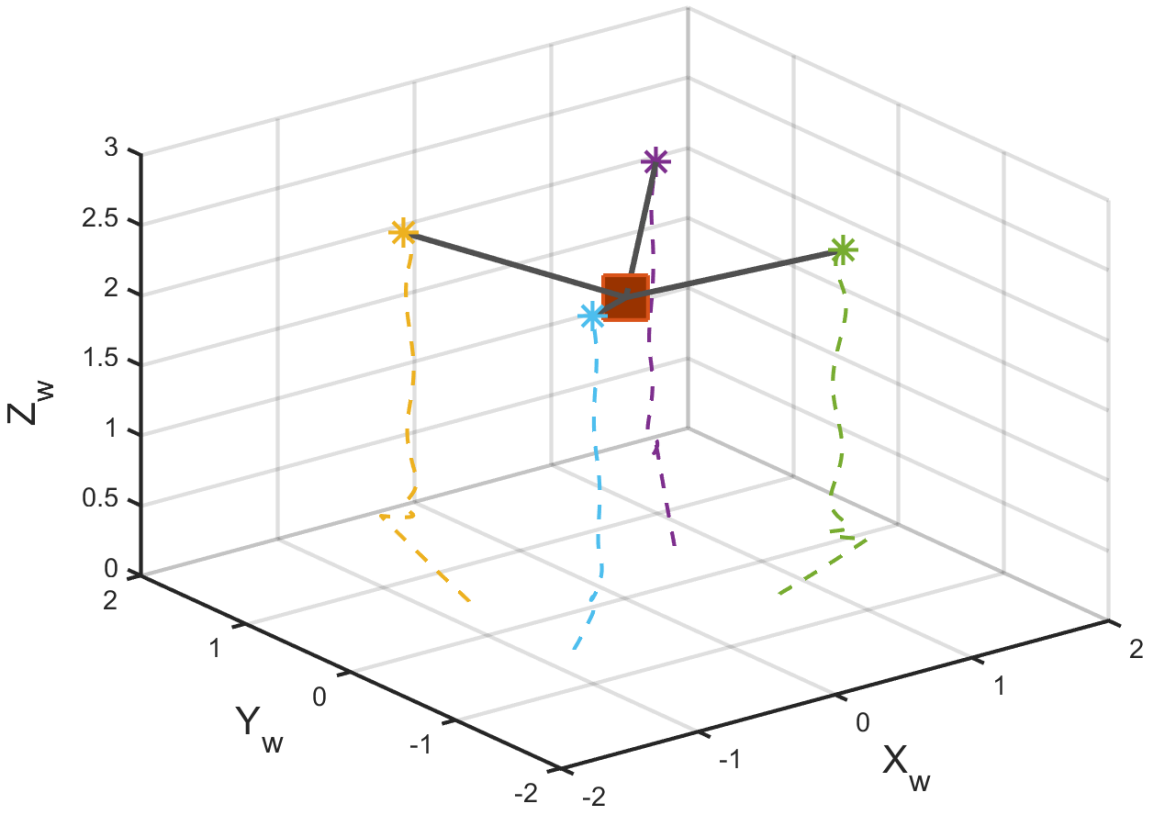



**Figure** 5.36: Cooperative transportation: Simulation considering improvements of positioning and minimization of effort.

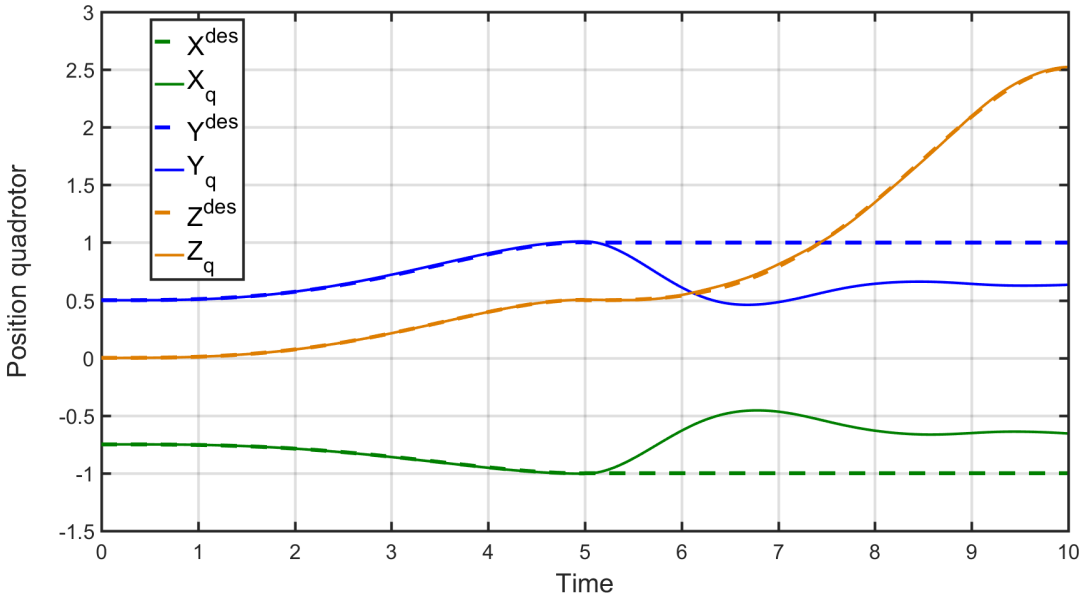



**Figure** 5.37: Cooperative transportation: Position evolution in x,y, and z using conventional geometric control.

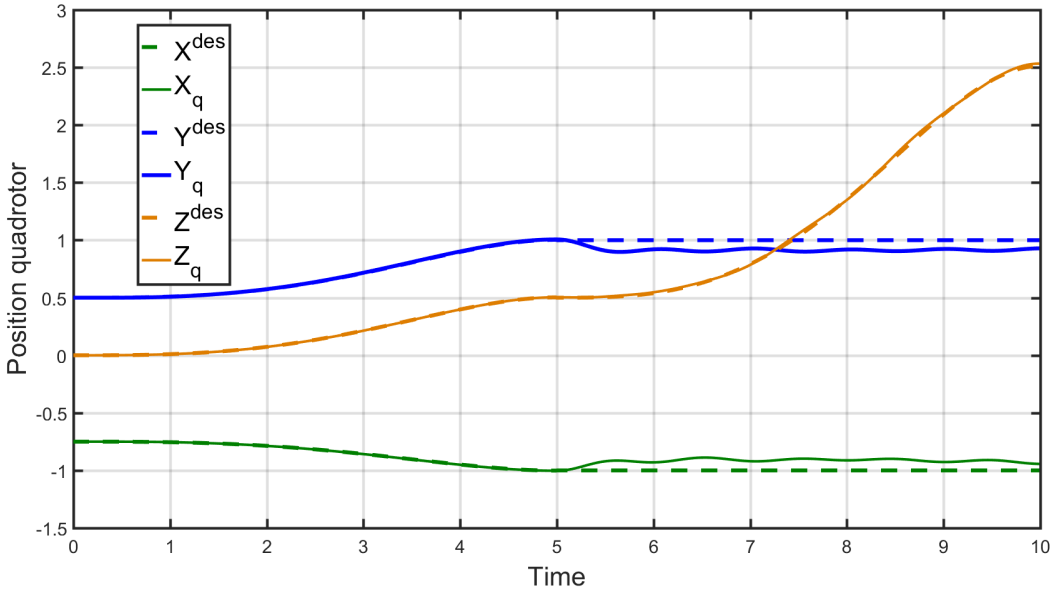



**Figure** 5.38: Cooperative transportation: Position evolution in x,y, and z considering improvements of positioning and minimization of effort.

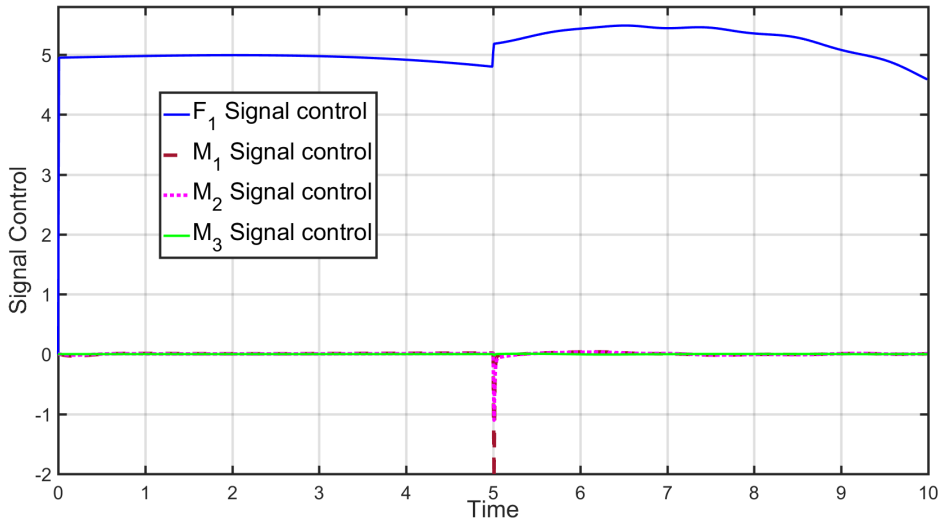



**Figure** 5.39: Cooperative transportation: Control Signals, using conventional geometric control.

As mentioned before, the x and y-axis are the most affected by the use of multiple robots to lift a suspended-cable load. The reason is that the forces are not just in the z-axis since the vectors going from quadrotors to mass $\mu_i$ have a point in each component. Figure 5.37 and Figure 5.38 show the evolution of each axis over time. Orange lines represent the z-axis, the x-axis with blue, and the y-axis with green lines. In the same way, dashed lines are the desired trajectory generated by solving the cost function, and solid lines are the evolution of the $r_{q_1}$ states. As shown in the second simulation, considering the improvements is way better than in the first simulation, where just the conventional geometric control is implemented. Regarding the z-axis, both satisfy the task, but just the modified controller can make the position error tend to zero. In the second simulation, quadrotors can correct the position in x and y faster than in the first simulation. This could be thanks to the modification in the control law as this term tries to maintain the alignment between the opposed quadrotors, thus minimizing the efforts and adding additional force to these axes in the controller.

Finally, Figure 5.39 and Figure 5.40 show the signal control applied to quadrotor 1 in both cases, first, without additional term in the geometric power and second with it, respectively. The solid blue line represents the $F$ output control, the solid green line is the first term of vector $M$ in the geometric control, the dashed magenta line is the second term, and the dashed red line is the third one. As can be seen, the $F$ signal changed in the second simulation with some oscillations in control caused by the additional term. In the same way, vector $M$ presents oscillations in the second and third terms, which are the signal produced to correct the position faster and avoid the trajectory separating in the initial lifting effort.

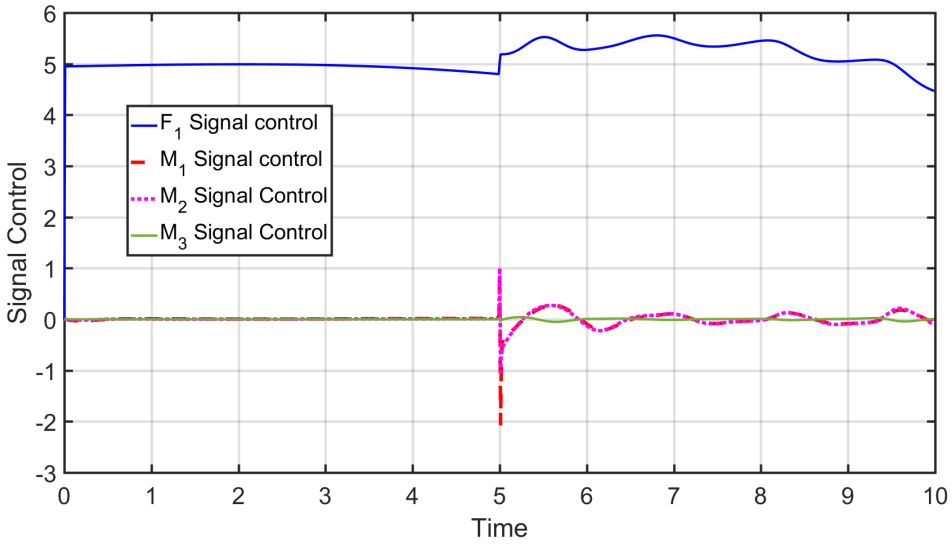



**Figure** 5.40: Cooperative transportation: Control Signals, considering improvements of positioning and minimization of effort.

## 5.3.2. Adaptive Control

A simulation is carried out to evaluate the algorithm's performance divided into two subsections: the DMRAC and the entire simulation.
To accomplish the simulation the following parameters are considered: $r_{q1}(0) = [-0{,}75, 0{,}5, 0]$, $r_{q2}(0) = [0{,}75, 0{,}5, 0]$, $r_{q3}(0) = [0{,}75, -0{,}5, 0]$, $r_{q4}(0) = [0{,}75, -0{,}5, 0]$, $v_{qi}(0) = [0, 0, 0]$, $J_i = I_3[11{,}7, 11{,}7, 23{,}4]$, $m_{qi} = \text{Rand}[0{,}3 - 0{,}7]$, $l_i = 1{,}5$, $k_p = I_3[1{,}58, 1{,}56, 1{,}73]$, $k_v = I_3[1{,}9, 0{,}79, 0{,}71]$, $k_R = I_3[11, 11{,}56, 7{,}73]$, $k_\omega = I_3[1{,}9, 1{,}7, 0{,}29]$, $k_\Omega = I_3[0{,}2, 0{,}2, 0]$, $m_l = 0{,}15$, $b_i = \text{Rand}[0{,}5 - 1]$, $k = \text{Rand}[0{,}5 - 1]$, $\gamma = 10$, $Q = 100 * I_2$, $f_i(x_i) = 0{,}1\sin(x_{2i})$, $\eta_i = 0{,}1\sin(t)$, $\delta_i = 2$ and all coupling vectors are initialized in zero.

### DMRAC with $\sigma$-Modification

The main objective of this control is to achieve consensus in the $z_W$ axis and synchronization in the $y_W$ and $x_W$ axis. Fig. 5.41 shows the consensus performed by the quadrotors in the $z_W$ axis. It is distinguishable the trajectory of each quadrotor and the virtual leader. Each quadrotor begins at a different value, and the consensus is performed to a desired altitude. At 4.5 seconds of the simulation, the consensus is satisfactory with a small bounded error. At the beginning of the simulation is noted that the quadrotors suffer trying to achieve the consensus. This can be caused by the gravity force, which is not contemplated in the virtual leader. Moreover, it is noted that the perturbation vanishes with time, having less and less effective as time continues.
Figure 5.42 shows the robot synchronization accomplished by quadrotors in the $x_W$ axis.

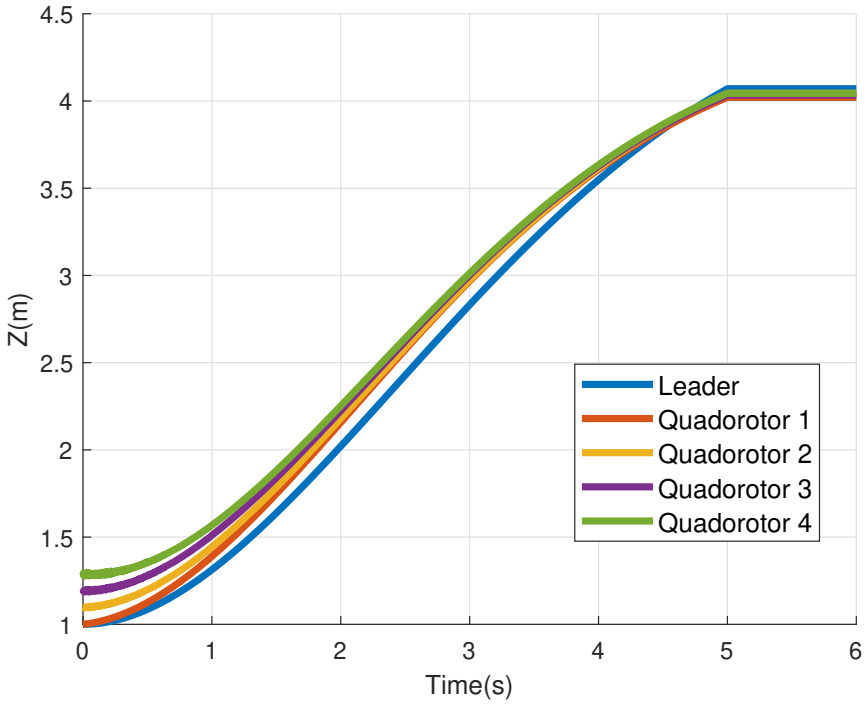



**Figure** 5.41: DMRAC in $z_W$ states synchronization of multiple quadrotors.

On the one hand, quadrotors 1 and 4 try to stick to the leader reference because they are aligned in the configuration. On the other hand, quadrotors 3 and 4 make synchronization with $\delta_i = 2$. This kind of synchronization is needed in the $x_W$ and $y_W$ axis, and in contrast to the one performed $z_W$ axis, sticking to the reference is rapidly achieved. In the case of $x_W$ and $y_W$ axis, the $F_i^x$ and $F_i^y$ are used just for the geometric control and are not contemplated by the quadrotor model.

### Full Simulation

To start analyzing the results, one simulation is shown in a 3-D plot and a top view plot, Fig. 5.43 and Figure 5.44, respectively, where dashed lines correspond to the trajectory performed by each quadrotor. Asterisks represent the quadrotors, and the solid dark gray lines are the cables attached to the brown square load. Figure 5.43 depicts the top view of the simulation to show the behavior of quadrotors in the $x_W$ and $y_W$ axis, which is interesting since it is noticeable how quadrotors synchronize trajectories with the quadrotor that is on the other side of the load right in the opposite place.

Finally, Fig. 5.44 shows the entire simulation in $x_W$, $y_W$, and $z_W$ axis. As can be seen, quadrotors maintain shape and take the load perfectly. Regardless of the lack of information from other quadrotors, behavior synchronization is achieved by the virtual leader. Even when each quadrotor has input and model time-varying uncertainties, the control can overcome

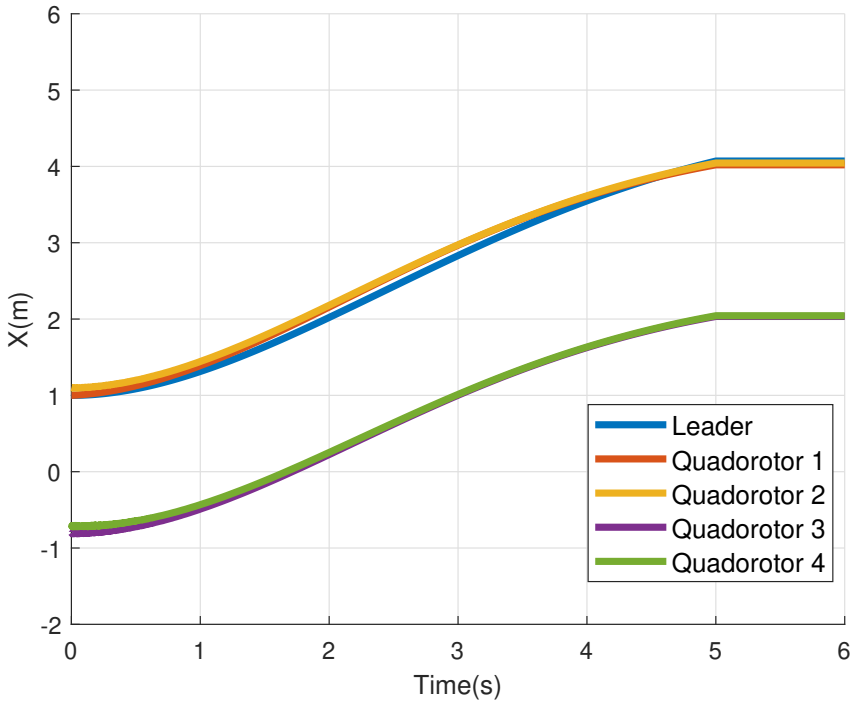



**Figure** 5.42: States synchronization in $x_W$ considering fixed $\delta$'s.

this and reach the desired behavior.

### 5.3.3. Bilevel Adaptive Control

To show the performance of the algorithm, we perform a numerical simulation in Matlab, and then in CoppeliaSim robotics simulator, the parameters used in both experiments are $\mathbf{J}_i = \mathbf{I}_3[11{,}7, 11{,}7, 23{,}4]10^{-3} kg \cdot m^2$, $\mathbf{I}_3$ is the identity matrix, $m_1 = 0{,}505\ kg$, $m_2 = 0{,}405\ kg$, $m_3 = 0{,}605\ kg$, $m_4 = 0{,}5\ kg$, $m_l = 0{,}5\ kg$, $l_i = 1{,}5\ m$, the proportional control matrices are $\mathbf{K}_R = diag[0{,}79, 0{,}79, 0{,}71]$, $\mathbf{K}_\omega = diag[0{,}19, 0{,}17, 0{,}29]$.

First, let us remember the type of graph we are using in this experiment, which is multi-layer. We use a directed graph to work on the communication system between robots and a non-directed, fully connected graph to capture the physical interaction from the cables. On the one hand, it can be seen in Fig. 5.45 the dashed blue and directed arrows represent how we establish the communication on how robots receive the trajectory reference. Quadrotor 1 is the only one with direct communication to the trajectory reference. Quadrotor 2 and quadrotor 3 receive the reference from quadrotor 1. Finally, quadrotor 4 gets the reference from quadrotor 2 and quadrotor 3.

On the other hand, solid yellow lines represent the edges that capture the physical interaction between the cables that are attached to the same point in the load. Notice that the movement from one of the quadrotors will affect the dynamics of the others since they are directly

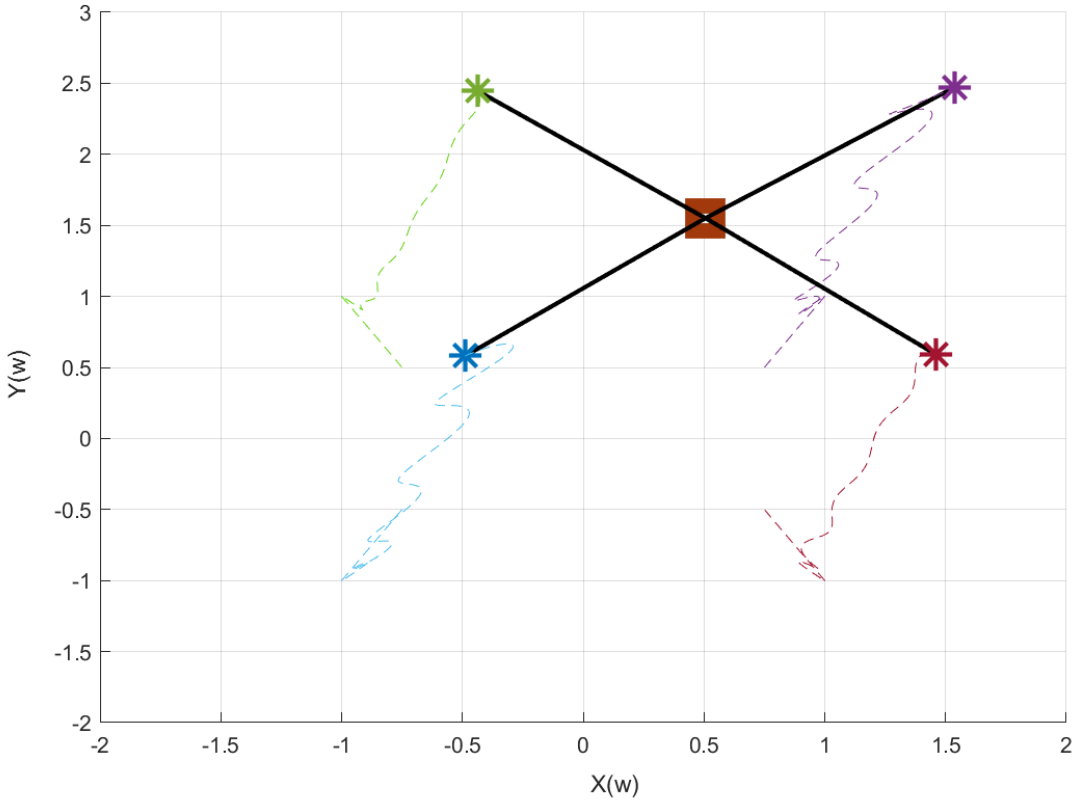



**Figure** 5.43: Full simulation top view.

connected by stretched cables. Even when they are following a trajectory due to delays or disturbances, the physical connection will directly affect the performance of each of the flight dynamics, which we handle by the adaptive distributed algorithm described in Section 4.4.4.

In Figure 5.46, it can be seen a constant reference trajectory in the $x - axis$. It is noticeable that the quadrotors exhibit a trajectory perturbation at the beginning. This is caused by the random initial location of the quadrotors, then they start to tighten the cable and generate a force to other quadrotors in the opposite direction. Once the algorithm adapts to the external perturbations, it converges to the reference. Notice that there are two clusters of signals, blue and red solid lines, which are the velocity and position of the quadrotors, respectively. It is worth mentioning that despite reaching the same position, we avoid collisions for those aligned in $y - axis$. We achieve this by setting a displacement in the reference signal $\mathbf{r}_b^d$ to each quadrotor $\mathbf{r}_i^d$.

To demonstrate that the algorithm can work with time-varying reference trajectories, we perform a second simulation that works with a sinusoidal reference trajectory shown in Figure 5.47. Here, even when the reference trajectory is time-varying, the system is not disturbed when the reference changes its tendency. Disturbances to following the trajectory are just

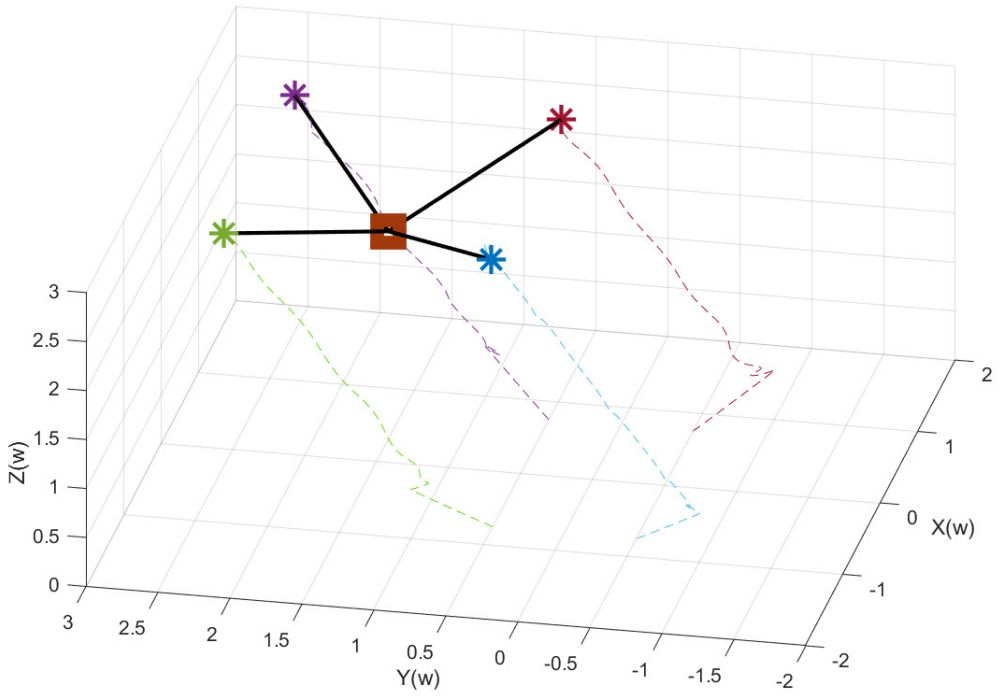



**Figure** 5.44: Cooperative transportation using multiple quadrotors simulation.

given in the beginning. Once the system stabilizes, the rest of the simulation quadrotors can follow the trajectory and remain in a bounded error.

## 5.4. Hybrid Systems

Here we consider hybrid systems, as introduced in Chapter 2.4, where we have the interaction and abstraction of the problem by considering continuous dynamics and discrete dynamics.

In the case of exploration and mapping, we can consider different states that track the progress of the satisfaction of a mission. Then, continuous dynamics is the motion that moves robots from their current location to their goals, and discrete dynamics defines whether the robot is tasked to visit a specific region. In the case of exploration and mapping, we can consider different states that track the progress of the satisfaction of a mission. Then, continuous dynamics is the motion that moves robots from their current location to their goals, and discrete dynamics defines whether the robot is tasked to visit a specific region.

For navigation and victim, detection can easily consider two different states, from navigating the environment to stopping and circulating the region whenever a victim is detected. All

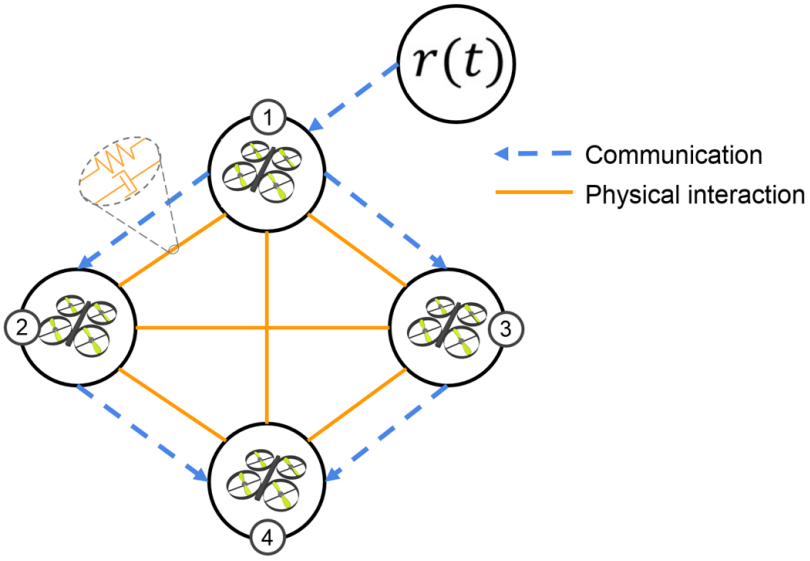



**Figure** 5.45: Multi-layer graph: Communication graph, and physical interaction graph.

movements in the environment are considered continuous dynamics. Note that the navigation also has inner states for going to the goal region and avoiding collision with other robots and obstacles.

When considering the quadrotors attached to a load using cables, notice that we can abstract the problem in three different states. First, when the distance between the quadrotors and the load is less than the length of the cables, here we can consider that quadrotors move freely in the environment. Second, when the distance between the quadrotors and the load is equal to the length of the cable here, quadrotors start to experience an external force that can lead to system instabilities. Lastly, quadrotors carry the load and must compensate for the force exerted by the load's weight and external disturbances, as other quadrotor forces and wind.

Finally, a Finite State Automaton could capture the whole mission where states are the exploration and mapping, navigation and victim detection, and load transportation allowing for high-level planning, coordinating when and where to satisfy every task, and tracking the progress of satisfaction of the mission.

### 5.4.1. High-Level Planning

In this section, we briefly describe the main concepts from automata theory and temporal logic, which are crucial for understanding the problem formulation and solution. We first describe the basics of Linear Temporal Logic (LTL) which helps describe logic specifications over an environment.

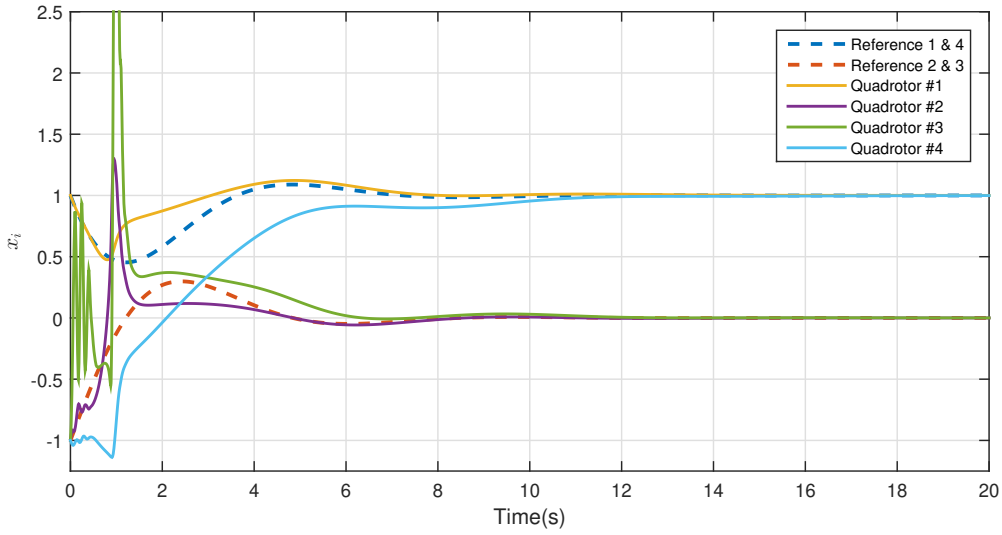



**Figure** 5.46: Constant trajectory reference for the load and response of the quadrotors in the $x-axis$.

**Definition 3 (LTL)** *An LTL formula $\phi$ over a set of properties is defined using standard Boolean operators: $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction), and temporal operators, $\bigcirc$ (next), $U$ (until), $\Diamond$ (eventually), and $\Box$ (always). The semantics of LTL formulae over $P$ is given concerning infinite words over $2^{AP}$, where $AP$ are atomic propositions.*

An LTL formulation is widely used to describe desired logical specifications for a system to satisfy. For instance, let us assume there is an environment with selected regions $R_1$, $R_2$, and an obstacle $O_1$, if there would be the necessity to describe or impose a particular behavior such as *"visit $R_1$ or $R_2$ infinitely many times while avoiding collisions with obstacles"* this could be encoded in an LTL form as $\Box\Diamond((R_1 \vee R_2) \wedge \neg O_1)$. For the semantics of LTL, please refer to [70].
When capturing the motion capability of a robot in the environment or configuration space, we make use of DTS (Deterministic Transition System), which is defined as follows

**Definition 4 (Transition System)** *A weighted transition system (TS) is a tuple $\mathcal{T} = (X, x_0^{\mathcal{T}}, \delta_{\mathcal{T}}, AP, h, w_{\mathcal{T}})$, where:*

- *$X$ is a finite set of states;*
- *$x_0^{\mathcal{T}} \in X$ is the initial state;*
- *$\delta_{\mathcal{T}} \subseteq X \times X$ is a set of transitions;*
- *$AP$ is a set of properties (atomic propositions);*
- *$h : X \to 2^{AP}$ is a labeling function;*

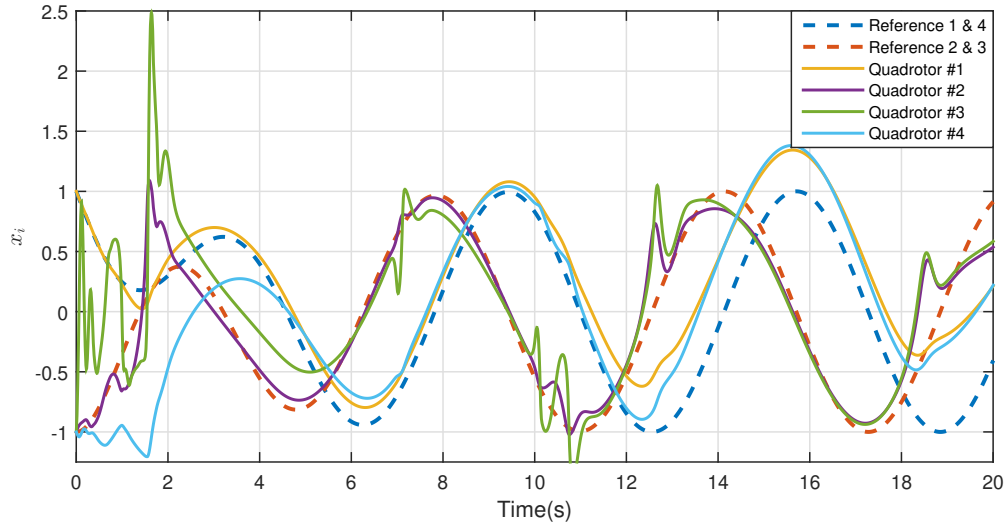



**Figure** 5.47: Sinusoidal trajectory reference for the load and response of the quadrotors in the $x - axis$.

- $w_{\mathcal{T}} : \delta_{\mathcal{T}} \to \mathbb{Z}_{>0}$ *is a weight function.*

We use the definition of a transition as stated in [6] as $(x, x') \in \triangle$ by $x \to x'$. Also, a trajectory is an infinite sequence of states that belongs to the transition system $\mathbf{x} = x_0 x_1 ...$ such that $x_k \to x_{k+1}$, $\forall\, k \geq 0$.
Even though the LTL formulation can capture the logic desired for the system, it is not easy to mix it with the transition system that describes the motion in the configuration space to get a solution. One of the better solutions comes from translating the LTL formulation into an automaton. Different automatons can be used, but for this project, the one that better encapsulates the LTL specification is the Büchi Automaton, which is defined as follows

**Definition 5 (Büchi Automaton)** *A (non-deterministic) Büchi Automaton is a tuple* $\mathcal{B} = (S_{\mathcal{B}}, S_{\mathcal{B},}, \Sigma, \delta, F_{\mathcal{B}})$ *where:*

- $S_{\mathcal{B}}$ *is a finite set of states;*
- $S_{\mathcal{B},} \subseteq S_{\mathcal{B}}$ *is the set of initial states;*
- $\Sigma$ *is the alphabet of inputs;*
- $\delta : S_{\mathcal{B}} \times \Sigma \to 2^{S_B}$ *is the transition function;*
- $F_{\mathcal{B}} \subseteq S_{\mathcal{B}}$ *is the set of final or accepting states.*

In the same way as in the transition system, a transition $(s, s') \in \delta(s, \sigma)$ is defined as $s \to_{\mathcal{B}} s'$. A trajectory of the Bučhi automaton $(s_0 s_1 ...)$ is generated by an infinite sequence of symbols

$(\sigma_0\sigma_1...)$ if $(s0 \in S_{\mathcal{B}_0}$ and $s_k \rightarrow_{\mathcal{B}} s_{k+1}\ \forall\, k \geq 0$. An infinite input sequence over $\Sigma$ is said to be accepted by a Bučhi automaton $\mathcal{B}$ if it generates at least one trajectory of $\mathcal{B}$ that intersects the set $F_{\mathcal{B}}$ infinitely many times.
The way to ensure that an agent satisfies an LTL specification while considering the transition system, which describes the way the robot can move in the environment, is through a product automaton between the transition system $\mathcal{T}$ and the Büchi Automaton $\mathcal{B}$ which is defined below

**Definition 6 (Product Automaton)** *Given a transition system $\mathcal{T} = (X, x_0^{\mathcal{T}}, \delta_{\mathcal{T}}, AP, h, w_{\mathcal{T}})$ and a Büchi automaton $\mathcal{B} = (S_{\mathcal{B}}, s^{\mathcal{B}_0}, 2^{AP}, \delta_{\mathcal{B}}, F_{\mathcal{B}})$, the product automaton (PA) $\mathcal{P} = \mathcal{T} \times \mathcal{B}$ is the tuple $\mathcal{P} = (S_{\mathcal{P}}, S_{\mathcal{P}_0}, \delta_{\mathcal{P}}, \omega_{\mathcal{P}}, F_{\mathcal{P}})$, where*

- $S_{\mathcal{P}_0} = \{x_o\} \times S_{\mathcal{B}}$ *is the set of initial states;*
- $S_{\mathcal{P}} \subseteq X \times S_{\mathcal{B}}$ *is a set of states which are reachable from some initial state;*
- $\delta_{\mathcal{P}} \subseteq S_{\mathcal{P}} \times S_{\mathcal{P}}$ *is the set of transitions, defined by* $((x, s), (x', s')) \in \delta_{\mathcal{P}}$ *iff* $x \rightarrow_{\mathcal{T}} x'$ *and* $s \rightarrow_{\mathcal{B}} s'$*;*
- $\omega_{\mathcal{P}} = \delta_{\mathcal{B}} \rightarrow \mathbb{R}^+$ *is inherited from* $\mathcal{T}$ *such that* $\omega_{\mathcal{P}}((x, s)(x', s')) = \omega((x, x'))$*;*
- $F_{\mathcal{P}} = (X \times F_{\mathcal{B}}) \cap S_{\mathcal{P}}$ *is the set of accepting states of* $\mathcal{P}$*.*

Finally, the solution comes from using algorithms such as Dijkstra's algorithm to find an optimal path in the product automaton and projecting it back to the environment. Such a solution can be computed using off-the-shelf tools such as Optimal Multi-Agent Planner (LOMAP) [50].

# 6 Conclusions and Future Work

In this thesis, we have explored the application of bioinspired approaches to enhance the capabilities of multirobot systems in search and rescue operations. Specifically, we investigated the bioinspired behavior of ants and leveraged their exploration, navigation, load transportation, and coordination strategies to develop advanced algorithms for various aspects of search and rescue missions. The thesis comprised four chapters addressing a different aspect of bioinspired multirobot systems. The first chapter focused on exploration and mapping using multirobot systems.

We employed potential functions, Voronoi tessellation, and Q-learning to enable efficient exploration and mapping of the search area. Potential functions effectively guided the robots through the environment while avoiding obstacles and maximizing coverage. Voronoi tessellation facilitated partitioning the search area among the robots, enabling them to explore different regions simultaneously. Finally, Q-learning, a reinforcement learning technique, allowed the robots to learn and adapt their exploration strategies based on the feedback received during the search process. Through these approaches, we achieved improved exploration efficiency and accurate mapping of the environment.

The second chapter delved into swarm navigation and victim detection. We utilized potential functions for swarm navigation, enabling the robots to move in a coordinated manner while avoiding collisions and efficiently exploring the search area. Neural networks were employed for victim detection, leveraging their ability to recognize and classify victims based on visual cues and sensor data. Furthermore, we implemented a consensus approach to generate distributed consensus among the swarm robots regarding the presence of a victim. This approach allowed for efficient victim detection and improved decision-making within the swarm. Combining these techniques, we developed a robust and effective system for swarm navigation and victim detection in search and rescue scenarios.

In the third chapter, we focused on cooperative load transportation using quadrotors. We employed geometric and adaptive control techniques to enable the quadrotors to carry and transport heavy loads attached to cables. Geometric control precisely controlled the quadrotors' positions and orientations during load transportation, ensuring stability and distribution. Adaptive control techniques were utilized to handle uncertainties and changes in the load characteristics, enhancing the robustness and adaptability of the system. Through these approaches, we developed a cooperative load transportation system that expanded the capabilities of quadrotors in handling heavy payloads, thus contributing to the effectiveness of search and rescue operations.

Lastly, in the fourth chapter, we showed case studies for every previous chapter and introduced automata theory for high-level planning of multirobot systems in search and rescue operations. Automata theory allowed for the specification and synthesis of complex behaviors and coordination strategies within the multirobot system. By formalizing the desired high-level behaviors and constraints, we generated plans and strategies that ensured efficient coordination, task allocation, and resource optimization within the system.
Overall, this thesis demonstrated the potential of bioinspired algorithms and strategies for enhancing the capabilities of multirobot systems in search and rescue operations. Furthermore, the findings and insights gained from this work have the potential to significantly enhance the efficiency, adaptability, and coordination of multirobot systems in critical search and rescue operations.

## 6.1. Future Work

By continuing to build upon the foundations laid in this thesis, we can further advance the field and contribute to developing innovative solutions for saving lives and mitigating the impact of disasters.
Future research directions involve

- Further refining and optimizing the proposed algorithms.
- Conducting extensive real-world experiments.
- Exploring additional bioinspired techniques to tackle the challenges in search and rescue missions.

We have already extended the work on this thesis in the following papers. We explored a distributed swarm particle optimization approach presented in [67] for swarm navigation.
For quadrotors connected by a rope manipulating and transporting objects as presented in [36], [14], [15], and [35].
For the high-level planning of algorithms using temporal logic and optimization approaches presented in [19], [17], and [16] and a partial satisfaction approach in case the mission cannot be fully satisfied presented in [21] and [22].

# References


[1] Arevalo-Castiblanco, M ; Calderon, J: Robust adaptive synchronization of interconnected heterogeneous quadrotors transporting a cable-suspended load. En: *2021 IEEE International Conference on Robotics and Automation (ICRA)* IEEE, 2021, p. 31–37

[2] Aulinas, Josep ; Petillot, Yvan ; Salvi, Joaquim ; Lladó, Xavier: The SLAM problem: a survey. En: *Artificial Intelligence Research and Development* (2008), p. 363–371

[3] Bai, He ; Arcak, Murat ; Wen, John: *Cooperative control design: a systematic, passivity-based approach.* Springer Science & Business Media, 2011

[4] Baldi, Simone ; Frasca, Paolo: Adaptive synchronization of unknown heterogeneous agents: An adaptive virtual model reference approach. En: *Journal of the Franklin Institute* 356 (2019), Nr. 2, p. 935–955

[5] Bell, John E. ; McMullen, Patrick R.: Ant colony optimization techniques for the vehicle routing problem. En: *Advanced engineering informatics* 18 (2004), Nr. 1, p. 41–48

[6] Belta, Calin ; Yordanov, Boyan ; Gol, Ebru A.: *Formal methods for discrete-time dynamical systems.* Vol. 15. Springer, 2017

[7] Bernard, Markus ; Kondak, Konstantin ; Maza, Ivan ; Ollero, Anibal: Autonomous transportation and deployment with aerial robots for search and rescue missions. En: *Journal of Field Robotics* 28 (2011), Nr. 6, p. 914–931

[8] Bhatti, Shehroze ; Desmaison, Alban ; Miksik, Ondrej ; Nardelli, Nantas ; Siddharth, N ; Torr, Philip H.: Playing doom with slam-augmented deep reinforcement learning. En: *arXiv preprint arXiv:1612.00380* (2016)

[9] Bruemmer, David J. ; Dudenhoeffer, Donald D. ; McKay, Mark D. ; Anderson, Matthew O.: A robotic swarm for spill finding and perimeter formation / IDAHO NATIONAL ENGINEERING AND ENVIRONMENTAL LAB IDAHO FALLS. 2002. – Informe de Investigación

[10] Bullo, Francesco ; Cortes, Jorge ; Martinez, Sonia: *Distributed control of robotic networks: a mathematical approach to motion coordination algorithms*. Princeton University Press, 2009

[11] Cardona, GA ; Bravo, C ; Quesada, W ; Ruiz, D ; Obeng, M ; Wu, X ; Calderon, JM: Autonomous Navigation for Exploration of Unknown Environments and Collision Avoidance in Mobile Robots Using Reinforcement Learning. En: *2019 SoutheastCon* IEEE, 2019, p. 1–7

[12] Cardona, GA ; Tellez-Castro, D ; Mojica-Nava, E: Cooperative transportation of a cable-suspended load by multiple quadrotors. En: *IFAC-PapersOnLine* 52 (2019), Nr. 20, p. 145–150

[13] Cardona, Gustavo A. ; Calderon, Juan M.: Robot swarm navigation and victim detection using rendezvous consensus in search and rescue operations. En: *Applied Sciences* 9 (2019), Nr. 8, p. 1702

[14] Cardona, Gustavo A. ; D'Antonio, Diego S. ; Fierro, Rafael ; Saldana, David: Adaptive control for cooperative aerial transportation using catenary robots. En: *2021 Aerial Robotic Systems Physically Interacting with the Environment (AIRPHARO)* IEEE, 2021, p. 1–8

[15] Cardona, Gustavo A. ; D'Antonio, Diego S. ; Vasile, Cristian-Ioan ; Saldaña, David: Non-prehensile manipulation of cuboid objects using a catenary robot. En: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* IEEE, 2021, p. 5270–5275

[16] Cardona, Gustavo A. ; Kamale, Disha ; Vasile, Cristian-Ioan: Mixed Integer Linear Programming Approach for Control Synthesis with Weighted Signal Temporal Logic. En: *Proceedings of the 26th ACM International Conference on Hybrid Systems: Computation and Control*, 2023, p. 1–12

[17] Cardona, Gustavo A. ; Leahy, Kevin ; Vasile, Cristian-Ioan: Temporal Logic Swarm Control with Splitting and Merging. En: *2023 IEEE International Conference on Robotics and Automation (ICRA)* IEEE, 2023, p. 12423–12429

[18] Cardona, Gustavo A. ; Ramirez-Rugeles, Juan ; Mojica-Nava, Eduardo ; Calderon, Juan M.: Visual victim detection and quadrotor-swarm coordination control in search and rescue environment. En: *International Journal of Electrical and Computer Engineering* 11 (2021), Nr. 3, p. 2079

[19] Cardona, Gustavo A. ; Saldaña, David ; Vasile, Cristian-Ioan: Planning for Modular Aerial Robotic Tools with Temporal Logic Constraints. En: *2022 IEEE 61st Conference on Decision and Control (CDC)* IEEE, 2022, p. 2878–2883

[20] Cardona, Gustavo A. ; Tellez-Castro, Duvan ; Calderon, J ; Mojica-Nava, Eduardo: Adaptive Multi-Quadrotor Control for Cooperative Transportation of a Cable-Suspended Load. En: *2021 European Control Conference (ECC)* IEEE, 2021, p. 696–701

[21] Cardona, Gustavo A. ; Vasile, Cristian-Ioan: Partial Satisfaction of Signal Temporal Logic Specifications for Coordination of Multi-robot Systems. En: *Algorithmic Foundations of Robotics XV: Proceedings of the Fifteenth Workshop on the Algorithmic Foundations of Robotics* Springer, 2022, p. 223–238

[22] Cardona, Gustavo A. ; Vasile, Cristian-Ioan: Preferences on Partial Satisfaction using Weighted Signal Temporal Logic Specifications. En: *2023 European Control Conference (ECC)* IEEE, 2023, p. 1–6

[23] Cardona, Gustavo A. ; Yanguas-Rojas, David ; Arevalo-Castiblanco, Miguel F. ; Mojica-Nava, Eduardo: Ant-Based Multi-Robot Exploration in Non-Convex Space without Global-Connectivity Constraints. En: *2019 18th European Control Conference (ECC)* IEEE, 2019, p. 2065–2070

[24] Chandrasekhar, Arjun ; Gordon, Deborah M. ; Navlakha, Saket: A distributed algorithm to maintain and repair the trail networks of arboreal ants. En: *Scientific reports* 8 (2018), Nr. 1, p. 1–19

[25] Cook, Zoe ; Franks, Daniel W. ; Robinson, Elva J.: Exploration versus exploitation in polydomous ant colonies. En: *Journal of theoretical biology* 323 (2013), p. 49–56

[26] Countryman, Stefanie M. ; Stumpe, Martin C. ; Crow, Sam P. ; Adler, Frederick R. ; Greene, Michael J. ; Vonshak, Merav ; Gordon, Deborah M.: Collective search by ants in microgravity. En: *Frontiers in Ecology and Evolution* 3 (2015), p. 25

[27] Cronin, Adam L.: Conditional use of social and private information guides house-hunting ants. En: *PloS one* 8 (2013), Nr. 5

[28] Cronin, Adam L.: Ratio-dependent quantity discrimination in quorum sensing ants. En: *Animal cognition* 17 (2014), Nr. 6, p. 1261–1268

[29] Cruz, Patricio J. ; Fierro, Rafael: Cable-suspended load lifting by a quadrotor UAV: hybrid model, trajectory generation, and control. En: *Autonomous Robots* 41 (2017), Nr. 8, p. 1629–1643

[30] Davids, Angela: Urban search and rescue robots: from tragedy to technology. En: *IEEE Intelligent systems* 17 (2002), Nr. 2, p. 81–83

[31] Davidson, Jacob D. ; Arauco-Aliaga, Roxana P. ; Crow, Sam ; Gordon, Deborah M. ; Goldman, Mark S.: Effect of interactions between harvester ants on forager decisions. En: *Frontiers in ecology and evolution* 4 (2016), p. 115

[32] Debout, Gabriel ; Schatz, Bertrand ; Elias, Marianne ; Mckey, Doyle: Polydomy in ants: what we know, what we think we know, and what remains to be done. En: *Biological Journal of the Linnean Society* 90 (2007), Nr. 2, p. 319–348

[33] Denker, Ahmet ; İşeri, Mehmet C.: Design and implementation of a semi-autonomous mobile search and rescue robot: SALVOR. En: *2017 International Artificial Intelligence and Data Processing Symposium (IDAP)* IEEE, 2017, p. 1–6

[34] Ducatelle, Frederick ; Di Caro, Gianni A. ; Pinciroli, Carlo ; Mondada, Francesco ; Gambardella, Luca: Communication assisted navigation in robotic swarms: self-organization and cooperation. En: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems* IEEE, 2011, p. 4981–4988

[35] D'antonio, Diego S. ; Cardona, Gustavo A. ; Saldana, David: The catenary robot: Design and control of a cable propelled by two quadrotors. En: *IEEE Robotics and Automation Letters* 6 (2021), Nr. 2, p. 3857–3863

[36] D'Antonio, Diego S. ; Xu, Jiawei ; Cardona, Gustavo A. ; Saldaña, David: Customizable-ModQuad: a Versatile Hardware-Software Platform to Develop Lightweight and Low-cost Aerial Vehicles.

[37] Elibol, Ercan ; Calderon, Juan ; Llofriu, Martin ; Moreno, Wilfrido ; Weitzenfeld, Alfredo: Analyzing and Reducing Energy Usage in a Humanoid Robot During Standing Up and Sitting Down Tasks. En: *International Journal of Humanoid Robotics* 13 (2016), Nr. 04, p. 1650014

[38] Elibol, Ercan ; Calderon, Juan ; Llofriu, Martin ; Quintero, Carlos ; Moreno, Wilfrido ; Weitzenfeld, Alfredo: Power usage reduction of humanoid standing process using q-learning. En: *Robot Soccer World Cup* Springer, 2015, p. 251–263

[39] Freas, Cody A. ; Schultheiss, Patrick: How to navigate in different environments and situations: lessons from ants. En: *Frontiers in psychology* 9 (2018), p. 841

[40] Fujisawa, Shoichiro ; Kurozumi, Ryota ; Yamamoto, Toru ; Suita, Yoshikazu: Path planning for mobile robots using an improved reinforcement learning scheme. En: *Proceedings of the IEEE Internatinal Symposium on Intelligent Control* IEEE, 2002, p. 67–74

[41] García, Pedro ; Gómez, A: ANTS framework for cooperative work environments. En: *IEEE Computer* 36 (2003), Nr. 3, p. 56–62

[42] Giraldeau, Luc-Alain ; Caraco, Thomas: *Social foraging theory*. Princeton University Press, 2018

[43] Goebel, Rafal ; Sanfelice, Ricardo G. ; Teel, Andrew R.: Hybrid dynamical systems. En: *IEEE control systems magazine* 29 (2009), Nr. 2, p. 28–93

[44] Gorbatsevich, Vladimir V. ; Vinberg, Ernest B.: *Lie Groups and Lie Algebras I: Foundations of Lie Theory Lie Transformation Groups*. Vol. 20. Springer Science & Business Media, 1996

[45] Gordon, Deborah M.: The dynamics of foraging trails in the tropical arboreal ant Cephalotes goniodontus. En: *PLoS One* 7 (2012), Nr. 11

[46] Gordon, Deborah M.: The evolution of the algorithms for collective behavior. En: *Cell systems* 3 (2016), Nr. 6, p. 514–520

[47] Haumann, Dominik ; Willert, Volker ; Listmann, Kim D.: DisCoverage: from coverage to distributed multi-robot exploration. En: *IFAC Proceedings Volumes* 46 (2013), Nr. 27, p. 328–335

[48] Ioannou, Petros ; Fidan, Bariş: *Adaptive control tutorial*. SIAM, 2006

[49] Jurdjevic, Velimir: *Geometric control theory*. Cambridge university press, 1997

[50] Kamale, Disha ; Karyofylli, Eleni ; Vasile, Cristian-Ioan: Automata-based optimal planning with relaxed specifications. En: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* IEEE, 2021, p. 6525–6530

[51] Kotaru, Prasanth ; Wu, Guofan ; Sreenath, Koushil: Dynamics and control of a quadrotor with a payload suspended through an elastic cable. En: *2017 american control conference (ACC)* IEEE, 2017, p. 3906–3913

[52] Kumar, Vijay R. ; Waldron, Kenneth J.: Force distribution in closed kinematic chains. En: *IEEE Journal on Robotics and Automation* 4 (1988), Nr. 6, p. 657–664

[53] Latty, Tanya ; Ramsch, Kai ; Ito, Kentaro ; Nakagaki, Toshiyuki ; Sumpter, David J. ; Middendorf, Martin ; Beekman, Madeleine: Structure and formation of ant transportation networks. En: *Journal of The Royal Society Interface* 8 (2011), Nr. 62, p. 1298–1306

[54] LeCun, Yann ; Bottou, Léon ; Bengio, Yoshua ; Haffner, Patrick: Gradient-based learning applied to document recognition. En: *Proceedings of the IEEE* 86 (1998), Nr. 11, p. 2278–2324

[55] Lee, Taeyoung ; Leok, Melvin ; McClamroch, N H.: Geometric tracking control of a quadrotor UAV on SE (3). En: *49th IEEE conference on decision and control (CDC)* IEEE, 2010, p. 5420–5425

[56] León, Jose ; Cardona, Gustavo A. ; Botello, Andres ; Calderón, Juan M.: Robot swarms theory applicable to seek and rescue operation. En: *Intelligent Systems Design and Applications: 16th International Conference on Intelligent Systems Design and Applications (ISDA 2016) held in Porto, Portugal, December 16-18, 2016* Springer, 2017, p. 1061–1070

[57] Li, Caihong ; Zhang, Jingyuan ; Li, Yibin: Application of artificial neural network based on q-learning for mobile robot path planning. En: *2006 IEEE International Conference on Information Acquisition* IEEE, 2006, p. 978–982

[58] Liu, Yugang ; Nejat, Goldie: Robotic urban search and rescue: A survey from the control perspective. En: *Journal of Intelligent & Robotic Systems* 72 (2013), p. 147–165

[59] Loianno, Giuseppe ; Kumar, Vijay: Cooperative transportation using small quadrotors using monocular vision and inertial sensing. En: *IEEE Robotics and Automation Letters* 3 (2017), Nr. 2, p. 680–687

[60] Mahony, Robert ; Kumar, Vijay ; Corke, Peter: Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. En: *IEEE robotics & automation magazine* 19 (2012), Nr. 3, p. 20–32

[61] McCreery, Helen F. ; Breed, MD: Cooperative transport in ants: a review of proximate mechanisms. En: *Insectes sociaux* 61 (2014), p. 99–110

[62] Mellinger, Daniel ; Kumar, Vijay: Minimum snap trajectory generation and control for quadrotors. En: *2011 IEEE International Conference on Robotics and Automation* IEEE, 2011, p. 2520–2525

[63] Mesbahi, Mehran ; Egerstedt, Magnus: *Graph theoretic methods in multiagent networks*. Princeton University Press, 2010

[64] Messina, Elena ; Jacoff, Adam: Performance standards for urban search and rescue robots. En: *Unmanned Systems Technology VIII* Vol. 6230 SPIE, 2006, p. 639–650

[65] Murphy, Robin R. ; Burke, Jennifer L.: Up from the rubble: Lessons learned about HRI from search and rescue. En: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* Vol. 49 SAGE Publications Sage CA: Los Angeles, CA, 2005, p. 437–441

[66] Nguyen, Nhan: *Model-Reference Adaptive Control. A Primer*. Springer, March 2018. – 123 p.

[67] Paez, David ; Romero, Juan P. ; Noriega, Brian ; Cardona, Gustavo A. ; Calderon, Juan M.: Distributed particle swarm optimization for multi-robot system in search and rescue operations. En: *IFAC-PapersOnLine* 54 (2021), Nr. 4, p. 1–6

[68] Prabhakar, Balaji ; Dektar, Katherine N. ; Gordon, Deborah M.: The regulation of ant colony foraging activity without spatial information. En: *PLoS computational biology* 8 (2012), Nr. 8

[69] Rizvi, Syed Muhammad A. ; Ahmed, Rameen M. ; Alamdar, Khawaja G. ; Khorasani, Mehdi R. ; Memon, Junaid A.: Human Detection and Localization in Indoor Disaster Environments Using UAVs. En: *2022 4th International Conference on Robotics and Computer Vision (ICRCV)* IEEE, 2022, p. 159–163

[70] Sadraddini, Sadra ; Belta, Calin: Formal guarantees in data-driven model identification and control synthesis. En: *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (part of CPS Week)*, 2018, p. 147–156

[71] Siegwart, Roland ; Nourbakhsh, Illah R. ; Scaramuzza, Davide: *Introduction to autonomous mobile robots*. 2011

[72] Sreenath, Koushil ; Kumar, Vijay: Dynamics, control and planning for cooperative manipulation of payloads suspended by cables from multiple quadrotor robots. En: *rn* 1 (2013), Nr. r2, p. r3

[73] Stephens, David W.: Decision ecology: foraging and the ecology of animal decision making. En: *Cognitive, Affective, & Behavioral Neuroscience* 8 (2008), Nr. 4, p. 475–484

[74] Stormont, Daniel P.: Autonomous rescue robot swarms for first responders. En: *CIHSPS 2005. Proceedings of the 2005 IEEE International Conference on Computational Intelligence for Homeland Security and Personal Safety, 2005.* IEEE, 2005, p. 151–157

[75] Thrun, Sebastian ; Liu, Yufeng: Multi-robot SLAM with sparse extended information filers. En: *Robotics Research. The Eleventh International Symposium: With 303 Figures* Springer, 2005, p. 254–266

[76] Turgut, Ali E. ; Çelikkanat, Hande ; Gökçe, Fatih ; Şahin, Erol: Self-organized flocking in mobile robot swarms. En: *Swarm Intelligence* 2 (2008), p. 97–120

[77] Villa, Daniel K. ; Brandão, Alexandre S. ; Sarcinelli-Filho, Mário: Load transportation using quadrotors: A survey of experimental results. En: *2018 International conference on unmanned aircraft systems (ICUAS)* IEEE, 2018, p. 84–93

[78] Villa, Daniel K. ; Brandao, Alexandre S. ; Sarcinelli-Filho, Mário: A survey on load transportation using multirotor UAVs. En: *Journal of Intelligent & Robotic Systems* 98 (2020), p. 267–296

[79] Wehner, Rüdiger ; Boyer, Martin ; Loertscher, Florian ; Sommer, Stefan ; Menzi, Ursula: Ant navigation: one-way routes rather than maps. En: *Current biology* 16 (2006), Nr. 1, p. 75–79

[80] Yanguas-Rojas, David ; Cardona, Gustavo A. ; Ramirez-Rugeles, Juan ; Mojica-Nava, Eduardo: Victims search, identification, and evacuation with heterogeneous robot networks for search and rescue. En: *2017 IEEE 3rd Colombian Conference on Automatic Control (CCAC)* IEEE, 2017, p. 1–6

[81] Zavlanos, Michael M. ; Egerstedt, Magnus B. ; Pappas, George J.: Graph-theoretic connectivity control of mobile robot networks. En: *Proceedings of the IEEE* 99 (2011), Nr. 9, p. 1525–1540