Distributed control of multi-agent systems: formation strategies and consensus in networked applications

Department of Electrical and Information Engineering
ELECTRICAL AND INFORMATION ENGINEERING
Ph.D. Program
SSD: IINF-04/A–AUTOMATICA

Final Dissertation

Distributed Control of Multi-Agent Systems: Formation Strategies and Consensus in Networked Applications.

---

by
Mohammad Amin REZAEI NAGHADEHI

Supervisors:

Prof. Eng. Saverio Mascolo

Prof. Eng. Luca De Cicco

Coordinator of Ph.D. Program:
Prof. Eng. Mario Carpentieri

Course n°38, 01/04/2023-31/03/2026

این پایان‌نامه دکتری را با تمام عشق و وجودم تقدیم می‌کنم به:

۱. خانواده عزیزم، به پاس حضور همیشگی و حمایت‌های بی‌دریغشان در تمام مسیر زندگی‌ام؛

۲. دوستان ارزشمندم، چه در ایران و چه در ایتالیا، که با باور، همراهی و دلگرمی‌شان همواره در کنارم بودند؛

۳. هم‌وطنان عزیزم در ایران، که آغاز اعتراضاتشان هم‌زمان با پایان این مسیر برای من بود.

اندوه از دست دادن جاویدنامانی که برایشان اشک ریختم، در سطر به سطر این پایان‌نامه با من جاری است و هرگز از خاطرم نخواهد رفت...

This doctoral thesis is dedicated, with all my love and heart, to:

1- My beloved family, for their constant presence and unwavering support throughout my journey;

2- My dear friends, both in Iran and in Italy, who believed in me and stood by my side with encouragement and kindness;

3- My dear compatriots in Iran, whose protests began at the same time as the completion of this chapter of my life.

The sorrow of losing compatriots for whom I have wept is woven into every line of this thesis and will never fade from my memory.

I also wish to sincerely thank ***Prof. Saverio Mascolo****,* ***Prof. Luca De Cicco****,* ***Prof. Farzad Hashemzadeh****,* ***Prof. Silviu Iulian Niculsecu,*** *and* ***Prof. Arben Cela*** *as well as my colleagues at C3LAB, for their guidance, patience, and support throughout this challenging period. Their presence---both academically and personally---provided not only mentorship but also a sense of stability and solidarity, helping me move forward even in the most difficult moments. I am deeply grateful for the discussions, shared efforts, and the quiet encouragement that made this journey less lonely.*

Abstract

This thesis focuses on two important problems in the area of Multi-Agent Systems (MAS) and distributed control: the coordination of drone swarms for area coverage and video acquisition, and the consensus-based management of backup batteries in DC islanded grids. Both problems are studied with the aim of developing control strategies that are scalable, adaptive, and suitable for real-world applications, respecting specific constraints.

In the first part of the thesis, a leader–follower framework based on Model Predictive Control (MPC) is proposed for the coordination of a swarm of drones. Within this framework, a swarm of drones is adopted to perform area surveillance with mounted downward-facing cameras to capture overlapping videos, as well as maintain a distance-based formation while following a predefined path. The framework ensures collision avoidance while sending the captured videos to the Ground Control Station (GCS) for further processing such as video stitching, object detection, etc.

To improve the overall coverage, an online path planning strategy is introduced, allowing the leader to detect uncovered areas and guide the swarm accordingly for rectangle shaped target areas. Then, an improved version of the path planner has been proposed, capable of covering any convex target areas. In addition, the approach takes into account the available network bandwidth by adapting the reference flight altitude to maintain the highest possible video quality. Moreover, an event-triggered approach has been suggested for communication among the drones to improve the efficiency of bandwidth consumption.

A grid-based coverage method is also presented as an alternative solution, where the agents form a structured configuration around a central leader. For this case, a nonlinear controller is designed and supported by mathematical analysis to ensure proper system behavior. Sensitivity analyses are carried out to better understand the influence of the design parameters on system performance.

In the second part of the thesis, the problem of power sharing among backup batteries in DC islanded grids is addressed. A consensus-based control strategy is developed to coordinate batteries with different capacities and initial states of charge, ensuring a balanced distribution of power. To provide a more realistic representation of the system, a converter transfer function is included in the modeling. The results show that the proposed method can achieve reliable and coordinated operation under practical conditions.

Overall, the results of this thesis show that combining the MPC control method and the MAS framework can effectively address complex coordination problems in real-world applications. Specifically, the proposed methods offer practical solutions for applications in drone-based systems and smart energy networks, while maintaining scalability and robustness. Simulation results through MATLAB, Python, and IsaacSim validate the effectiveness of the proposed methods.

Table of Contents

List of Figures

List of Abbreviations

ABR Adaptive BitRate

ABS Aerial Base Station

AI Artificial Intelligence

BESS Battery Energy Storage System

CCM Coulomb Counting Method

CNN Convolutional Neural Network

CPP Coverage Path Planning

DC-MGs DC Microgrids

DCS Distributed Control System

DMC Dynamic Matrix Control

DMPC Distributed Model Predictive Control

ESR Equivalent Series Resistance

ESS Energy Storage Systems

ESU Energy Storage Unit

ET-DMPC Event-Triggered Distributed MPC

ETC Event-Triggered Communication

FCS Fieldbus Control Systems

FoV Field of View

G-CPP Grid-based Coverage Path Planning

GCS Ground Control Station

GPC Generalized Predictive Control

GPS Global Positioning System

GSD Ground Sample Distance

LiDAR Light Detection and Ranging

MAS Multi-Agent Systems

MIMO Multi-Input-Multi-Output

MOS Mean Opinion Score

MPC Model Predictive Control

MPPT Maximum Power Point Tracking

NLS Nonlinear Least Squares

NMPC Nonlinear Model Predictive Control

PSO Particle Swarm Optimization

PV Photovoltaic

QoE Quality of Experience

QoS Quality of Service

RES Renewable Energy Sources

RGB Red Green Blue

RL Reinforcement Learning

RMSE Root Mean Square Error

SG Smart Grids

SISO Single-Input-Single-Output

SoC State of Charge

SSIM Similarity Index Measure

TT Time-Triggered

TT-DMPC Time-Triggered Distributed Model Predictive Control

UAV Unmanned Aerial Vehicle

UUV Unmanned Underwater Vehicle

VQM Video Quality Metric

Scientific Contributions

The scientific contributions that summarize all the activities carried out during the Ph.D. are listed below.

International Conferences:

- **MA. Rezaei**, G. Manfredi, VA. Racanelli, L. De Cicco and S. Mascolo. 2024 Decentralized Control of UAV Swarms for Bandwidth-Aware Video Surveillance Using NMPC. International Conference on Unmanned Aircraft Systems (ICUAS), Chania, Greece, 2024, pp. 947-954.

- **MA. Rezaei**, G. Manfredi, VA. Racanelli, L. De Cicco and S. Mascolo. 2025. An Online Path Planner for Bandwidth-aware Aerial Camera Networks. 17th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), Florence, Italy, pp. 300-305.

- **MA. Rezaei**, M. Rajabinasab, L. De Cicco, SI. Niculescu, A. Cela, S. Mascolo, and Marco Liserre. 2026. Consensus Strategy for State-Aware Coordination of Renewable Energy Sources and Battery Storage Systems in Islanded DC Microgrids. Accepted to 24th European Control Conference (ECC), Reykjavík, Iceland, July 2026.

- VA. Racanelli, **MA. Rezaei**, W. Brescia, N. Barone, L. De Cicco, S. Mascolo. UAV Formation Control with Event-Triggered Communication Using Nonlinear MPC for Coverage Applications. Submitted to IEEE/RSJ International

Conference on Intelligent Robots and Systems (IROS), PITTSBURGH, USA, September 2026. (Under Review)

International Journals:

- **MA. Rezaei**, G. Manfredi, VA. Racanelli, L. De Cicco and S. Mascolo. Formation Control and Path planning of UAV Camera Networks for Coverage Missions. Submitted to IEEE Transactions on Control Systems Technology (Under Review).

- **MA. Rezaei**, F. Hashemzadeh , G. Manfredi, L. De Cicco and S. Mascolo. Swarm Cohesion Control for Coverage Missions with Variable Sensor Scanning Areas. Submitted to IEEE Transactions on Automatic Control (Under Review).

Chapter 1

Introduction

All engineering systems are designed to achieve prescribed performance specifications. These specifications may vary depending on the nature of the system, operating conditions, and external influences. However, such systems may encounter issues that not only degrade performance but can also lead to instability. In certain cases, these failures may result in catastrophic events, including explosions, fires, and significant economic losses. The Texas City refinery explosion of March 23, 2005, which caused 15 fatalities and approximately $200 million in damages, is a well-known example of the consequences of failures in process monitoring and control. Such events show the potential impact of systems deviating from their intended operating conditions and exhibiting unstable behavior. Control engineering addresses these challenges by providing systematic methods for modeling, analysis, and design to ensure that systems satisfy performance requirements while maintaining stability under practical constraints. Its role is therefore critical, particularly in applications where safety and human lives are at risk.

Control engineers develop mathematical models of physical systems to enable their analysis and to characterize fundamental properties such as stability and performance in the presence of disturbances and modeling uncertainties. For this purpose, a wide range of control strategies have been developed for different applications, each

with its own advantages and limitations. Depending on the specific application, appropriate control laws are selected to ensure the desired behavior of the system under consideration.

Before the 20th century, the core concept of feedback, where a system corrects itself based on its output, predates modern control theory. The earliest known feedback device is Ktesibios's water clock from Alexandria around 270 BC, which used a float to regulate water levels. For centuries, similar principles were used in float regulators and other automatic devices. After the Industrial Revolution, in 1788, James Watt's famous centrifugal governor automatically regulated the speed of his steam engine, becoming an iconic symbol of early control. As systems became more complex, so did the need for their analysis. In 1868, physicist James Clerk Maxwell published a mathematical analysis of the governor, explaining the oscillations and instabilities observed in these systems and laying the groundwork for control theory. By the end of the century, mathematicians such as Routh and Hurwitz had developed formal criteria to determine linear system stability later on generalized to non-linear systems by Lyapunov.

Historical Development of Control Engineering (1800-1930)

Between 1800 and 1930, control engineering evolved from practical mechanical regulators to an engineering discipline. The industrial revolution played an important role particularly because of the need to regulate steam engines and industrial machinery. One of the most well-known and influential inventions was the centrifugal governor developed by James Watt for speed regulation of the steam engine. On the other hand, engineers and scientists began analyzing stability and feedback behavior mathematically. Crucial contributions by James Clerk Maxwell and Edward John Routh established the theoretical foundations of stability analysis. By the early twentieth century, advances in electrical engineering, telecommunication, and automatic steering systems expanded the field, leading to modern control theory after 1930 [1].

Classical Control and Institutionalization (1930–1955)

This period was a rapid development for control engineering due to the increasing need for automatic regulation in industrial processes, communication systems, aircraft, and military applications. Moreover, classic control became more popular through the development of feedback systems, servomechanisms, and frequency-response techniques. From the military applications viewpoint, World War II played a major role in enhancing the research, especially in radar tracking, fire-control systems, and aircraft control where, important analytical methods such as Nyquist stability analysis, Bode plots, and root locus methods were introduced and adopted. Later, the early foundations of modern control theory, including state-space approaches and optimal control concepts also emerged [2].

The Modern Control Era (1955–1980s)

Cold War and the Space Race influenced modern control theory due to the high technology demands. These challenges highly required precise control of complex systems, including missiles and spacecraft. This era is characterized by the introduction of state-space methods, which describe dynamic systems using sets of first-order differential equations. A key contributor to this framework was Rudolf Kalman. Unlike classical approaches, state-space representations allow for the systematic treatment of multi-input multi-output (MIMO) systems. In addition, this period saw the development of optimal, robust, and adaptive control theories, significantly expanding the scope of the field.

The Digital Age and Contemporary Developments (1980s–Present)

The advent of low-cost and powerful microprocessors in the 1980s led to a paradigm shift in control engineering. Digital computers became integral components of control systems, enabling the development of digital control theory and discrete-time analysis techniques such as the Z-transform. In industrial environments, this transition facilitated the evolution from centralized control architectures to Distributed Control System (DCS), improving both reliability and scalability. Subsequently, the

1990s saw the emergence of Fieldbus Control Systems (FCS), which enabled fully digital, bidirectional communication between field devices.

More recently, control engineering has increasingly incorporated concepts from Artificial Intelligence (AI), including Reinforcement Learning (RL). These approaches are playing a growing role in addressing modern challenges in areas such as robotics, autonomous systems, and large-scale networked control.

1.1 Multi-Agent Systems

In 1970-1980, the concept of distributed artificial intelligence arose where complicated tasks were broken down into a variety of simple tasks to be done. So, these simple problems could be tackled by several computational units to result in solving a more complex task. The main contributions to the field were initiated by Carl Hewitt where an actor model was developed for concurrency in 1973, and by Marvin Minsky who proposed "Society of Mind" in 1986. Then, with the evolution of technology, new ideas based on these concepts were proposed including the concept of *agent*, which is defined as a computational unit that can perceive, decide, and act to perform the assigned tasks. The concept of agent has been assigned to different subjects such as robots, houses, power units, drones, and even human beings.

A *multi-agent system* consists of several agents (subsystems) where each of them performs assigned tasks to reach the global goal of the whole system in a shared environment. The agents are named homogeneous or heterogeneous depending on their hardware and software features. As an example, consider a team of exactly the same mobile robots where they cooperate and communicate to do a specific task; therefore, the robots are homogeneous agents since all of them are the same type of software and hardware. On the other hand, think about a team of ground robots communicating with several aerial robots to cover an area from above and on the ground. In this case, the agents are heterogeneous due to the fact that mobile

robots (agents) are different from the aerial ones in both software and hardware. The applications of such systems can extend to several fields, including water management [3] and human science [4], rather than being limited solely to engineering.

Communication is one of the crucial features employed by multi-agent systems to improve efficiency. Instead of having several agents working independently without any awareness of the other agents, they communicate to improve awareness of the global task and state of the system. Thanks to high-speed communications among the agents, multi-agent systems became more and more efficient and trustworthy. Modern AI methods use multi-agent-based training for their agents to take advantage of parallel learning and model sharing which can be found in federated and distributed learning.

Since a global and complex task is broken down to smaller tasks to be solved, the needed time, energy, and computational burden are also divided. This means that using multi-agent systems, if done correctly, leads to solving complicated problems in a shorter time with simpler technology, and with cheaper agents instead of expensive ones. However, this comes at the cost of further challenges such as the design of efficient communication protocols and algorithms to provide parallel computations and actions among the agents. Nevertheless, the advantages of these systems highly outperform the disadvantages leading to a wide range of applications in industries and research.

The theory of multi-agent systems is divided into several topics including Consensus [5], Formation Control [6], Flocking [7], Surrounding [8], Containment [9], etc. Each of these topics deals with specific mathematical formulations to perform the desired tasks using homogeneous or heterogeneous agents based on predefined goals.

The applications of multi-agent systems find applications in many fields such as robotics, power systems, military applications, search and rescue, parallel computations, decision making, and etc. This thesis focuses on robotics and power systems, where drones and backup batteries are modeled as agents within a multi-agent system.

These agents cooperate to accomplish assigned missions while satisfying application-specific constraints.

1.1.1 Consensus

The most well-known topic in the literature of multi-agent systems is consensus, where a number of agents pursue the same/different goals by having an agreement on a specific value, behavior, performance, etc. Consensus is not limited to engineering problems since there are various researches regarding decision making or decision consensus even in AI [10] and in human science [4]. Concerning engineering applications, consensus control plays a significant role in networked science, robotics, resource allocation, power systems distribution, etc. For instance, the position control of several robots when they want to reach a specific meeting point with only exchanging information between the neighbors is a consensus control problem. Another example can be power systems distribution in which several power units need to distribute the demanded energy among the users, and to do so, the equilibrium point which is the consensus point is reached by different power units by sharing their energy based on their capacities while satisfying specific criteria using the consensus protocol.

1.1.2 Formation Control

Formation control is a specific form of a consensus problem where the goal of the agents or swarm is to form a specific geometric shape to fulfill some goals. This method is widely used in the motion planning of mobile robots and drones to have a specific cooperation by taking advantage of their movement structure and geometric shapes. As an example, the Spectacular Intel Drone Light Show in Olympic 2020 in Tokyo was an example of using a large number of drones where they kept time-varying distances from each other to form the desired shapes in the sky. Consider that coordinating a huge number of drones is not an easy task due to the high amount

of communications and risks of collisions among the drones. Another example of formation control is in jet fighter shows where several jet fighters fly together with a specific distance among them. Generally, this also has military applications for the aircraft when they move together in order to make the enemy confused in choosing which one to shoot at. Lots of several examples can be mentioned for this topic which shows its wide range of applications in different areas.
Other applications of formation control include agricultural robots/tractors for efficient harvesting, self-driving cars and platooning, cooperative sonar scans where underwater robots provide a comprehensive scan of the given area, and numerous other applications where coordination of a group of agents is needed to keep a specific distance from each other.

Formation control has several methods including:

1. Distance-based formation control,
2. Displacement-based formation control,
3. Leader-follower,
4. Virtual Structure,
5. Behavior-based.

Each of the above methods solves the problem from different perspectives, adding/removing challenges with respect to the others. Distance-based formation control is the one which has been used in this thesis, where a desired distance among the agents is set to be respected to form a specific coordination satisfying performance-related goals.

1.2 Model Predictive Control

MPC is an advanced control technique that is widely adopted in industry and research. Its origins can be traced back to process control in the oil and chemical industries in the 1960s. Early industrial implementations were primarily based on heuristic

and model-based predictive strategies. The formalization of MPC using state-space models and the receding horizon control framework emerged in the 1970s. In the 1980s, Dynamic Matrix Control (DMC) was introduced, relying on step response models to compute control actions, followed by the development of Generalized Predictive Control (GPC) which is based on impulse (or pulse) response models and a more explicit optimization framework.

In industrial practice, simplicity and reliability are often preferred, and PID controllers remain widely used, particularly for fast and well-understood processes. However, for systems with slower dynamics, multivariable interactions, constraints, or the availability of future reference or disturbance information, MPC can solve control problems where practical constraints exist such as a limited amount of energy provided in a short time, limited bounds for specific variables in the system, etc.

A key feature of MPC is its ability to explicitly handle constraints on inputs, states, and outputs within the control design. Moreover, MPC provides a systematic framework for controlling Multi-Input-Multi-Output (MIMO) systems without substantially increasing design complexity compared to Single-Input-Single-Output (SISO) cases. MPC computes control actions that account for predicted future system behavior and desired performance objectives by solving an optimization problem over a finite prediction horizon.

MPC is also suitable for time-delay systems, as these effects can be naturally incorporated into the prediction model. At each sampling instant, the control problem is solved again using updated measurements, following the receding horizon principle. This allows the controller to compensate for disturbances, model uncertainties, and noise. However, this repeated online optimization introduces a non-negligible computational burden.

Despite these challenges, the ability of MPC to handle constraints, multivariable interactions, and predictive information has made it one of the most widely used advanced control strategies in both industrial and research contexts.

1.3 Coverage and Path Planning

One of the main challenges in robotics is the computation of feasible and efficient trajectories that allow a robot to accomplish a given task while satisfying performance criteria such as time, energy consumption, and safety. Typical applications include firefighting in complex environments [11], search and rescue operations [12], surveillance [13], and border monitoring [14].

These applications rely on robotic platforms such as drones and mobile robots, which must operate under practical constraints including limited energy, restricted operating time, and environmental obstacles. As a result, robots require appropriate motion planning strategies to achieve their objectives efficiently while avoiding collisions and respecting forbidden regions.

Depending on the task, the problem may take different forms. In path planning, the goal is to compute a collision-free trajectory from an initial to a target configuration. In coverage problems, instead, the objective is to ensure that an entire area is visited or observed by the robot's sensing capabilities. Formally, given a region of interest and a robot equipped with a sensor (e.g., camera or Light Detection and Ranging (LiDAR)) with a given footprint, the goal is to design a trajectory that guarantees complete or sufficient coverage of the area.

Several challenges arise in these problems. Robots often exhibit non-holonomic dynamics, which impose constraints on their motion. Additional limitations include obstacle avoidance, communication range constraints in multi-robot or data-driven applications, and energy efficiency requirements. Furthermore, in dynamic or partially unknown environments, planning must be updated online based on newly acquired information, leading to exploration problems.

An effective trajectory should balance multiple objectives, such as minimizing travel time or energy consumption, satisfying system and environmental constraints, and achieving task-specific goals such as complete coverage. However, these objectives are often conflicting, making the problem inherently complex and motivating

ongoing research in the field.

In recent years, aerial robots such as drones have become increasingly important due to their high maneuverability and flexibility. They are widely used in applications such as surveillance and search and rescue, where they can provide real-time data (e.g., video streams) to support decision-making. To accomplish these tasks, both offline and online planning methods are employed, depending on the level of prior knowledge about the environment.

1.4 Quality of Experience, Quality of Service

In recent years, social media and online platforms have enabled the widespread distribution of video content to users worldwide via Internet-based infrastructures. A substantial volume of data is continuously transmitted across these networks, a significant portion of which consists of high-resolution and live video streams delivered through platforms such as YouTube and Instagram. Due to the inherently large size of video data, network congestion and communication bottlenecks may arise under various conditions. For instance, high traffic load on specific transmission links and limited available bandwidth are among the primary factors that can degrade network performance. Consequently, users may experience interruptions in video playback, including buffering, freezing, increased latency—particularly in live streaming scenarios—and overall degradation in visual quality.

To evaluate video quality, a subjective feature named Quality of Experience (QoE) has been suggested, which shows how good a video is from the perspective of a human. Note that this metric is a subjective feature, which means that for the same video, people propose different QoEs. As an example, a video that is 720p is satisfactory for one user leading to a higher QoE while it receives a lower QoE from another user. Note that this is not specifically about the video quality but also extends to the other features of a video while playing including the pauses during the video, delayed play,

and even the content.

Unlike QoE, Quality of Service (QoS) is considered objective since it is characterized by system-level metrics. Some of these metrics can be bitrate, packet loss, delay, jitter, etc. QoS includes features that can be measured to exactly compare videos or corresponding transfer networks. The provided service directly affects the QoE, but there is no exact relationship between QoE and QoS due to the nature of QoE. Therefore, a high QoS does not always mean a high QoE and vice versa.

1.5 DC Islanded Grids

Currently, due to global warming, human beings try to provide their required energy using Renewable Energy Sources (RES) such as Photovoltaic (PV), wind farms, hydropower, and etc. Various energy resources can replace fossil fuels that are polluting the environment. Traditionally, power plants produce electricity and distribute it to all parts of a country forming a connected energy network. The same method cannot work for RES units since sending them over far distances will increase the waste of energy, meaning that there is no efficiency at all. Therefore, people try to take advantage of natural resources for energy in remote and sensitive areas. This idea led to islanded grids where each unit produces, regulates, and consumes its own produced energy. The idea works well if the design of the grid can be done accurately based on the amount of consumption and production; however, this is not possible ideally.

Since the amount of demanded energy and the produced one are not always the same, islanded grids take advantage of backup batteries where they produce (absorb) the demanded (exceeded) energy in the grid. This idea made the grids capable of being detached from the main fuel-based electricity line. However, one should note that the backup batteries have a lifespan, and after some amount of time, they need to be replaced with new ones. Generally, the cost of these batteries is high which makes

grid designers consider this parameter while designing the grid. Batteries' lifespan depends on their total number of charge or discharge cycles, and if this process can be managed intelligently, it can prolong the batteries' effective life leading to a lower cost of energy.

1.6 Thesis Outline

This thesis addresses two interconnected problems within the field of distributed control systems: the motion control of drone swarms and the consensus problem in backup batteries for DC islanded grids. Both topics are studied under the common framework of multi-agent systems, with an emphasis on coordination, scalability, and robustness in the presence of system constraints.

Motivation and Research Gap

Recent advances in multi-agent systems have enabled significant progress in applications such as autonomous aerial surveillance and smart energy systems; however, several challenges remain open. In drone swarm applications, achieving efficient area coverage while maintaining high-quality data acquisition under unpredictable available network bandwidth is still a complex problem. Existing approaches have not considered the coverage problem of a swarm of drones while maximizing the QoE of the captured videos considering the available network bandwidth. Moreover, one step forward has been taken, which is proposing an online path planning for the swarm to improve the coverage.

Another open challenge in this specific application is the efficient consumption of the available network bandwidth, where using event-triggered communication among the agents would lead to saving the available network bandwidth for video streaming, resulting in higher-quality videos.

Apart from these, since the application of surveillance using Unmanned Aerial

Vehicle (UAV)s is not addressed well, a grid-based coverage approach has been suggested taking into account a nonlinear controller that is used to drive the agents to positions maximizing the coverage with respect to the previously suggested methods. This method considers the limited velocity of the drones while providing a mathematical proof for the proposed controller to guarantee the convergence of the agents toward the given reference positions.

Similarly, in DC islanded grids, the coordination of heterogeneous backup batteries with different capacities and states of charge remains a critical issue. Conventional control strategies may not guarantee fair and stable power sharing, particularly when realistic system dynamics are considered.

This thesis aims to address these gaps by developing novel control and coordination strategies based on MPC and consensus-based methods.

Research Objectives

The main objectives of this thesis are as follows:

- To develop an MPC-based framework for coordinated motion control of drone swarms that maximizes area coverage and video quality under stochastic available network bandwidth.
- To design an online path planning strategy that enables adaptive exploration of unknown or partially covered areas.
- To propose an event-triggered communication framework for the drones to consume the available network bandwidth efficiently.
- To propose an alternative grid-based coverage method with improved efficiency and stability guarantees.
- To formulate and analyze a consensus-based control strategy for power sharing in heterogeneous backup battery systems within DC islanded grids while

validating it with a real converter model.

Main Contributions

The main contributions of this thesis can be summarized as follows:

- A leader–follower MPC-based control framework for drone swarms that ensures collision avoidance, formation maintenance, and bandwidth-aware motion to maximize the video quality.
- An online path planning algorithm that dynamically guides the swarm toward uncovered regions to enhance area coverage.
- An event-triggered communication framework for the drones to save the available network bandwidth for the video streaming.
- A grid-based coverage strategy with a prescribed nonlinear controller, including a formal mathematical stability analysis.
- A consensus-based control approach for heterogeneous battery systems that achieves balanced power sharing despite differences in initial states of charge and capacities.
- The incorporation of a realistic converter transfer function model to validate the effectiveness of the proposed consensus strategy.

Thesis Organization

The remainder of this thesis is organized as follows:

- Chapter 1 provides a general introduction, outlining the history, motivation, objectives, and scope of the thesis.

- Chapter 2 reviews the fundamental concepts of UAVs, multi-agent systems, formation control, consensus theory, QoE, MPC, and DC islanded grids, along with relevant literature.
- Chapter 3 presents the related work on swarm control of drones and DC islanded grids, including the research that has been conducted within these framework.
- Chapter 4 introduces the methodologies and simulation results to validate the effectiveness of the proposed method for motion control of drones with a predefined path for a rectangle-shaped area.
- Chapter 5 discusses the online path planning for the leader agent for both rectangle-shaped and convex-shaped areas. In addition, a deep sensitivity analysis has been performed on the hyperparameters of the cost functions of the optimization problem.
- Chapter 6 suggests an event-triggered communication framework for the drones, as well as simulation results validating the method.
- Chapter 7 discusses the grid-based coverage method with a nonlinear prescribed controller to enable a comparison with the MPC-based approach.
- Chapter 8 introduces the consensus-based control approach for DC islanded grids and validates it using the real transfer function model of a converter.
- Chapter 9 concludes the thesis and discusses potential directions for future research.

Overview of the Proposed Approaches

In the first part of the thesis, the problem of coordinating a swarm of drones is investigated. Starting from a predefined trajectory, a leader–follower framework is

adopted in which the leader agent follows a reference path while the follower agents maintain a distance-based formation. This structure enables the swarm to capture overlapping video streams while avoiding collisions.

An online path planning method is then proposed, allowing the leader agent to identify uncovered areas and guide the swarm accordingly. In both predefined and adaptive scenarios, MPC is employed to ensure formation tracking and altitude regulation, while maximizing video quality based on available network bandwidth. A sensitivity analysis is conducted to evaluate the impact of weighting factors in the MPC cost function.

In addition, an event-triggered communication framework has been suggested to optimize bandwidth consumption while minimizing the number of communication occurrences among the drones without losing the desired performance of the system.

To further enhance coverage efficiency, a grid-based method is introduced in which the leader is positioned at the center of the formation and the remaining agents form a polygonal structure. A prescribed nonlinear controller is designed to ensure accurate trajectory tracking while limiting the velocity of the drones, and a formal mathematical proof is provided to guarantee the tracking of the reference position, which automatically guarantees the performance of the whole system.

In the second part of the thesis, the focus shifts to DC islanded grids, where a consensus-based control strategy is developed for coordinating backup batteries in DC islanded grids. The proposed approach ensures balanced power sharing among heterogeneous units with different initial conditions. A realistic converter model is incorporated to represent system dynamics and validate the effectiveness of the control strategy.

Chapter 2

Background

In this chapter, the necessary background and basic concepts used in this thesis will be presented to provide an understanding for the later chapters. Section 2.1 provides an overview of aerial robotics, discussing their evolution, applications, advantages and disadvantages, and key features that make them well-suited for research and engineering applications. Then, Section 2.2 presents the general concepts of MAS, addressing two main subtopics: consensus and formation control. The necessary graph theory notations are introduced to facilitate the subsequent mathematical formulations of consensus and formation control. Short examples have been provided to make the reader aware of the topics and how they are being used in the proposed methodology. Section 2.3 introduces the MPC and its formulations. It provides a general framework of MPC with a cost function including several terms as well as some constraints to show how practical limitations may be included in the formulation of the MPC.

2.1 Aerial Robotics

Robotics is the field of engineering that deals with the design and control of machines to perform the assigned tasks with a certain degree of autonomy. Different robotics

fields include Mobile robotics, industrial robotics, aerial robotics, biorobotics, and so on, where the goal in all of them is to do the tasks instead of humans to save energy, time, money, and even lives in hazardous situations. A Greek philosopher and mathematician named Archytas of Tarentum is said to have built a steam-powered wooden bird that could fly for about 200 meters. This device, known as "The Pigeon," is often considered the first recorded instance of a self-propelled flying object and a precursor to all aerial robotics, occurring in 350 BCE. After WWI, aerial robotics started to grow fast thanks to radio technology. Different types of flying bombs have been developed during the war, and later after the war, drones and UAVs were born. Nowadays, drones offer a very wide range of applications which makes them the first choice to use instead of huge flying vehicles such as airplanes and helicopters.

Generally, aerial robots are categorized into different types such as Multi-rotor, Fixed-wing, Single-Rotor Helicopter, Flapping-wing (Ornithopter), Solar Powered High Altitude UAVs, and etc. Each of these robots has specific features with specific applications; however, they can be used interchangeably in some cases. The differences among them mostly go back to their design purpose and structure. Based on the application, the design procedure becomes complicated or simplified. For instance, a quadrotor that is used for taking videos or photos from high altitudes does not need fast calculations, speed, and maneuverability, so it can be built with a simple structure. On the other hand, a drone that will be used for firefighting tasks needs a huge amount of energy (a huge battery) to carry a water tank to a high altitude; therefore, it needs a complicated structure capable of carrying a heavy battery and water tank.

Fixed-wing UAVs are designed to fly similar to airplanes which means that they do not have the capability of standing still to do a task but they have to fly non-stop. This makes them perfect to be used for tasks where they need long-range and long-endurance missions such as firefighting, especially for vast areas where they act like firefighting planes, military missions due to their fast speed, large-scale agricultural

monitoring, large-scale search and rescue missions, and etc.

Among the mentioned types of aerial robots, the one that is considered in this thesis is multi-rotors since they are more common and come in different sizes and capabilities, and they are used more in the context of capturing videos for surveillance. Multi-rotors are divided into different types including Tricopter (three rotors), Quadrotor (four rotors), Hexacopter (six rotors), Octocopter (eight rotors), etc. Increasing the number of rotors adds complexity and better maneuverability at the same time. They are used for a variety of purposes from a simple toy to firefighting tasks. Thus, they are the most common drones to be used. They are widely accepted in the research communities to develop ideas based on them which led to providing nonlinear and linear dynamic models for multi-rotors. The advantages of multi-rotors with respect to the other types of drones are:

1. Available in various sizes and capabilities,
2. Flexible in maneuver,
3. Ability to stand still to do tasks needing this feature,
4. Simpler mathematical model,
5. Affordability due to the lower price.

2.2 Multi-agent Systems

Multi-agent systems received huge attention due to their application in industry and research. Agents share required information among themselves to increase situational awareness about the environment while making better decisions to fulfill the goals. There are several classifications for multi-agent systems in which the structures of these systems differ to perform a wide range of tasks. Some of the significant classifications have been introduced to show the differences among them. Based on the

agents' roles, the system can be divided into i) Leader-Follower [15] and ii) Leaderless [16]. In the leader-follower case, one or several agents are the leader agents in which the choice has been made for specific reasons. The leader agent typically has greater situational awareness than the others, may have more communication links, can be computationally more powerful, and may have access to global information for decision-making. On the other hand, when there is a leaderless multi-agent system, there is no difference in the situational awareness or computational capabilities among the agents. However, in the literature, we refer to the leader agent as the one having more information with respect to the others. Computational capabilities can be different for a team of agents due to their heterogeneousness but still, none of them has more information with respect to the others leading to a leaderless multi-agent system.

Another classification for multi-agent systems is being a Centralized [17] or Decentralized [18] multi-agent system. This is more related to how the agents perform actions and computations, which directly affects the efficiency of the system. In a centralized approach, the agents communicate with the leader or a central system to share all the information. Then, this central agent or leader, based on the global information, computes and assigns the commands for each agent, sending them back to act based on those commands. This can provide a globally optimal solution as well as providing a stability proof for the whole system; however, the scalability of the system would be difficult, the amount of communications would add delay or packet losses, and the system is highly prone to failure in case the central agent or the leader fails.

In a decentralized multi-agent system, agents communicate locally with neighboring agents, sharing information and coordinating their actions without a central coordinator. This approach provides a high degree of scalability which is one of the main strength points of the multi-agent systems. Also, the system is not relying on a single failure point such as the central case which improves the robustness of the

whole system against the possible faults. Moreover, the amount of needed communication decreases drastically which increases the robustness of the communication where the delay or packet losses are as low as possible. On the contrary, reaching an optimal solution for the team is not guaranteed due to the fact that there is no global information regarding the whole system but the agents use partially shared information. Providing formal stability guarantees in decentralized systems is mathematically challenging due to the coupled nature of agent dynamics, the lack of global state information, and the potential for asynchronous decision-making. However, stability can be achieved under certain conditions, such as when interaction graphs are connected and control laws satisfy network-wide Lyapunov criteria.

2.2.1 Graph Theory

This subsection discusses the central concepts about graph theory needed for the rest of the thesis, including a brief introduction and mathematical formulations. A ***graph*** is denoted by $G(\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = v_1, \ldots, v_N$ is the set of nodes, also identified simply by their indices $i = 1, \ldots, N$, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges such that an edge (v_i, v_j) denotes the information flow from node i to node j. Moreover, the set of neighbors of node v_i is defined as $\mathcal{N}_i = \{v_j \in \mathcal{V} : (v_j, v_i) \in \mathcal{E}\}$. A ***connected graph*** is a graph in which there is a path from any node to any other node, and a ***fully connected graph*** is a graph in which there is a direct edge between any pair of nodes. A ***tree*** is a connected graph with N nodes and $N-1$ edges that contains no cycles. Moreover, removing any edge from a tree results in a disconnected graph. In order to show the connections between various nodes, a matrix named ***Adjacency*** matrix is used as $A = \{a_{ij} = 1 : (v_i, v_j) \in \mathcal{E}\}$. For an undirected graph, the ***Degree*** matrix is a diagonal matrix that shows the number of connected edges to each node defined as $D = \{d_{ij} : \Sigma_{j=1}^{N} a_{ij} \text{ if } i = j\}$. The degree and adjacency matrices are used to define an important matrix known as the ***Laplacian*** matrix, which serves as a fundamental tool in multi-agent systems by capturing the key features of the

connections among the nodes of the system. The Laplacian matrix is defined as $L = D - A$ where for an undirected graph, it would be a symmetric matrix. For a connected graph, the Laplacian matrix has an eigenvalue equal to 0 while the other eigenvalues are greater than 0. Therefore, a connected underlying graph leads to a semi-positive definite Laplacian matrix which introduces a crucial feature to analyze the stability of the connection network.

2.2.2 Consensus

The most famous topic in multi-agent systems is consensus, where a swarm of agents tries to reach the same value for their states [19], outputs [20], or any other feature. Applications include sensor fusion, mobile robots, drones, power systems, etc. There is a wide literature regarding the consensus among the agents in the computer science and control engineering communities. Here, a brief mathematical introduction to the consensus problem will be presented.

Consider a team of mobile robots that, starting from different initial conditions, try to reach each other with only sharing information locally, and without global information. The states of this multi-agent system are the positions in $2D$. Therefore, $X_i = [x_1, x_2]$ is the position vector for each of the agents. A first-order dynamic can be chosen for the agents to make the problem simpler where $U_i = [u_x, u_y]$.

$$\dot{X}_i = U_i \tag{2.1}$$

In the consensus problem, agents share their states based on the underlying communication graph to compute the error from the neighbors' positions to calculate the required U_i that compensates for this error to reach an agreement point. Therefore, in a classical consensus protocol, the control input is defined as a scaled difference between the states of the neighbors and the state of the agent itself.

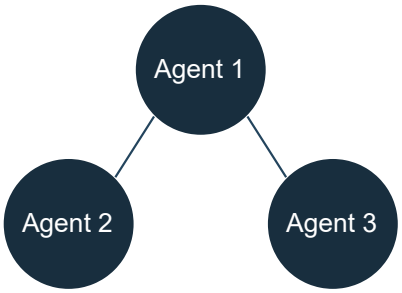


Figure 2.2.1: Communication graph for three agents

$$U_i = \Sigma_{j \in N_i} a_{ij}(X_j - X_i) \tag{2.2}$$

where N_i is the neighbor set of agent i, and a_{ij} are the entries of the adjacency matrix representing the underlying communication graph. Substitution of (2.2) into (2.1) will result in (2.3).

$$\dot{X}_i = \Sigma_{j \in N_i} a_{ij}(X_j - X_i) \tag{2.3}$$

Consider a team of three agents where the communication graph among them is as in Fig. 2.2.1.

Using (2.3), one can compute the evolution of the states of the system as follows.

$$\begin{bmatrix} \dot{X}_1 \\ \dot{X}_2 \\ \dot{X}_3 \end{bmatrix} = \begin{bmatrix} (X_1 - X_1) + (X_2 - X_1) + (X_3 - X_1) \\ (X_1 - X_2) + (X_2 - X_2) \\ (X_1 - X_3) + (X_3 - X_3) \end{bmatrix}$$

Now, transforming to a state-space form, we get:

$$\begin{bmatrix} \dot{X}_1 \\ \dot{X}_2 \\ \dot{X}_3 \end{bmatrix} = \begin{bmatrix} -2 & 1 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix}$$

If we write the adjacency, degree, and Laplacian matrices for the graph shown in Fig. 2.2.1 based on the explanation in section 2.2.1, we can easily notice that the Laplacian matrix has emerged in the formulation of the states $(-L)$. This is the reason why the Laplacian matrix is used and why it is important since it shows the relation between the agents as well as the connection level. Also, it is used to show the stability of a multi-agent system.

$$A = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, D = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, L = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}$$

The given example was a roadmap to show how the Laplacian matrix is formed and then we can derive the formula for multi-agent systems.

$$\dot{X} = -LX \tag{2.4}$$

Equation (2.4) with a semi-positive definite L would be desired since it will steer the states of the multi-agent system toward a bounded constant value (so the agents would have stable states) [21]. This section provided a brief explanation about the consensus among several agents for a continuous-time dynamic. The same results hold for the discrete-time case with some small changes affected by the sampling time.

2.2.3 Formation Control

The other interesting topic in the context of multi-agent systems is formation control. It determines how a team of agents should keep relative distances (or errors between their states) from each other to form a specific geometric shape. The main application of this method can be robotics since mobile robots and drones, when moving in a team, need to avoid collisions while keeping specific distances to stay in the defined communication range; otherwise, they may lose the connection leading to an unstable system. Formation control, as previously explained, includes several methods to be used to form the desired formation among the agents. Throughout this thesis, we use the most famous method which is called "*Distance-Based Formation Control*" in which the agents keep the specified distance among their neighbors. To do this, the method introduced in the consensus section will be used; however, with a small difference which is adding the specified distance among the states of the agents.

2.2.3.1 Distance-Based Formation Control

The formulation of the distance-based formation control is provided below and is very similar to the case of consensus control. Consider $X_i = [x_1, x_2]$ as a $2D$ vector of the ith agent where it is desired to define a formation among the agents to keep a relative distance. As an example, consider a team of four agents that are required to form a quadrilateral-shaped formation. To do this, consider Fig. 2.2.2 as the desired formation of the agents. The desired distances are shown in the figure; so the agents will adjust their positions to reach the assigned distances among them.

To reach the desired distances, one can use distance-based formation control, which is inspired by consensus control for agents with simple integrator dynamics, as described in (2.1). The control input for this case will consider $e_{ij} = X_i - X_j - d_{ij}$ as the error to be compensated. If e_{ij} goes to zero, then the distance between X_i and X_j will become d_{ij}, and the formation will be reached.

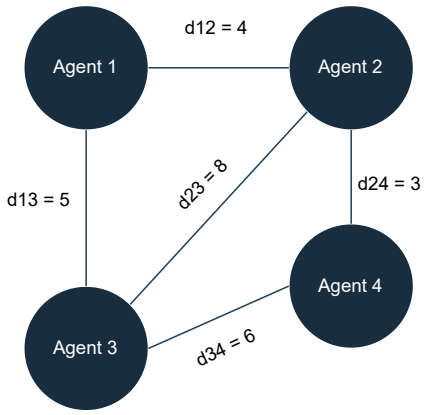


Figure 2.2.2: A network of four agents with desired distances among the agents.

$$U_i = \Sigma_{j \in N_i} a_{ij}(X_j - X_i - d_{ij}) \tag{2.5}$$

As seen from Eq. (2.5) and Eq. (2.3), distance-based formation control is a special case of consensus control but differs in the error between the states of the agents. For consensus, the error must be zero, and for formation control, this error will be assigned a non-zero value.

2.3 Model Predictive Control

MPC is an advanced control method which is built on both Optimal Control and Adaptive Control since it finds the optimal control input based on the constraints and situations that may vary. MPC is widely used in industries such as the petroleum industry, refineries, chemical reactions, etc. In addition to industrial applications, it is widely adopted in the research community, robotics, aerospace, biology, etc. Due to its wide range of applications, as well as its powerful features in handling several factors such as delay, Multi-Input-Multi-Output (MIMO) systems, dead-time, and constraints, MPC became one of the most highly attractive controllers among control

engineers. As mentioned, there are several advantages for this controller to be chosen; however, its greatest disadvantage is its high amount of computation at each time sample. Moreover, as it comes from its name, it needs the exact model of the system to produce the optimal control input which is not trivial to provide in all cases. However, there are several methods including robust MPC, belief-space MPC, and Data-Driven MPC where they deal with the uncertainties that may appear in the modeling of the plant.

In most cases, MPC is characterized by a plant model, a cost function to be minimized (J), a reference signal/state to be tracked (X_{ref}, R_{ref}), an optimal control input to be injected into the system (u), a prediction horizon (N_p), a control horizon (N_c), a terminal state or condition (X_N), and there can exist some constraints to be respected. As an example, a basic formulation for an MPC problem is illustrated below to show how it is formed and what kind of problem it solves. Consider a first-order discrete system as in Eq. (2.6) where x is the state vector of the system, u is the input vector of the system, T_s is the sampling time, and k is a time sample.

$$x(k+1) = x(k) + u(k)T_s \tag{2.6}$$

Assume that we want to steer the state x toward a reference state such as X_{ref} while satisfying i) the constraint for the dynamics of the system in Eq. (2.6), ii) saturating u between $-M$ and M, and iii) saturating the rate of the control input between $-N$ and N to make the problem more practical. So, we assumed three constraints to be satisfied while optimizing the cost function to derive the control input. The general problem has the form as in Eq. (2.7).

$$
\begin{aligned}
\text{argmin}_u\, J = {} & \Sigma_{k=0}^{N_p-1}|x(t|t+k+1) - X_{ref}(t+k+1)|_2^Q \\
& + \Sigma_{k=0}^{N_c-1}|u(t|t+k)|_2^R + \Sigma_{k=0}^{N_c-1}|\Delta u(t|t+k)|_2^W \\
& + |x(t|t+N_p) - X_{ref}(t|t+N_p)|_2^P \\
\text{S.t.} & \\
& 1) x(k+1) = x(k) + u(k)T_s \\
& 2) |u| \leq M \\
& 3) |\Delta u| \leq N \\
& 4) ...
\end{aligned}
\tag{2.7}
$$

State Tracking Term ($x \rightarrow X_{ref}$): The first term of the cost function relates to the tracking of the state of the system during the prediction horizon. The term $x(t|t+k+1)$ means the prediction of the state at the future time instant $t+k+1$ while we are in the current time instant t. This term gives a vector for the optimization whose size depends on the state size and the prediction horizon's length. Having a positive weighting matrix Q and using the norm two, the vector for the optimization will be provided. Since the current state of the system ($x(t)$) is measured and cannot be manipulated, the optimization starts from the next sample time which is $x(t+1)$.

Control Input Minimization Term (u): The second term is related to the optimization of the magnitude of the control input to reach the goal with the minimum amount of control input which can be translated to the required energy. As you can see, the optimization includes the current input $u(t)$ unlike the first term which does not include the state of the current time instant $x(t)$ because it is desired to optimize the current control input.

Control Input Rate Term (Δu): The third term minimizes the changes in the control input to reach the desired goal with the minimum changes of the control input. As an example, for applications with fast-changing reference signals, this will limit the oscillations in the control input since each oscillation will add a minimization cost

to the problem. Also, in some applications, velocity control is preferred where the control input is the speed and therefore, Δu becomes the acceleration which directly represents the energy for the system.

Terminal Cost Term ($x(t|t+k+N_p) \rightarrow X_{ref}(t+k+N_p)$): The fourth term in the cost function is a term to guarantee that the state of the system will reach the desired state by the end of the prediction horizon. Another reason to add the terminal cost is to provide a guarantee to have asymptotic stability for the system since we ask the optimization problem to provide control inputs where, in the end, the states move to the equilibrium point. Note that sometimes engineers add this term as a constraint which then completely provides a stability guarantee if the optimization problem is convex.

Prediction Horizon (N_p): Prediction horizon determines how far the system can see and use the future information to perform the act of prediction and compute the control law based on that. Obviously, taking a huge value for N_p will make the computation burden heavier and even useless. There is no straightforward method to determine the optimal prediction horizon for the optimization; therefore, trial-error is still the best possible way to determine the best prediction horizon.

Control Horizon (N_c): Control horizon determines how many future control inputs should be optimized or how flexible the control inputs can be chosen to adjust the system with the given reference signal or the reference state. Again, choosing a huge value for the control horizon will lead to a heavier computation burden but at the same time it will give flexibility and a better transient response. As mentioned for the prediction horizon, there is no straightforward method to determine the optimal control horizon value.

In general, there is a trade-off in choosing the values for N_p and N_c, and the computation burden of the optimization. Choosing huge values would lead to better responses since more information will be optimized; however, this requires greater computational power and vice versa. The basic rule is to choose these values as

$N_c \leq N_p$, where the equality case provides the highest flexibility for the controller.

Weighting Matrices in Eq. (2.7) are used to show the importance of each term for the optimizer. Having a huge weight drives the optimizer to minimize that term more than the other terms. The weight matrix Q weights the states of the system and in the case of a vector of states, Q can weight the states differently rather than having the same value for all of them. The same holds for R which weights the control inputs of the system meaning that a greater R will lead to expensive control; so, the control inputs will be small. Again, for the case of a vector of control inputs in an MPC framework, the matrix R can be used to weight each control input independently. In addition to the previously defined weights, the matrix P assigns a penalty to the terminal state (or final condition) of the system, encouraging the system to reach the desired final condition through the applied control inputs.

The constraints in (2.7) refer to the practical limitations that must be considered in a real engineering problem. The first term means that the states must respect the evolution rule based on the computed control inputs. The second term limits the control inputs between two values which shows that in practice the control inputs cannot be produced as much as the user wants but there are limitations based on the structure of the system under control. With the same explanation, the third term showing the rate of change for the control input is required to take into account the limitations of the actuators in providing the required change in control inputs in a short amount of time.

In general, MPC is an optimization problem in a prediction window while considering real-world limitations to provide the optimal control inputs while respecting the structure of the controlled system. In addition, future information is taken into account to modify the control inputs accordingly which is a distinctive feature of MPC compared to classical controllers such as PI, PID, PD, etc. Researchers also investigated possible extensions for MPC using data-driven methods, AI, robust control, etc., to make it even more adaptable for complicated cases where there are

model mismatches or inaccurate information.

2.4 Quality of Experience

Video content is the highest internet traffic that users watch or share through different websites such as Netflix, YouTube, Instagram, etc. Also, consider video conferences and live broadcasting of live events through various platforms. According to the statistics, by 2018, video data is 79% of the global Internet traffic demonstrating the huge volume of data regarding videos. Moreover, video content reaches people in a wide range of ages from a two-year-old kid up to a ninety-year-old man which shows how all the people with different ages and features are being affected by this video data. Consider a football match between two great clubs or countries which is broadcast through the Internet, and a high number of users want to watch this match at the same time. How can this number of users be managed to provide data to all of them with the same quality, without any delay, and without pauses in the video? This is a challenge that made researchers and engineers work to propose various approaches to tackle it. Control theory together with networked control systems are considered as one of the methods to deal with this problem.

QoE refers to the level of satisfaction perceived by users when consuming video content, which directly influences their decision to continue or discontinue watching. Although various models have been proposed to quantify this metric in a general and consistent manner, achieving a universally applicable formulation remains challenging due to the inherent variability in individual user preferences. As an illustrative example, consider a child watching a cartoon on YouTube. If the video frequently stalls due to limited network bandwidth, the child may become frustrated and stop watching, while another child in a different context might tolerate the same interruptions and continue viewing. Similarly, a video streamed at a relatively low resolution, such as 480p, may be acceptable to one user, whereas another user may find any qual-

ity below 1080p unsatisfactory and choose not to watch it at all.

These examples highlight that, although objective metrics can be used to quantify video quality and compare different streaming conditions, user decisions are not always aligned with such standardized measures. Instead, they are strongly influenced by subjective factors, including personal expectations, context of use, and individual tolerance to impairments. Consequently, modeling QoE requires not only objective assessment of system performance but also consideration of the diverse and inherently subjective nature of user perception.

Practically, video data is delivered to the clients after a series of actions. First, a video is uploaded to an Internet server to be broadcast. Then, the video is divided into several chunks with a fixed duration. These chunks are later compressed to several resolutions and encoding bitrate which are called *Levels*. The server sends these video chunks through a connection network to the clients or other destinations. This network includes several nodes being utilized to transfer the information based on the location of the sender and receiver. At the client side, there is a buffer which absorbs the mismatch between the download rate and the play rate by storing the chunks inside to be served later. If the buffer becomes full, then the packets that are received afterward will be discarded. This phenomenon is called *Packet Loss* that will lead to issues with the streamed video. On the other hand, if the buffer becomes empty, then the played video will be paused. This issue is called *Rebuffering*, which negatively affects the QoE, and denotes a lack of synchrony between sender and receiver. An empty buffer may result from insufficient available bandwidth to download video segments at the required rate. Conversely, a full buffer can arise from underutilization of the available network capacity, where, despite high bandwidth conditions, lower-quality video segments are downloaded instead of selecting higher-quality ones that better match the network capability. Researchers tried to model this issue mathematically to leverage control theory with the aim of producing a feedback control for the client side asking the level of the video which satisfies the

required features. A model for this problem is proposed as:

$$\dot{q}(t) = \frac{r(t)}{l(t)} - d(t) \tag{2.8}$$

where $q(t)$, $r(t)$, $l(t)$, and $d(t)$ denote the buffer queue size, download rate, selected quality level, and draining rate (i.e., playback rate), respectively. The evolution of the buffer queue is influenced by the download rate, the chosen video quality level, and the rate at which the buffer is depleted, namely how quickly the client plays the video. In general, $d(t)$ is considered as 1 since at each second, one second of the video is played on the user's screen. The playback speed can be increased to, for example, $\times 2$, in which case $d(t) = 2$, meaning that two seconds of video are consumed for every second of real time. In order to adjust the variables in a way that the rebuffering and underutilization are prevented, the feedback control would be used on the client side in the controller. This controller adaptively changes the video level ($l(t)$) to keep the balance in the size of the queue for a proper display which is called Adaptive BitRate (ABR). There are two kinds of ABR algorithms including i) Classical ABR, and ii) QoE-based ABR.

2.4.1 Classical ABR

In this method, the feedback loop contains the data including buffer length and estimated available bandwidth to the controller to produce the related control input. Classical ABR is also divided into *Rate-Based* and *Buffer-Based* methods where the difference is in the decision variable for the control action. In the Rate-Based method, the decision is taken based on the estimated available bandwidth while in the Buffer-Based method, buffer length determines the suitable decision to be made. Although none of these features are directly linked with the QoE, they improve the data transmission by maximizing the usage of the bandwidth, and preventing the queue from becoming empty. The control diagram of this method is shown in Fig.

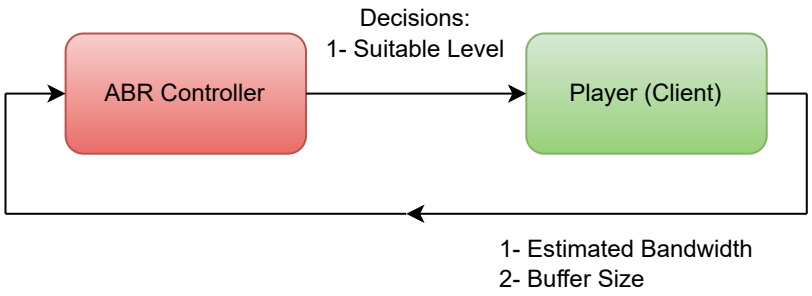


Figure 2.4.1: Control diagram of classical ABR algorithm

2.4.1.

2.4.2 QoE-Based ABR

A QoE-based method aims to maximize the perceived quality of experience by leveraging QoE-related metrics as feedback signals within the control mechanism. QoE-Based ABR includes two methods named *Model-Based* and *Model-Free*. In the Model-Based method, there is an explicit model relating the feedback parameters to a provided QoE model. Using the model, the QoE parameter can be maximized; however, the existing models of the QoE cannot comprehensively describe the related features since they are highly nonlinear as well as changing depending on the person watching the video as mentioned earlier. If a model with an acceptable accuracy can be suggested, the controllers such as MPC can be highly useful due to their optimal behavior while taking into account several practical limitations. Since the model of the QoE is not always available, or it can be inaccurate, and also the user may change his/her opinion over time, a model-free method is proposed to adapt the features of the video and related parameters to overcome these issues. This method uses learning-based algorithms to learn and update the previous models according to the changes that occur in the QoE. The famous learning-based approaches include reinforcement learning, deep learning, and other existing learning-based algorithms. Fig. 2.4.2 illustrates the control diagram of this method.

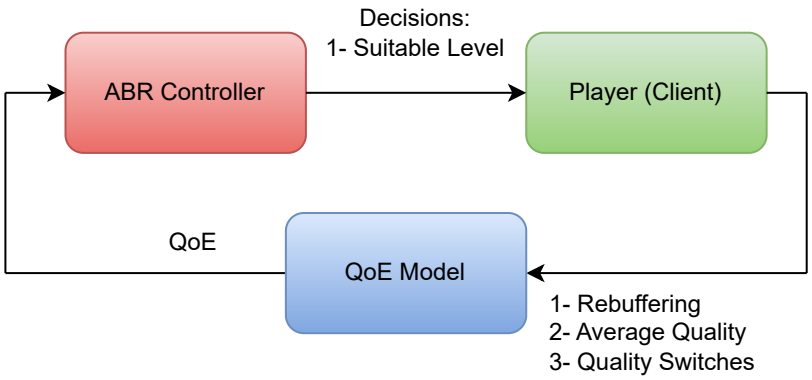


Figure 2.4.2: Control diagram of QoE-based ABR algorithm

As can be found in Fig. 2.4.2, the rebuffering, average quality, and level switches are the variables that are sent through the feedback to the ABR controller to select the best level for the video segments to be downloaded and played. The best possible solution is the one having the highest average quality, a minimum number of switches between the levels to have a smoother play, and without any rebuffering phenomenon to satisfy the users leading to an increase in QoE.

2.5 DC Islanded Grids

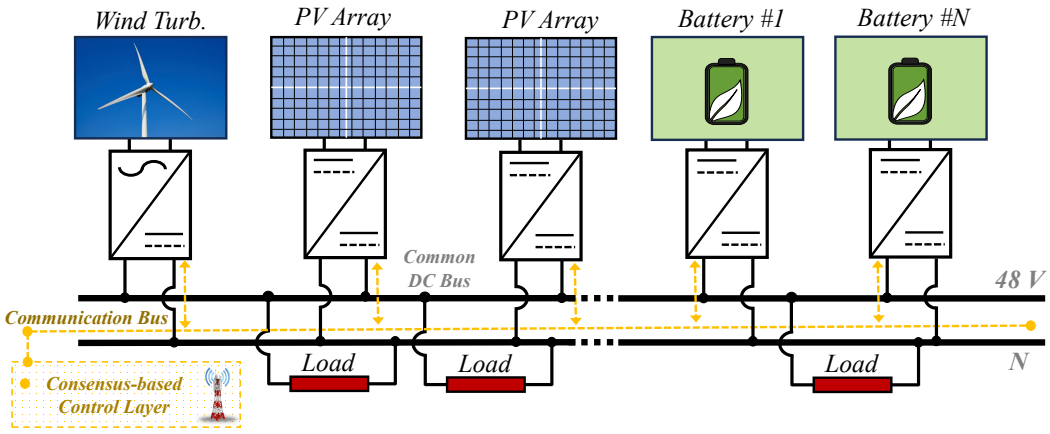


Figure 2.5.1: The components of an islanded grid

In recent years, global warming has become a crucial concern of the modern world which highly depends on the consumption of fossil fuels. Although there are numerous ways to produce green electricity energy, fossil fuels are still the main sources of electricity production in most countries. Due to the fact that technology and the digital world completely rely on electricity, and the main source of this electricity is coming from fossil fuels, the advancements of technology can change the current situation of the earth, making it impossible to live in due to the polluted and warmed world. However, technology cannot be slowed down or even stopped to support the earth; but new, safe, and clean resources can replace fossil fuels. The main green energy resources are wind turbines and solar panels that are currently used in the world to provide the required energy for the consumption of various users.

Smart Grids (SG) and Islanded Grids are hot topics covered by many researchers to propose possible solutions for cities to provide their demanded energy. Traditional power distribution systems transmit electricity in a single direction, from power plants to consumers, and can be affected by various issues, such as faults in transmission lines. Moreover, traditional power transmission does not take into account the demand from consumers in real time, causing loss of demand in peak times and overload in other times, leading to imbalance in electricity production. Due to the fact that the sources of electricity are large-scale nuclear power, coal, gas, and fossil fuels, beforehand planning for energy production is not trivial since there are lots of parameters needed to be adjusted accordingly. This leads to a slow response time in traditional power sharing methods. Faults cannot be predicted beforehand, leading to the occurrence of faults and then their resolution instead of prediction and avoidance. Another advantage is the easy scalability of smart grids by adding several grids and technologies without too many changes and efforts. Usage of smart grids controls the price of electricity in a balanced way for both consumers and producers by exchanging energy evenly. This is highly welcomed by customers since they can also use free energy at certain moments or even gain money by selling their produced

energy to the main grid.

All in all, smart grids provide flexible energy distribution with various advantages which exceed traditional energy management and distribution. Smart grids can be categorized into AC grids and DC grids depending on the grid features. Currently, the AC smart grid is dominant compared to the DC one; however, DC smart grids provide many advantages such as lower required regulations, simpler control design, compatibility with renewable energy resources, lower conversions which save energy, no need to manage reactive power, etc.

Now that the basics of the grids and their advantages have been covered, let us introduce islanded grids. Islanded grids are grids that are not connected to the main grid, which means that they should produce and consume energy all by themselves. This idea is highly beneficial in remote areas that cannot be reached or are hard to be reached by the main grid. Also, specific buildings such as hospitals and military bases must be able to meet their energy demand independently of the main grid due to their delicate nature. Therefore, the design plan for islanded grids must be done accurately since there cannot be any grid compensating for the exceeded or shortage of energy. In general, the structure of an islanded grid consists of renewable energy resources, converters, DC bus lines, loads to be fed, communication channels if needed, and backup storage units such as batteries, supercapacitors, etc. Fig. 2.5.1 shows the structure of an islanded grid where the mentioned components and the related connections have been shown.

In this thesis, we focus on the backup batteries inside the islanded grid to optimize their lifespan by providing control-based algorithms. Since these batteries are expensive, they must perform optimally to prevent their degradation while balancing the demanded energy in the case of mismatched production and consumption.

Chapter 3

Related Work

This chapter summarizes the related scientific research on both the motion control of drones and the synchronization of backup batteries in DC islanded grids. For each topic, a brief introduction is provided, followed by a discussion of related work, with an emphasis on the differences of our approach and the resulting research gap addressed in this thesis. Therefore, in Section 3.1, motion planning of drones for monitoring and coverage tasks is discussed and compared with the state of the art. Later, Section 3.2 discusses the challenges of backup batteries in islanded grids, which have been addressed in the literature, and how our approach proposes a novel method to tackle them.

3.1 Motion Control of Swarm of Drones

3.1.1 Introduction to Swarm Control of Drones

Drones are widely used to perform tasks that are time-consuming, expensive, or impossible for humans such as firefighting [22], remote sensing [23], search and rescue [24], surveillance [25], and object detection [26]. Based on the specific application, drones use different types of sensors such as 3D LiDARs [27], mmWave radars [28], Red Green Blue (RGB), and event-based cameras [29]. The data gathered through

these sensors can be stored at the drones or sent in real time through a wireless network connection such as 4G/5G to a GCS for further processing.

A coordinated swarm of drones can operate in massive missions where it is beyond the capability of one drone or, due to energy and timing issues, a swarm of drones suggests a better solution. For example, tasks related to search and rescue must be done accurately in a short time due to the hazardous situation that people are dealing with. A single drone—even a highly advanced one—is less effective than a group of smaller drones that can disperse over an area and detect people more quickly [30]. As another example, consider the occurrence of a fire in a forest where, with a bit of wind, the fire would spread extremely fast in different directions. Several firefighting drones can stop or at least slow the spread of fire while a super-fast and high-tech drone cannot stop it at all. These examples illustrate the importance of having a swarm of drones for specific situations compared to a single one. To perform the mentioned tasks, communication among drones and the control commands of the related actuators are required to be harmonized. This is one of the challenging parts of swarm control since there should be algorithms for the drones to follow while respecting practical constraints and collision avoidance. As mentioned previously, multi-agent systems are one of the best options to be used for the design in order to coordinate a team of drones. Several methods have been proposed in the literature of multi-agent systems such as consensus control [31], formation control [32], flocking [33], surrounding [8], etc., for drone coordination under different situations.

The main controller for drones varies based on the type of the drone, the scenario being dealt with, the information being shared among the drones if they operate as a swarm, etc. Several control approaches have been proposed by researchers and engineers to control drones, ranging from simple PID [34] controllers (classic control) to reinforcement learning [35] (learning-based control). Fuzzy control [36], robust control [37], optimal control [38], linear quadratic regulator, etc., are all taken into account in the literature of drone control. A simple control diagram of a drone is

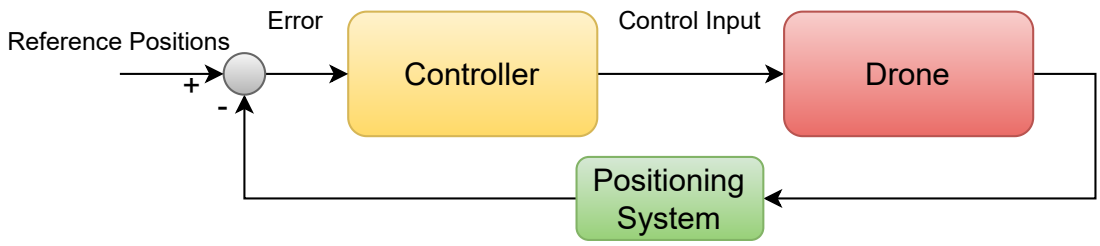


Figure 3.1.1: A simple control diagram of a drone

shown in Fig. 3.1.1, in which any type of controller can be placed in the controller block. One of the advanced controllers capable of providing the desired performance is MPC, which is also utilized in this thesis as the main controller. Since MPC can handle constraints and incorporate information about future system behavior, it is particularly well suited for drone tracking tasks.

3.1.2 Related works for the Motion Control of the Drones

This section provides a summary of the state of the art regarding the swarm of drones covering a given area while capturing videos from above for different purposes. Therefore, this section gathers these works to highlight their strengths and weak points while comparing them with our approach.

Authors in [39] proposed a swarm-based algorithm to detect mobile objects in search and rescue scenarios such as disasters. To do so, a team of drones searches a given area and takes photos of the scene to send to a ground team to respond accordingly by reaching to the area. The quality of the taken images has been taken into account using Similarity Index Measure (SSIM) and Video Quality Metric (VQM) methods. Also, the energy of the drones is considered in the mobility prediction for the next movement. A relay positioning of the drones has been provided in order to preserve the connection for the drones. This work does not consider a specific kind of controller for the drones; plus, the cooperative manner in this paper is not based on the standard multi-agent manner. The available network bandwidth has

not been taken into account, meaning that the drones have no constraint on using the available bandwidth to deliver the photos to the ground control station.

Drones are highly useful for disaster cases due to their maneuverability to monitor the area. [40] considers the problem of maximizing coverage of a given area while minimizing resource utilization. An optimization problem has been proposed where it is solved using Particle Swarm Optimization (PSO). Using the provided method, the drones reach their targets with optimal battery consumption while covering the maximum area. However, it does not mention an explicit controller for the drones, and they do not consider the available network bandwidth to communicate or transport the data. The coverage of the area is done by optimization and therefore, complete knowledge of the area is needed, and it has to be updated by all the agents.

In [41], Bist and Singhal propose a real-time immersive surveillance system utilizing a multi-agent based framework where a network of UAVs, coordinated via an anchor drone and GCS, covers partitioned zones of an inaccessible region. Each drone in the system uses a Raspberry Pi 3B+ as its main onboard controller to manage flight logic, Python-based camera control, and the processing of sensor data. While the core motion planning is based on reaching specific Global Positioning System (GPS) coordinates and executing pan-tilt camera cycles to maximize geometric coverage area, the available network bandwidth is implicitly considered by establishing an operational range limit (approximately 100 meters) based on acceptable packet loss and network latency. Furthermore, video and image qualities are primary considerations of the framework, as the authors evaluate the QoE and Mean Opinion Score (MOS) across various frame rates to ensure that the live feed remains immersive and functional without excessive frame freezing. The inclusion of available network bandwidth is not intended to improve motion control for better visual quality, which marks a difference from our approach. Moreover, the authors have not mentioned the main controller of the drones, but they state that the drones follow the given GPS coordinates. Still, there is a relay drone named Anchor drone which gathers

the information from the drones and sends them to the GCS. The existence of such a drone may lead to a single point of failure, which is not a suitable design for such scenarios.

Choi and Lee in [42] present a bandwidth-aware Coverage Path Planning (CPP) algorithm for large-scale swarms of UAVs performing wide-area searches with real-time video streaming requirements. The system utilizes a multi-agent based framework where a swarm of sensing UAVs is supported by an Aerial Base Station (ABS) that aggregates and relays data to a GCS, which serves as the central entity for command and control. A critical feature of this research is the explicit consideration of available network bandwidth in motion planning; the algorithm optimizes bandwidth allocation to each UAV to determine the specific mission altitude and path width necessary to satisfy communication constraints. Furthermore, video and image qualities are primary considerations, as the framework adjusts the three-dimensional flight paths to guarantee a required Ground Sample Distance (GSD), ensuring that resolution is maintained even when limited bandwidth necessitates lower bitrate. For this work, scalability is a major issue due to the amount of communication for the streamed videos and also for the commands coming back from the central controller. Therefore, one direction to improve this work is making the work decentralized or distributed, and proposing several relay agents to gather the streamed videos. GSD is a metric used in this paper to show the quality of the videos while flying. The formula for the GSD is presented in 3.1. This work does not provide a specific controller for drone control, solves collision avoidance by dividing the area rather than by setting an algorithm for that, uses a relay agent which increases the number of communications, does not provide constraints about energy usage of the drones, and finally, the setup is a centralized one which limits scalability.

$$GSD = \frac{SensorWidth \times WorkingDistance}{FocalLength \times ImageWidth} \tag{3.1}$$

Huang et al. propose a decentralized navigation scheme for a multi-agent based

framework consisting of a UAV network designed for real-time road traffic monitoring and blockage detection in [43]. This work suggests a very useful application of the drones and provides necessary insights into coverage issues related to traffic management. The main controller for each drone utilizes sliding-mode navigation laws to determine the heading and linear speed as control inputs across four distinct operational modes: initial, searching, accumulating, and monitoring. The system requires agents to share positions and sensor measurements to coordinate movement and minimize coverage overlap. The available network bandwidth is not explicitly considered in the motion planning. Furthermore, while the article mentions that the drones' altitude should be selected to ensure sufficient image resolution, specific video or image quality metrics are not integrated into the motion planning or optimization algorithms. One of the strength points of this work is having a decentralized manner to do the monitoring using the information from the neighbor agents.

In [44], the authors present a multi-UAV system for search-and-rescue missions that leverages 5G connectivity for real-time, high-bandwidth communication. The framework includes four guidance algorithms: formation control flying, collision avoidance, cooperative searching, and target tracking. For control, each UAV employs a PID controller within a centralized virtual structure formation scheme. Object tracking is performed using a Convolutional Neural Network (CNN) and two flight modes are supported: manual and autonomous. Two different formations have been considered to provide better coverage in different scenarios. Drones stream the captured videos to a GCS; however, the quality of the videos does not affect the motion of the drones. Also, the available network bandwidth has not been considered in the captured videos or motion planning of the drones.

Zahínos et al. [45] proposed a distributed control framework for multiple cameras mounted on robots to perform area surveillance. The distributed control directs the cameras to capture frames with high information content by minimizing the aggregate information loss per camera pixel. The control method leveraged for the robots is

a decentralized gradient-based control whose goal is the minimization of local cost functions. One of the interesting features of this work is that drones change their altitude to cover the given area in a complete way. For example, if the number of drones changes during the mission, other drones change their altitude to compensate. Although the proposed algorithm is efficient in terms of scalability and compensation of the drones, the network bandwidth and the quality of the captured videos are not considered in the motion planning of the drones.

Relay drones have become a prominent research topic due to their potential in challenging scenarios, such as when a base station is temporarily unavailable or when a large number of users require network connectivity. In cases involving video transmission, drones equipped with streaming capabilities play a crucial role. To address this, Chen et al. [46] investigated the problem of optimal bandwidth and power allocation for drone-based relay networks to ensure high and sustainable user QoE. Their model defines QoE in terms of two key factors–video freeze time and bitrate–while accounting for the channel model and available bandwidth in both the optimization and its constraints. In contrast, our work focuses on a different scenario, where UAVs are allowed to change their altitude to trade detail for coverage when the network bandwidth is scarce. In this work, the motion planning of the drone is not considered; therefore, having a stationary drone, the bandwidth is allocated to different users optimally in order to improve the QoE.

In [47], the authors present a flight path optimization strategy for UAV-mounted base stations operating within heterogeneous networks. The goal is to maximize users' QoE in video streaming scenarios by adapting the UAV's movement based on network conditions and user demands. The authors propose a Q-learning–based algorithm where the UAV acts as an intelligent agent that learns the optimal trajectory across user clusters. The reward function is designed around QoE-related metrics, primarily focusing on minimizing video packet delay, which directly impacts video smoothness and buffer stability. The study highlights the potential of reinforcement

learning for UAV path planning in multimedia applications. However, no area coverage or bandwidth-related path planning techniques are investigated since the goal of this paper is using a relay drone with reinforcement learning to provide high-quality videos for users.

Several studies have addressed the problem of achieving full coverage of a given area through path planning techniques for drones while meeting specific performance criteria. For instance, in [48], an energy-aware path planning approach for a single drone was proposed using a genetic algorithm consisting of a traveling salesman formulation to achieve area coverage while accounting for the drone's energy consumption. This work mainly focuses on the efficient flight of the UAVs with the lowest number of turns by providing path planning based on that. Although this paper does not explicitly discuss cooperative behavior or visual quality, it provides an interesting idea for the coverage of a given area.

The path planning for the coverage of an area using multiple drones in emergency scenarios has been investigated in [49]. The main goals set by the authors are minimizing the completion time of the mission and providing an efficient coverage path for multiple heterogeneous agents. A two-phase method called the Grid-based Coverage Path Planning (G-CPP) algorithm is proposed including grid-based area decomposition and path planning for multiple agents. The application primarily targets maritime search and rescue operations aimed at locating potential survivors. Note that no variation in the altitude of the drones is introduced since the application is different from ours.

3.2 DC Islanded Grids

3.2.1 Introduction of DC Islanded Grids

The increasing utilization of RES, especially solar and wind, is transforming power systems from centralized generation toward decentralized architectures. In this con-

text, DC Microgrids (DC-MGs) have emerged as a promising paradigm for the efficient integration of components with inherently DC characteristics, such as most RESs and loads, as well as Energy Storage Systems (ESS)s [50]. Fig. 2.5.1 illustrates a generic structure of an isolated DC-MGs. Isolated DC microgrids can be found in applications such as remote telecommunication sites, data centers, off-grid residential or industrial areas, and so on [51]. This paradigm reduces energy losses by eliminating multiple conversion stages. Moreover, the absence of frequency, reactive power, and grid synchronization requirements simplifies DC-MGs control [52].

In modern stand-alone DC-MGs that rely on intermittent RESs, maintaining reliable operation remains a primary challenge. The inherent variability of these sources can exacerbate the mismatch between supply and demand [51]. Since the system's power balance primarily affects the common bus voltage, this equilibrium must be maintained in real time to keep the bus voltage close to its nominal value [53]. To address this issue, Battery Energy Storage System (BESS)s, owing to their capability for bidirectional energy flow, represent a great solution to act as backup units and support the microgrid, thereby enabling an autonomous entity.

When multiple BESSs are connected in parallel, it is desirable for batteries with different capacities and initial State of Charge (SoC) levels to contribute to power compensation according to their specific characteristics. Following this, their SoC differences can be equalized over time [54]. Achieving such SoC balancing is desired to minimize the depth of discharge of individual batteries and to prevent overcharging or deep discharging [53]. Consequently, this decreases the degradation of the BESSs and ensures a longer service life [55].

Another aspect to be considered is the energy generation units. When multiple generation units (e.g., PVs and wind turbines) operate within the DC-MGs, it is desirable to perform in Maximum Power Point Tracking (MPPT) mode to harvest all available power. However, this is not always desired in isolated grids since it can lead to an excess of produced energy and potential overcharging of the batteries [56].

Therefore, in such cases with fully charged batteries, there should be a mode to curtail the power and manage the supply. To this end, various ratings of RESs should also be taken into account in determining their participation in power provision [55]. It can be concluded that, besides voltage regulation, SoC management of battery packs and accurate current/power sharing among in-service units are crucial for safe and reliable operation of the microgrid [54].

3.2.2 Related Work for the Islanded Grids

Authors in [57] propose a consensus tracking control method for managing the energy and SoC balancing of BESS within grid-connected AC microgrids. The proposed method adds a target power term to a consensus average control law, allowing batteries with different capacities and initial SoCs to follow a variable reference load curve. The strategy uses linearized state-space battery models and applies the linear quadratic regulator method to design feedback gain matrices that ensure system stability while minimizing the tracking errors. Individual battery agents share their local SoCs and power state vectors with neighboring units to reach consensus accordingly. The batteries have a target/leader agent which is different from our case where they work leaderless, and there is not any target value to reach for SoC but to meet the demanded energy and reach the SoCs to a consensus point which is time-varying.

The [58] introduces a distributed control algorithm designed for DC microgrids to achieve proportional current sharing and voltage regulation. The strategy uses a consensus-like algorithm to distribute the current load proportionally to each unit's capacity and employs sliding mode control to ensure the system reaches a stable state (manifold) in a finite time, regardless of initial conditions. To reach consensus the storage units share their provided currents with other batteries, and the method claims the prevention of circular currents among the batteries.

A distributed cooperative control scheme for Energy Storage Unit (ESU)s in DC microgrids has been proposed in [59]. It features a primary control level that dy-

namically adjusts droop coefficients based on SoC and a secondary control level that exchanges power information to ensure precise power distribution and bus voltage recovery. SoC information is being shared among the agents to achieve consensus and also to adjust the droop control coefficients. So, this paper considered both primary and secondary control structures in its design where consensus occurs at the secondary control level.

In [60], the authors propose a secondary distributed control approach to restore nominal voltage levels and properly distribute current among converters in islanded DC microgrids. It uses a "leader node" to guide voltage adjustments and implements correction terms to ensure precise current allocation between units even when line resistance varies. To reach consensus, agents share voltage tracking errors and current ratios (output current divided by rating) with their neighbors.

To sum up, the mentioned papers provide consensus algorithms to manage the states of charge of the storage units in islanded grids while meeting the demanded current. Different control methods have been used as the secondary or primary control levels. The main idea outlined in this work is providing a precise model of the converter for the batteries in which the reference currents and the real battery currents are involved. Details will be discussed in section 8.1.5.

Part I

Motion Planning for Drones

Chapter 4

Swarm Control with Predefined Path

4.1 Rectangle Area Coverage with a Predefined Path

In this section, we describe the scenario and the control goals to be achieved with a rectangular area to be covered with a predefined path. Let us consider a swarm of drones equipped with cameras – such as the one depicted in Figure 4.1.1 – that are required to track a predefined path and monitor an area (e.g. a field, town, etc.) for a specific purpose such as firefighting, border patrolling, agricultural applications, surveillance, object detection, etc. The camera mounted on each drone is supposed to point vertically to the ground and scan a square area whose width depends on the drone's altitude. The video content captured by the cameras is then sent to a GCS using a network connection such as a 5G link. Such a station is in charge of receiving in real-time the video flows sent from all the agents and stitching together the frames to obtain an overall video scanning the area beneath the drones.

Stitching algorithms between two or more video frames usually require a certain

amount of overlap, i.e. a scanned area in common, and high-quality videos [61, 62]. In fact, the first step of such algorithms is to perform feature detection in the frames, i.e. common feature points in the videos coming from different agents. For this purpose, the received videos need to be as high-quality as possible, otherwise, stitching techniques based on image features extraction might fail. In addition, since a certain overlap is required among videos, the drones should fly close together to provide the overlapped videos while avoiding the collisions and maximizing the covered area.

From a practical viewpoint, the drones are provided with an Internet connection to communicate with each other and with the base station to send the captured videos. Therefore, each drone is assumed to be able to exchange information with all the other drones according to a complete directed graph. Furthermore, in a realistic scenario, the available network bandwidth depends on the drone's location in the area and on the number of users using the network bandwidth. Thus, the available network bandwidth is a variable not known beforehand and modeled in this thesis as a measurable disturbance. As such, the available network bandwidth can be measured using both passive and active bandwidth estimation techniques [63].

Due to the variable bandwidth, each drone sends the video with varying visual quality. For this reason, when the available network bandwidth is low, it is desirable for the drone to fly in lower altitude, getting closer to the ground, in order to capture more detailed frames. On the contrary, when the bandwidth is high, the drone can increase its altitude to increase the coverage while at the same time maintaining a reasonable level of visual quality. A useful parameter that measures the size in meters of a pixel acquired by a camera is the GSD, which is a central feature in UAV-based photogrammetry [64].

At the same time, the swarm is supposed to keep a wide area coverage. To this end, it would be desirable for drones to fly as distant as possible from each other but without losing the overlap among the corresponding scanned areas. In addition, to

maximize coverage, they would also fly at the maximum altitude in order to get the widest scanned area possible.

In the described scenario, each drone has two contrasting goals: i) provide a good percentage of overlap among the videos and the best visual quality for each video captured and ii) increase coverage of the region of interest. To achieve the first goal, each drone would keep a relatively low altitude, which consequently implies a decrease in the GSD and an increase in the visual quality. On the other hand, the coverage is improved when the altitude is increased since the cameras mounted on the drones can capture a wider region. Therefore, there is a trade-off between the area coverage and QoE of the captured videos.

Moreover, to cover the area, a leader drone is chosen to track a predefined path while the other agents are simply required to catch up with the leader keeping a safe distance both from the leader and themselves. Notice that the altitude of each drone changes as they move on according to the measured available network bandwidth. The leader-follower framework is needed if the swarm of drones is not sufficient to cover the area completely, which is the more common case. In such scenarios, the drones need either to know the area and follow a predefined path or to plan a path to follow. This creates the need for a leader agent with global information about the environment and the mission objectives. A leaderless framework can be applied in scenarios where each drone is assigned a specific coverage area to follow a repetitive path pattern or perform path planning independently. However, this approach is costly, since each drone requires high computational capability and global knowledge of the environment.

4.2 NMPC for Predefined Path - Rectangle Area

Let us consider the agents' discrete-time dynamics as a first-order integrator as shown in (4.1).

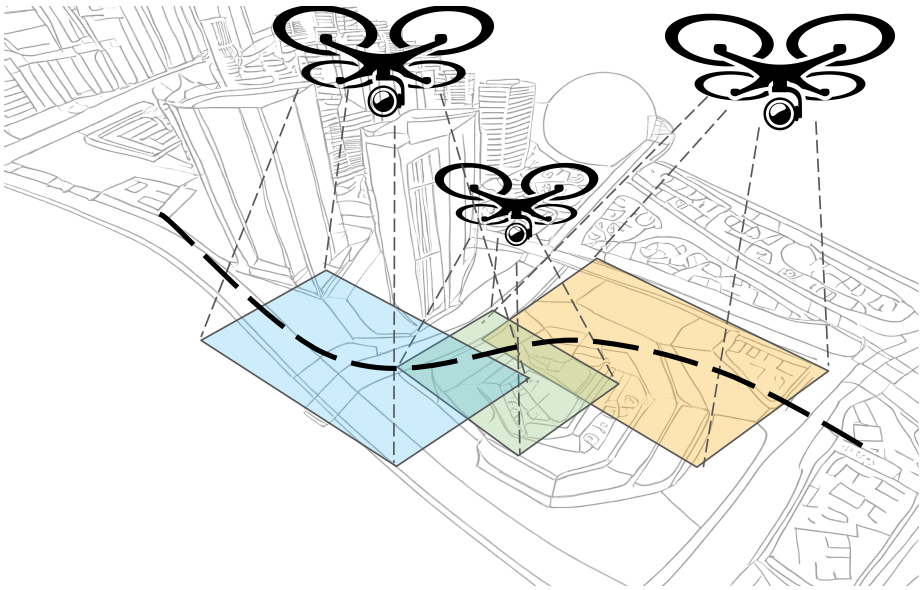


Figure 4.1.1: The leader drone follows a predefined path and the rest of the swarm are following the leader.

$$s_i(k+1) = s_i(k) + T_s u_i(k), \quad i = 1, \ldots, N \tag{4.1}$$

where $s_i = (x_i, y_i, z_i)^T$ represents the vector of the 3D coordinates of agent i, T_s is the sampling time, N is the number of agents, and $u_i = (u_{x,i}, u_{y,i}, u_{z,i})$ is the control input vector for agent i. The Nonlinear Model Predictive Control (NMPC) formulation for each agent is the following:

$$\begin{aligned}
\underset{u_i}{\text{Minimize}} \quad & \mathcal{J}_i && (4.2)\\
\text{subject to} \quad & s_i(k+1) = s_i(k) + T_s u_i(k) && (4.3)\\
& z_{\min} \leq z_i \leq z_{\max} && (4.4)\\
& \|X_i(t) - X_j(t)\|_2 \geq 2R_{\text{safe}} && (4.5)\\
& |u_i| \leq u_{\max} && (4.6)\\
& |\Delta u_i| \leq \Delta u_{\max} && (4.7)
\end{aligned}$$

where the cost function $\mathcal{J}_i$ assumes a structure that depends on the nature of the agent, i.e. leader or follower, and on the state of the swarm as discussed in section 4.3. For the practical view of the problem, several constraints have been defined as follows. Eq. (4.3) represents the system dynamics, Eq. (4.4) limits the flight altitude, Eq. (4.5) is set to avoid collisions where, $X_i = (x_i, y_i)^T$ is the 2D position of the i-th agent and R_{safe} defines the safety distance that the drones must always keep with the other drones, Eq. (4.6) limits the control input, and Eq. (4.7) limits the changes in the control input. All these constraints are dependent on the specific application meaning that they can change accordingly.

Note that the leader, from now on identified by index $i = 1$, tracks the reference path (computed by the path planner or a predefined path) neglecting the collision avoidance constraint (4.5), which is left in the NMPC of the followers to keep a safe distance from their neighbors including the leader agent.

Without loss of generality, we consider the area covered by each drone to be a square, whose dimensions are determined by the drone's altitude. The reason for this choice is the camera structure which always provides a rectangular (or square) shape for the captured video; therefore, to mimic what the drone sees, the square shape has been chosen as the coverage area of each drone. That is why the norm infinity has been chosen for the overlapping cost in Eq. (4.11). The relationship between the altitude of agent z_i and the side length L_i of the square defining its scanned region is as in Eq. (4.8), where ϕ denotes the camera's angular aperture [43].

$$L_i = 2z_i \tan(\phi/2) \tag{4.8}$$

There are cases in which the overlap between the agents decreases or is lost, thus generating a detachment. To avoid this detachment, the swarm switches between two possible states including the *Tracking State* and *Recovery State*.

4.3 Tracking and Recovery States

In the proposed scenario, we consider two situational states for the drones' motion named i) *Tracking State (T)*, and ii) *Recovery State (S)*. The reason for having these two states is to have the team moving together with the leader to avoid any kind of detachment among the captured videos. Due to the fact that MPC relies on solving an optimization problem, there are cases where the solution at certain time-steps cannot reach the optimal solution and instead follows a suboptimal solution. Based on the definition of the optimization problem, following the leader can be neglected in certain scenarios since the trade-off cost for the drones makes them stand still with the rest of the swarm instead of following the leader. As a result, the leader moves and the others stop, causing detachments among the received videos. In these situations, we stop the leader waiting for the others, and the others have to reach the leader as soon as possible. These suboptimal cases made us propose two moving steps for the drones.

Tracking State. The tracking state is for when the movement of the drones has overlapped Field of View (FoV) without any risk of detachment. Therefore, the leader follows the path, optimizes its control input, and tracks the reference flight altitude. The followers track the leader, optimize their control input, and track the reference flight altitude.

Recovery State. In this state, the distances among the drones initiate a risk of detachment among the captured videos. To avoid detachment, the leader stops moving waiting for the other agents to catch it, tracks the reference flight altitude, and optimizes its control input. On the other hand, the other agents track the leader with a higher optimization weight, regroup with the rest of the drones as a new optimization term which is added to their optimization cost function only in recovery state, track the reference flight altitude, and use their maximum possible control input to move reducing the risk of detachments.

4.4 Cost Functions for NMPC

The cost function adopted by the leader in the NMPC for tracking state (T) is illustrated in Eq. (4.9), and for the rest of the followers is as in Eq. (4.10):

$$\mathcal{J}_{1,T} = w_p J_p + w_z J_z + w_u J_u + w_{\Delta u} J_{\Delta u}, \tag{4.9}$$

$$\mathcal{J}_{i,T} = w_l J_l + w_z J_z + w_u J_u + w_{\Delta u} J_{\Delta u}, \tag{4.10}$$

$$J_p = \sum_{k=1}^{N_p} \left\| X_i(t+k|t) - \bar{X}(t+k) \right\|_\infty \tag{4.11}$$

$$J_z = \sum_{k=1}^{N_p} \left\| z_i(t+k|t) - \bar{z}_i(t+k) \right\|_2 \tag{4.12}$$

$$J_u = \sum_{k=1}^{N_c} \left\| u_i(t+k-1) \right\|_2 \tag{4.13}$$

$$J_{\Delta u} = \sum_{k=1}^{N_c} \left\| \Delta u_i(t+k-1) \right\|_2 \tag{4.14}$$

$$J_l = \sum_{k=1}^{N_p} \left(\left\| X_i(t+k|t) - X_1(t+k) \right\|_\infty - r(R_i + R_1) \right)^2 \tag{4.15}$$

In equations (4.11) to (4.15), N_p is the prediction horizon and N_c is the control horizon considered in the NMPC controller. J_p is the cost related to the path tracking. In particular, $X_1(t)$ is the 2D location of the leader agent at time t, whereas $\bar{X}(t)$ is the reference planar location produced by a predefined path or an online path planner at time t. J_z is associated to the altitude tracking. In particular, $z_1(t)$ denotes the leader's altitude and $\bar{z}_1(t)$ is the leader's reference altitude at time t computed according to the available network bandwidth. J_u is the cost for the

control effort and $J_{\Delta u}$ is the cost related to the variation of the control input. The term J_l makes agent i follow the leader. R_i is half the side of the square FoV of agent i at a certain time-step. By setting $0 < r < 1$, the agents strive to get an overlap with the leader.

In Eq (4.9) and Eq (4.10), w_p, w_z, w_u, $w_{\Delta u}$, w_l are weights to be properly tuned.

For the recovery state (S), as mentioned in section 4.3, the leader stops following the reference path waiting for the others to reach; therefore, the cost function related to the leader loses the term J_p responsible for the path tracking as follows:

$$\mathcal{J}_{1,S} = w_z J_z + w_u J_u + w_{\Delta u} J_{\Delta u} \tag{4.16}$$

Regarding the followers in recovery state, the cost function does not penalize the control input and its changes allowing full recovery from this state as soon as possible; therefore, the cost function for the followers is as in Eq. (4.17):

$$\mathcal{J}_{i,S} = w_l J_l + w_z J_z + w_f J_f \tag{4.17}$$

$$J_f = \sum_{j \in \mathcal{N}_i} (\sum_{k=1}^{N_p} (\|X_i(t+k|t) - X_j(t+k)\|_\infty - r(R_i + R_j))^2) \tag{4.18}$$

where J_f represents the term responsible for keeping the agents together, ensuring a desired level of overlap between their scanned areas. As in the tracking state (T), w_z, w_u, $w_{\Delta u}$, w_l, and w_f are weights to be properly tuned for the recovery state (S).

Considering a fully connected directed graph for J_f will lead to poor altitude tracking. To avoid this, we adopt a new graph which is named *Actuation Graph*. Actuation graph is a graph where the number of connections is lower than the fully connected graph leading to a faster computation and lower demanded communications. So, the neighbors inside this graph will be used to compute the overlap among the drones. For this new graph, and therefore for J_f, let $\mathcal{N}_i = \{j \in \mathcal{V} : (j,i) \in \mathcal{E}\}$, where $G(\mathcal{V}, \mathcal{E})$ is the actuation graph with the minimum amount of connections

shown in Fig. 4.4.1. The reason why this graph has been chosen is to have the minimum amount of connections among the agents, and to have a direct connection to the leader for all the agents to follow the leader. Similarly to Eq (4.15), the overlap between the agents' scanned regions is controlled by the term $r(R_i + R_j)$, where $0 < r < 1$ specifies the desired overlap ratio. A value of r close to 1 results in minimal overlap, whereas decreasing r causes the agents to move closer together, thereby increasing the overlap.

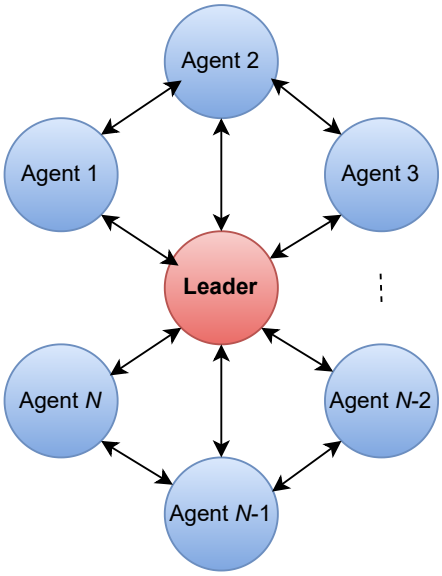


Figure 4.4.1: The actuation graph among the agents.

4.5 Correlation Between Bandwidth and altitude

To determine $\bar{z}_i$ in Eq. (4.12) for each agent, we use the following monotonically increasing function to build a relationship between the estimated available bandwidth b and $\bar{z}_i$.

Table 4.1: Values of the weights in the cost functions of the NMPCs

	w_l	w_p	w_f	w_h	w_u	$w_{\Delta u}$
$\mathcal{J}_{1,T}$	1.5	-	-	5	0.001	0.001
$\mathcal{J}_{i,T}$	-	0.5	-	1.5	0.001	0.001
$\mathcal{J}_{1,S}$	-	-	-	0.5	0.001	0.001
$\mathcal{J}_{i,S}$	1.5	-	0.3	1.5	-	-

$$\bar{z}_i(b) = \begin{cases} z_{\min} & b < b_{\min} \\ S \cdot b + z_{\max} - S \cdot b_{\max} & b_{\min} \leq b \leq b_{\max} \\ z_{\max} & b > b_{\max} \end{cases} \tag{4.19}$$

where S is defined as $S = (z_{\max} - z_{\min})/(b_{\max} - b_{\min})$. Notice that this function could be replaced by any other mapping able to describe the relationship between altitude, visual quality, and available bandwidth more effectively.

4.6 Simulation Results for Predefined Path - Rectangle Area

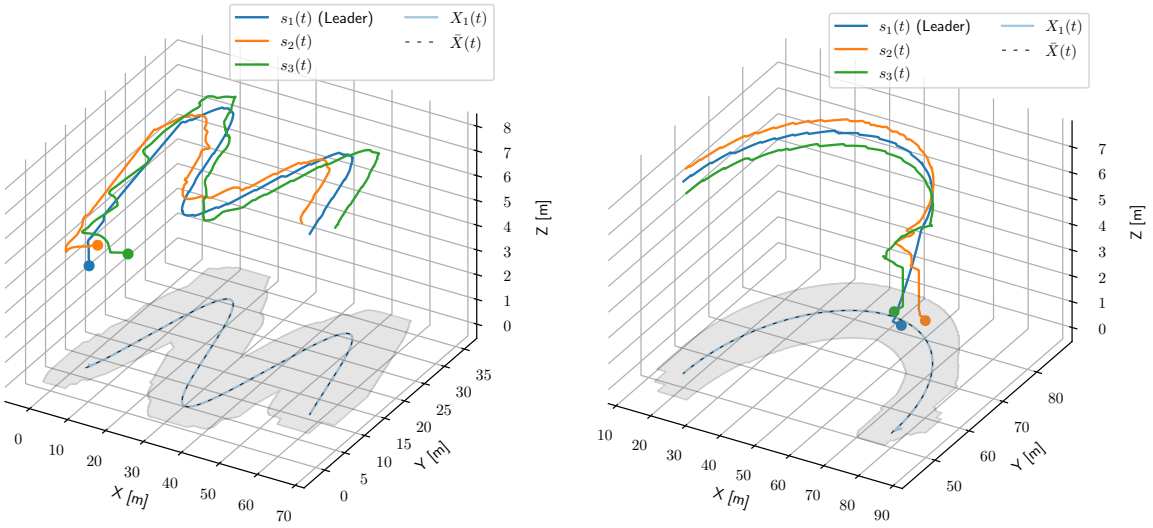
In this section, simulation results for the provided methodology have been illustrated to show the coverage of the swarm, trajectory of each drone, and altitude tracking of the drones. In particular, we discuss the results obtained for $N \in \{3, 5, 6\}$, where N is the number of agents, and for three possible predefined paths: a Sine-like, a Semicircle-like and a Road-like shape, which aims at simulating a road. Moreover, we have set the values for the weights in the cost functions of the MPC as in Table 4.1.

Then, to run the NMPC, we set $r = 0.4$, $N_p = 5$, $N_c = 2$, $R_{\text{safe}} = 2m$, $h_{\min} = 4m$, and $h_{\max} = 10m$. Finally, the upper and lower bounds for $\Delta u_x, \Delta u_y, \Delta u_z$ have been set to an absolute value of $0.1m/s^2$ for all agents. Notably, the velocity of the leader is intentionally set lower than that of the follower agents. This allows the swarm to

maintain formation, ensuring that the followers can keep pace with the leader. For this reason, $u_{x,\max} = 0.7m/s, u_{y,\max} = 0.7m/s$ in Eq. (4.6) for the leader, whereas $u_{x,\max} = 1m/s, u_{y,\max} = 1m/s$ for the other agents. Regarding $u_{z,\max}$, it has been set to $0.1m/s$ for all the agents to avoid abrupt changes in altitude and, therefore, in terms of covered area.

Concerning the available network bandwidth, we set it by drawing from a uniform random distribution of bandwidth values over the area to be scanned. Such values are then variable in space but assumed to be constant in time and altitude. This distribution has been employed to run all the following simulations for fair comparisons and considerations. Notice that the choice of generating random bandwidth between a minimum value of 500 kbit/s and a maximum value of 5000 kbit/s defines a realistic bandwidth profile due to its unpredictability. Moreover, to the best of our knowledge, no dataset providing measurements of wireless channels as a function of the geographical position of a drone exists.

Let us start by considering the case $N = 3$, where, as shown in Fig. 4.6.1(a) and Fig. 4.6.1(b), the leader is required to track a Sine-shaped path and a Semicircle path while the followers surround it. From the figures, it is possible to state that the drones, whose initial positions are denoted with round markers, strive to follow the leader in space while sticking together. On the x, y-plane, the reference path for the leader (black dashed line) and the leader's trajectory projection on this plane (light blue line) are depicted, thus showing good performance in terms of tracking. Moreover, this plane also displays in gray the total area scanned by the swarm. The z-axis represents the altitude of the drones, which is shown to change along the path due to its dependence on the bandwidth.

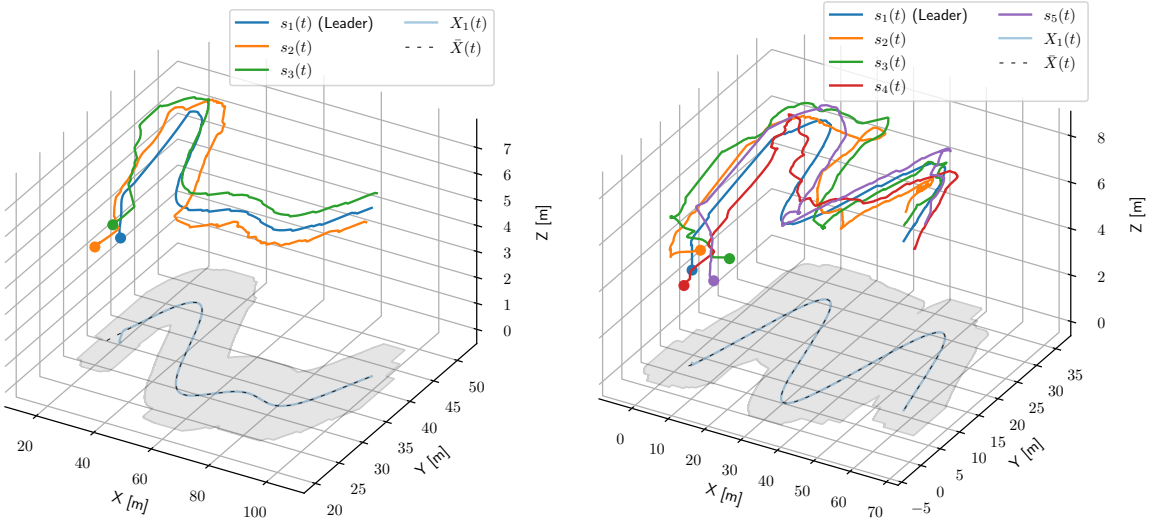


(a) $N = 3$ for Sine-like path.

(b) $N = 3$ for Semicircle-like path.

Figure 4.6.1: Drones' trajectories and area coverage for $N = 3$.

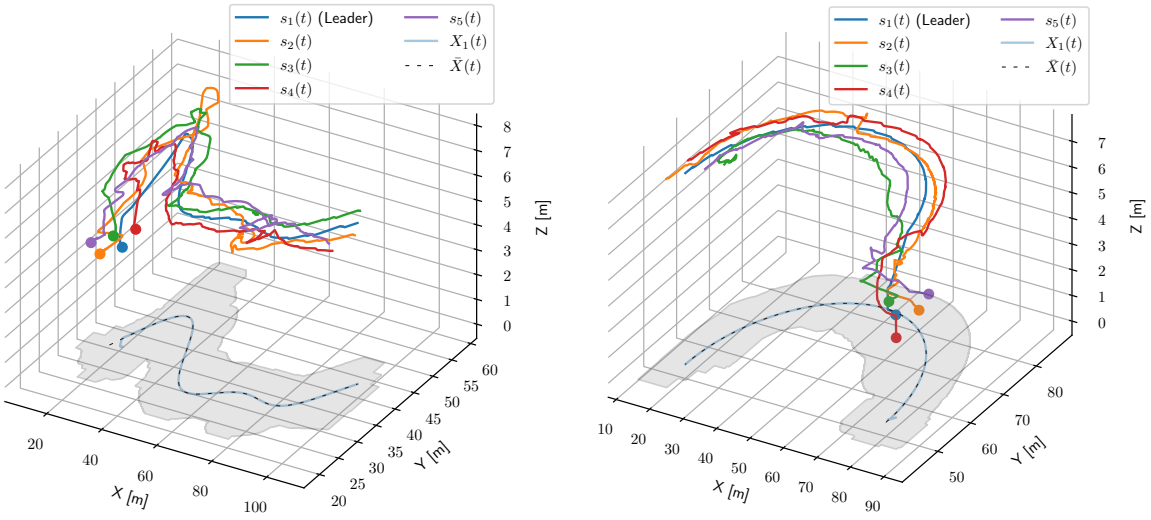
The visual comparison is presented for different numbers of drones and varying paths in Fig. 4.6.2 to Fig. 4.6.5.



(a) $N = 3$ for Road-like path.

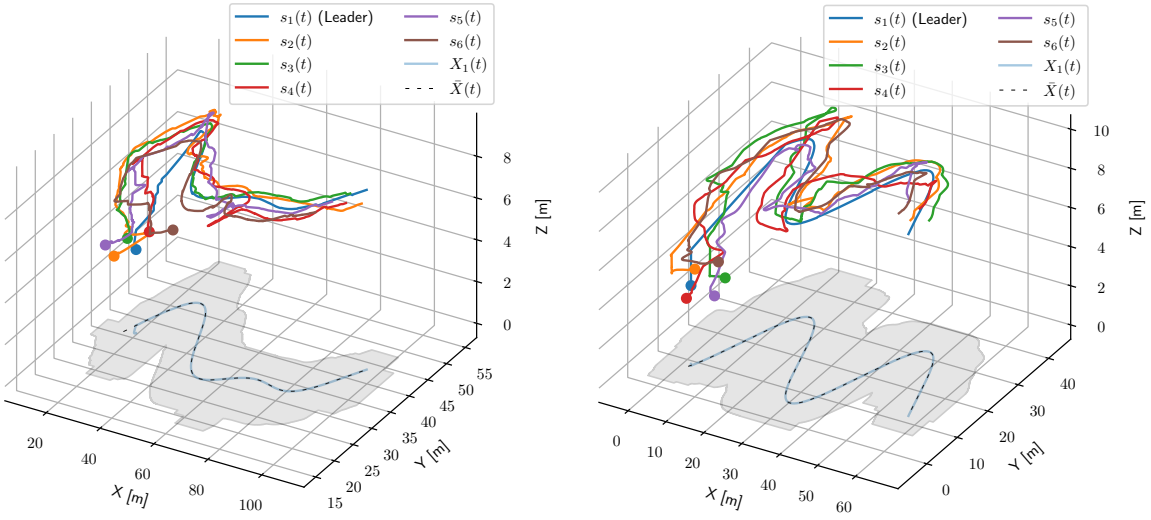
(b) $N = 5$ for Sine-like path.

Figure 4.6.2: Drones' trajectories and area covered for $N = 3$ and $N = 5$



(a) $N = 5$ for Road-like path. (b) $N = 5$ for Semicircle-like path.

Figure 4.6.3: Drones' trajectories and area covered for $N = 5$



(a) $N = 6$ for Road-like path. (b) $N = 6$ for Sine-like path.

Figure 4.6.4: Drones' trajectories and area covered for $N = 6$

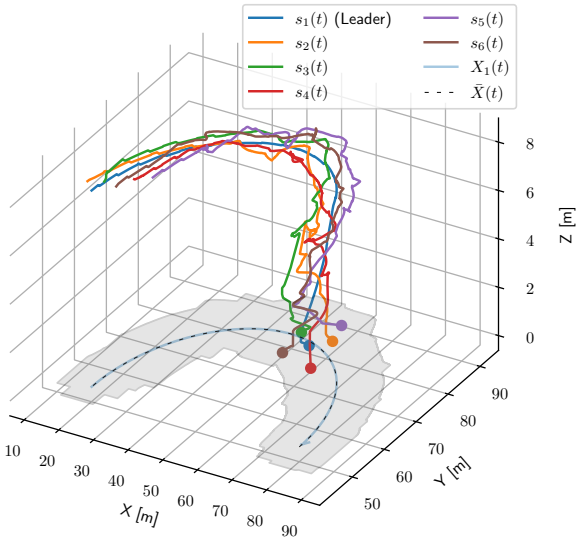


Figure 4.6.5: Drones' trajectories and area covered for $N = 6$ and Semicircle-like path.

Figure 4.6.6 shows the dynamics of the altitude tracking error for the case of $N = 6$ agents. In other words, it can be seen that, starting from the same initial altitude, each agent is able to track its reference altitude almost perfectly after a transient. Notice that, as already explained, each agent has a different reference altitude in that it depends on the particular position of the agent at each time-step and on the available network bandwidth in that point. Similar considerations can be made for the other cases.

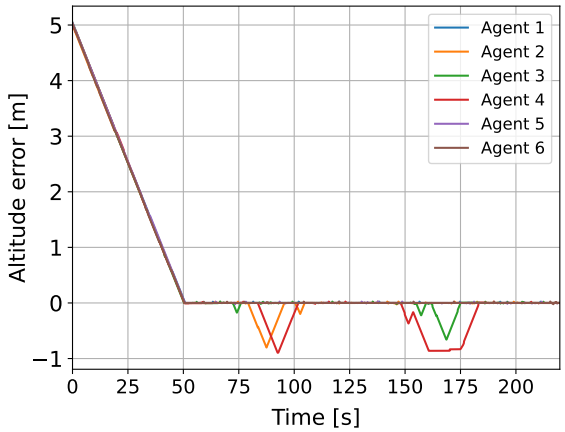


Figure 4.6.6: Tracking error of the altitude for $N = 6$

As expected, increasing the number of agents leads to an increased total scanned

area. In Table 4.2, the total coverage areas produced by the agents for $N \in \{3, 5, 6\}$ are listed. Note that, for each reference path, such areas are normalized with respect to the area obtained when $N = 3$. As expected, the data show a growing trend and the scanned areas experience an increase of about 30 % when passing from $N = 3$ to $N = 6$ for all the three paths.

Let us now carry out a sensitivity analysis for the case of $N = 5$ agents tracking a Sine-like path. To evaluate the efficiency in terms of detachment avoidance, we test the framework by varying the values of $u_{x,\max}$ and $u_{y,\max}$ of the leader from the set $\{0.7, 0.8, 0.95\}$. Subsequently, we are interested in evaluating the time spent by the swarm in the *recovery* state, i.e. when the leader stops tracking the desired path to wait for the other agents to catch up and regroup. The higher this amount of time, the more the efficiency of the framework deteriorates since more time is required to the swarm to complete the task. To the purpose, we define the following *efficiency index*:

$$\eta = 1 - \frac{T_S}{T_{\text{tot}}} \tag{4.20}$$

where T_S is the amount of time spent in *recovery* mode while T_{tot} is the time spent to accomplish the whole mission. Therefore, a higher η implies a better performance and a low amount of fraction of time spent in *recovery* state. Results show that, when the leader's maximum velocity is set to $0.7m/s$, the measured efficiency η is approximately 0.9. Increasing the velocity to $0.8m/s$ results in lowering the efficiency to 0.86, and further increasing it to $0.95m/s$ reduces η to 0.82. This observation suggests, as expected, that allowing the leader a higher velocity to track the reference path increases the likelihood of switching to the *recovery* state.

In other words, the leader will stop more times to wait for the rest of the swarm. Notice that, on the other hand, making the velocity of the leader too low would lead to tracking issues in that the number of samples of the reference path should be increased and the total time to complete the task-even without switching to the

Table 4.2: Normalized total scanned area as a function of N for each considered scenario

	$N = 3$	$N = 5$	$N = 6$
Sine	1.0	1.19	1.37
Road	1.0	1.17	1.28
SemiCircle	1.0	1.26	1.36

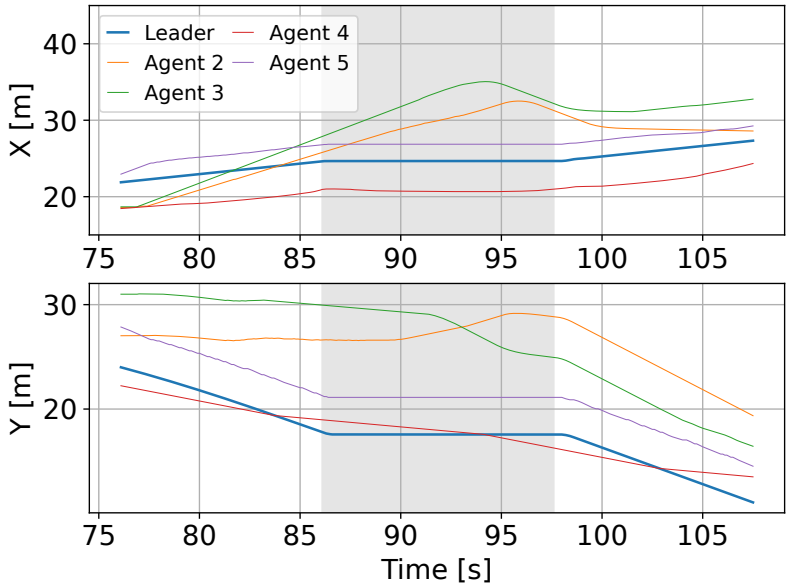


Figure 4.6.7: Agents' coordinates dynamics in the case of a switch between Tracking and Recovery states

recovery state-would increase, too. Therefore, finding an optimal balance in the leader's velocity is crucial for maintaining efficient swarm operation and minimizing the need for recovery interventions.

With this in mind, let us further investigate what happens when the swarm switches from the *tracking* to the *recovery* state and vice versa. During the simulation with $N = 5$ agents tracking a Sine-like path, one of the cases when the swarm enters the *recovery* state occurs in the interval 86–97 seconds. This period is depicted by the shaded gray area in Figure 4.6.7. This figure shows the dynamics of the x and y coordinates of the agents. In the shaded area, that is, when the swarm is in the *recovery* state, it can be observed that the coordinates of the leader do not change

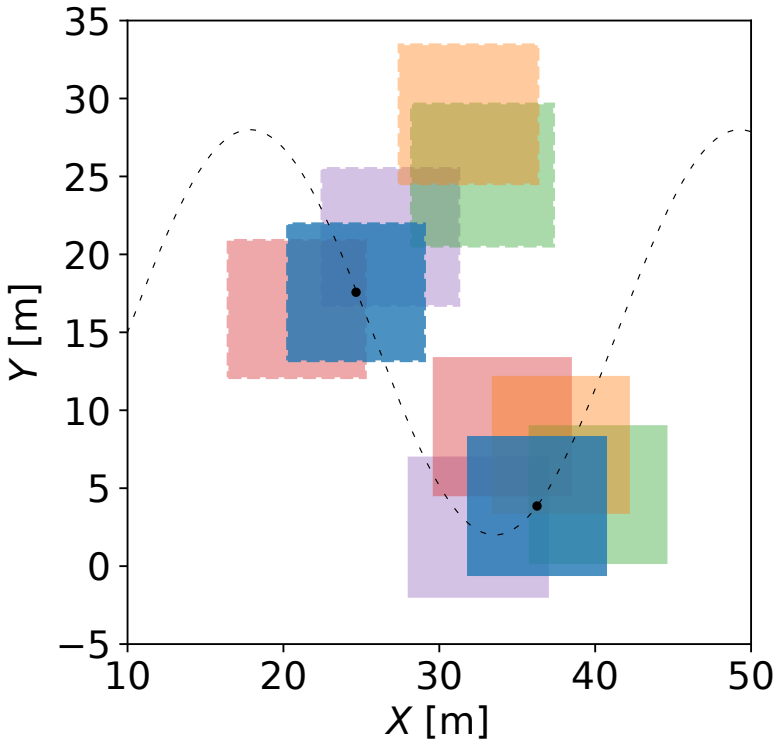


Figure 4.6.8: Areas scanned by the agents at two different time-steps

over time, i.e. the leader has stopped to allow the swarm to regroup.

By looking at the coordinate dynamics, it can be seen that, when in the *recovery* state, the agents are rather far from each other, whereas before or after this switch the coordinates present a converging trend. To better understand this concept, Figure 4.6.8 provides a visual representation of the areas scanned by the drones when in the *recovery* state (dashed squares) and in the *tracking* state (solid-lined squares). In the *recovery* state, the agents are not close enough to each other, increasing the risk of detachment. Therefore, the leader, represented in the figure as a blue square tracking the dashed reference path, does not move until the scanned area of each agent has enough overlap. The solid-lined squares show an example of the swarm in the *tracking* state, thus without risk of detachment, which presents a considerably higher total overlap.

Similar conclusions could be drawn for the case of a different number of agents

or a different reference path. Finally, Table 4.3 reports the measured efficiency index for all the considered cases. In general, when the number of agents increases, the efficiency tends to decrease. This is to be expected since, when the number of agents becomes large, it is harder to move together, thus increasing the possibility of detachment as well as the value of T_S.

Table 4.3: Efficiency η as a function of N for each considered scenario

	$N = 3$	$N = 5$	$N = 6$
Sine	0.94	0.87	0.91
Road	0.90	0.85	0.83
SemiCircle	0.91	0.87	0.88

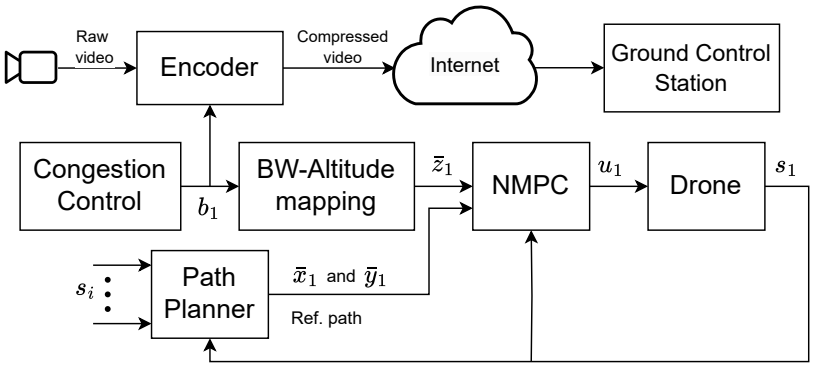
Chapter 5

Online Path Planning

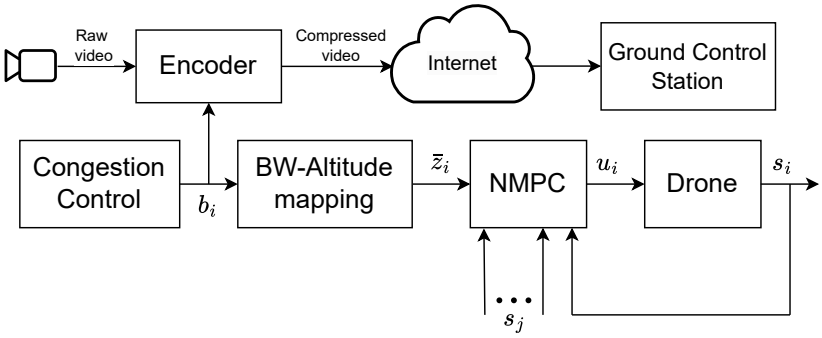
5.1 Rectangle Area with Online Path Planning

Consider the same scenario mentioned in section 4.1, where instead of having a predefined path for the leader drone, the leader does an online path planning based on the information gathered from the other agents' covered area.

To cover the entire area of interest and to actively explore uncovered areas, an online path planner at the leader has been designed. The *Path Planner* in Figure 5.1.1(a), based on the leader's location s_1 and on the other agents' coordinates s_i, $i = 2, \ldots, N$, where $s_i = (x_i, y_i, z_i)^T$ and N is the number of agents, computes the reference point $(\bar{x}_1, \bar{y}_1)$ that the leader has to reach at the next time-step. At this point, based on $(\bar{x}_1, \bar{y}_1)$ and $\bar{z}_1$, namely the coordinates to be tracked on the (x, y) plane and along the z-axis, respectively, the NMPC computes the velocity input u_1 that the leader has to set to get to the reference point. Notice that the path planner, which is inside the leader's architecture, is aware of the other agents' dynamics and locations. Therefore, each drone has two contrasting goals: i) maximize the video quality based on the available bandwidth with a good amount of overlap, and ii) maximize the coverage of the area.



(a) Leader architecture.



(b) Follower architecture.

Figure 5.1.1: Proposed control architecture for the leader and the followers.

Although the followers employ the same procedure to transmit the captured video to the GCS (see Figure 5.1.1(b)), their control inputs are computed differently, since they do not run the path planning algorithm. Based on the reference altitude mapping mentioned in Eq. (4.19), and on the planar coordinates (x_i, y_i) of all the swarm, the NMPC computes the control input u_i.

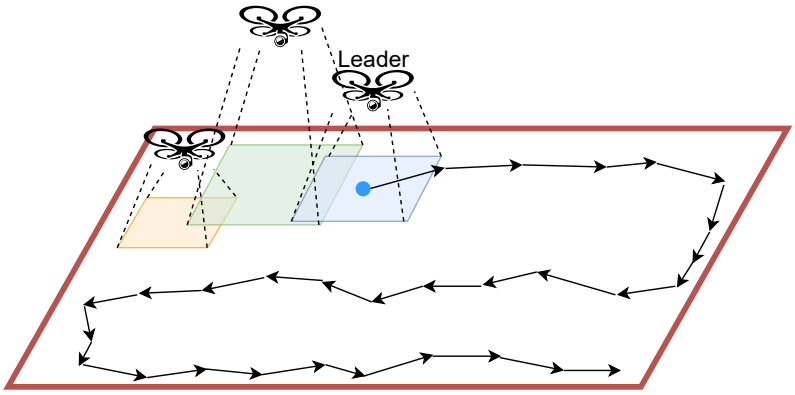


Figure 5.1.2: The leader plans the path to provide maximum coverage and the rest of the swarm are following the leader.

5.2 NMPC for Online Path Planning - Rectangle Area

In order to avoid repetition, in this scenario, we cover only the differences in the methodology with respect to the scenario mentioned in 4.1. So, the same dynamics in Eq. (4.1) are considered for the drones. The recovery and tracking states still apply to the problem as explained in section 4.3. The differences in this part include i) the online path planning for the leader based on the covered area of the swarm of drones, and ii) the constraint regarding the changes of the control input has been omitted from the optimization problem. Therefore, the formulation of the problem will change as follows.

$$\underset{u_i}{\text{Minimize}} \quad \mathcal{J}_i \tag{5.1}$$

$$\text{subject to} \quad s_i(k+1) = s_i(k) + T_s u_i(k) \tag{5.2}$$

$$z_{\min} \leq z_i \leq z_{\max} \tag{5.3}$$

$$\|X_i(t) - X_j(t)\|_2 \geq 2R_{\text{safe}} \tag{5.4}$$

$$|u_i| \leq u_{\max} \tag{5.5}$$

where the parameters are the same as in the previous formulation in section 4.2. No constraints are imposed on the rate of change of the control input, as doing so would increase computational complexity and may lead to infeasibility of the optimization problem. Although this can be considered as a weak point, several papers mention the same problem, that the solution for the optimization becomes infeasible when the number of constraints increases; therefore, the constraint regarding Δu can be omitted. The leader, denoted by $i = 1$, plans a path based on the gathered coverage information of the other agents, and tracks this path while disregarding the constraint (5.4), leaving the task of keeping a safe distance from the leader to the other agents. The same area scanning logic has been taken into account as in Eq. (4.8). The rest of the methodology is the same as stated in section 4.2.

5.3 Online Path planning - Rectangle Area

This section presents the proposed online path planning technique used to achieve coverage of a fixed rectangular area known to the leader of the swarm. Figure 5.3.1 illustrates a scenario with three UAVs, where UAV 1 acts as the leader. Obviously, the total scanning area of the UAVs at each time-step is not constant because it changes in size and shape due to variable altitude that is imposed by the available network bandwidth.

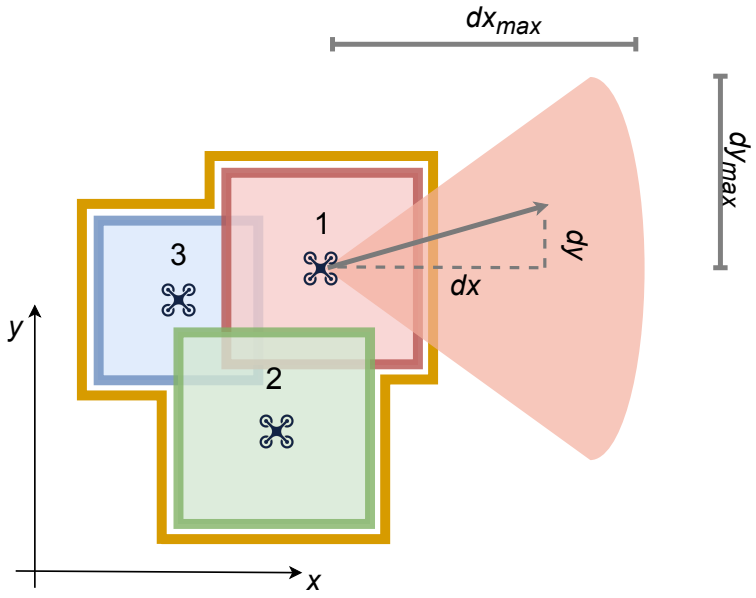


Figure 5.3.1: Formation scheme - online path planning

The proposed path planner is run by the leader and works as follows. Given the initial configuration and scanned area, each UAV communicates its position to the leader. At this point, the leader computes the union of all the scanned areas, denoted as *union shape* (orange shape in Figure 5.3.1), and runs the path planner, which, based on such an area and the map of the environment, computes the next reference point of the path. The leader then runs the NMPC to compute the required velocities to reach that point, while the other UAVs run their own NMPC to follow the leader and stay close together. At the next time-step, all the UAVs communicate their new positions to the leader and the procedure iterates.

It is important to point out that, in a realistic scenario, if a UAV moves from one point to another close point and such a movement is performed in a reasonably short amount of time, the available network bandwidth does not change abruptly.

A widespread approach to achieve coverage when the scanned area is constant is to follow a lawnmower path. Therefore, inspired by this approach, we have designed an adaptive path planner, i.e. a path planner dependent on the variable scanned area. At this point, locating, without loss of generality, the initial formation in a top

left area of the rectangular map, the general approach is to proceed along the x-axis on the right while also adjusting the path along the y-axis to improve coverage and avoid blind spots, i.e. uncovered areas, when the union shape changes. Once the right border is reached, the swarm moves downwards by a certain number of steps. Then, the swarm goes from right to left, and so on.

Concerning the coordinate on the x-axis of the next reference point of the path when going from left to right (right to left), the general lawnmower approach is first evaluated. In other words, the current union shape is projected straight onward along the x-axis on the right (on the left) with zero displacements on the y-axis. The step d_x along the x-axis is computed as shown in Algorithm 1 (Line 1).

Algorithm 1 Computation of d_x and d_y

1: $d_x \leftarrow (1 - 2 \cdot \mathrm{mod}(flip, 2)) \cdot dx_{\max}$
2: Compute p
3: $d_y \leftarrow 0$
4: **if** $p < L_{\text{thr}}$ **then**
5: $\quad d_y \leftarrow dy_{\max} \cdot (1 - \frac{p}{L_{\text{thr}}})$
6: **else if** $p \geq H_{\text{thr}}$ **then**
7: $\quad d_y \leftarrow -dy_{\max} \cdot \frac{p}{100}$
8: **end if**

$flip$ is an indicator of the direction and is used as follows: if it is even, the swarm goes from left to right; otherwise, the UAVs move from right to left. $dx_{\max}$ is the maximum reference distance that the path planner can impose in a time-step (see Figure 5.3.1). Notice that $dx_{\max}$ has to be chosen according to the maximum power of the drones.

The step in the y direction is computed on the basis of the percentage of overlap with the upper or lower previously scanned area or with the border of the map. We want to keep such a percentage within a tunable interval for two reasons: the lower bound L_{thr} guarantees full coverage of the area, whereas the upper bound H_{thr} prevents the swarm from going too much out of the unexplored area of interest. In the meantime, the overall polygon given by the union of all the union shapes at each

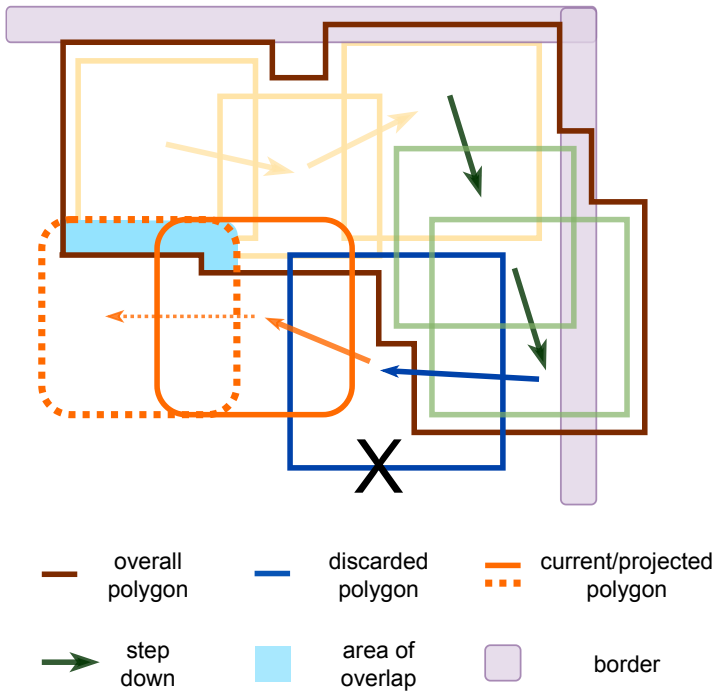


Figure 5.3.2: Polygon computation and turning phase

time-step is computed and stored. As the swarm moves across the map, whether from left to right or vice versa, the path planner evaluates the percentage of overlap between the current union shape and the overall polygon to determine the extent of coverage overlap. In addition, in this case, we want the current union shape to slightly overlap with the overall polygon above it. Also for this case, we want to keep the percentage within a tunable interval for the same reasons as before. It is important to point out that, in order to compute the overlap with the already explored area, the recently computed union shapes that overlap on the right or left with the current one have to be discarded from the computation of the overall polygon.

Figure 5.3.2 clarifies the aforementioned step of the proposed path planning algorithm. For the sake of simplicity, in the example shown, we assume that the union shape at each time-step is a square, but the algorithm works for polygons with general shape.

The overall polygon (brown shape) is computed by discarding the recently computed union shapes (blue square) that overlap with the current union shape (continuous orange square). This way, the area of overlap (light blue area in the figure) between the projection of the current polygon (dashed orange square) and the overall polygon is not affected by a possible lateral overlap with recent union shapes (blue square). Notice that this is not the case illustrated in the figure, but since the step forward can be small, it is likely to occur. It is also the case to specify that the blue square in the figure is discarded at the current time-step but will be taken into account to grow the overall polygon later in time when it does not overlap the current union shape.

Due to these considerations, the step along the y direction has to depend on the percentage of overlap related to the overall polygon (light blue area in Figure 5.3.2) or to the border of the map. Such a step is computed according to Algorithm 1 (Lines 2–8).

p is the percentage of overlap of the current union shape with the border or the overall polygon that has to be computed at each time-step. L_{thr} and H_{thr} are the lower and upper thresholds for the percentage of overlap, respectively, and $dy_{\max}$ is the maximum reference distance that the path planner can impose in a time-step along the y-axis (see Figure 5.3.1). Notice that $dy_{\max}$ has to be set according to the maximum power of the drones. In short, when $p < L_{\text{thr}}$, d_y is a fraction of $dy_{\max}$, which increases as p decreases, and vice versa. If $p \geq H_{\text{thr}}$, then d_y is a fraction of $-dy_{\max}$, which increases as p increases, and vice versa. Only when complete overlap occurs, i.e. $p = 100$, it results in $d_y = -dy_{\max}$. Notice that if $L_{\text{thr}} \leq p < H_{\text{thr}}$, $d_y = 0$, i.e. the swarm goes straight without adjustments in the y direction.

When the swarm reaches one of the two vertical borders, it goes vertically down for a number of steps equal to $\lfloor \gamma \cdot V \rfloor$, where V is the maximum length of the current union shape along the y-axis and $\gamma \in \mathbb{N}$ is a tunable strictly positive parameter whose value does not affect the correct flow of the algorithm. It has been observed that,

Table 5.1: Values of the weights in the cost functions of the NMPCs

	w_l	w_p	w_f	w_z	w_u	$w_{\Delta u}$
$\mathcal{J}_{1,T}$	-	15	-	2	0.001	0.001
$\mathcal{J}_{i,T}$	1.5	-	-	3	0.001	0.001
$\mathcal{J}_{1,S}$	-	-	-	2	0.001	0.001
$\mathcal{J}_{i,S}$	3	-	1	3	-	-

for better performance, a good choice for such a parameter is a value in the interval $[10, 20]$.

Once these steps have been taken, $flip$ is increased by 1. In other words, if the swarm is going from left to right, i.e. $flip$ is even, and meets the vertical border, it goes down and $flip$ becomes odd, thus changing the direction of d_x. This procedure is displayed in Figure 5.3.2 with green arrows and squares. As soon as the swarm touches the boundary, it takes two steps downwards before updating $flip$ and turning around. As one can notice from the figure, for the steps taken downwards both d_y and d_x are nonzero. In particular, $d_y = -dy_{\max}$ and d_x is proportional to the overlap with the lateral boundary. During these steps, similarly to the computation of d_y in Algorithm 1, the algorithm computes d_x to obtain a nonzero tunable overlap to increase coverage. Choosing $d_x \neq 0$ ensures greater coverage because the swarm is pushed toward the lateral border before turning around.

5.4 Simulation Results for Online Path Planning - Rectangle Area

In this section, simulation results are presented and discussed for the case of $N \in \{3, 4, 6, 8, 10\}$. As previously stated, the given area to cover is assumed to be an obstacle-free rectangle. Finally, we have set the values for the weights in the cost functions previously introduced, as in Table 5.1. A study on hyperparameter tuning for this problem is presented in Section 5.7.

Concerning the NMPC, $N_c = 2$, $N_p = 5$, $R_{\text{safe}} = 2m$, $r = 0.4$, and $T_s = 0.1\,s$.

Additionally, we have imposed upper and lower bounds for the drones' altitude, in particular $z_{\min} = 6\,m$ and $z_{\max} = 15\,m$. The velocity of the leader has been intentionally set lower than that of the follower agents to make sure that they can keep up with the leader. Therefore, $u_{x,\max} = u_{y,\max} = 0.7\,m/s$ in Eq. (4.6) for the leader, whereas $u_{x,\max} = u_{y,\max} = 1\,m/s$ for the followers. Finally, $u_{z,\max}$ has been set to $0.1\,m/s$ for all agents to avoid abrupt changes in altitude. Regarding the path planner, we have imposed a desired overlap in the range 20–60% and a maximum reference distance as $dx_{\max} = 0.1\,m$ and $dy_{\max} = 0.1\,m$. γ has been set to 15.

The available network bandwidth has been modeled as a uniform random distribution over the area to be scanned and has been assumed to be constant in time and altitude. Since, to the best of our knowledge, no dataset providing bandwidth measurements as a function of the geographical position of a drone is available, we have chosen a minimum value of 500 kbit/s and a maximum value of 5000 kbit/s.

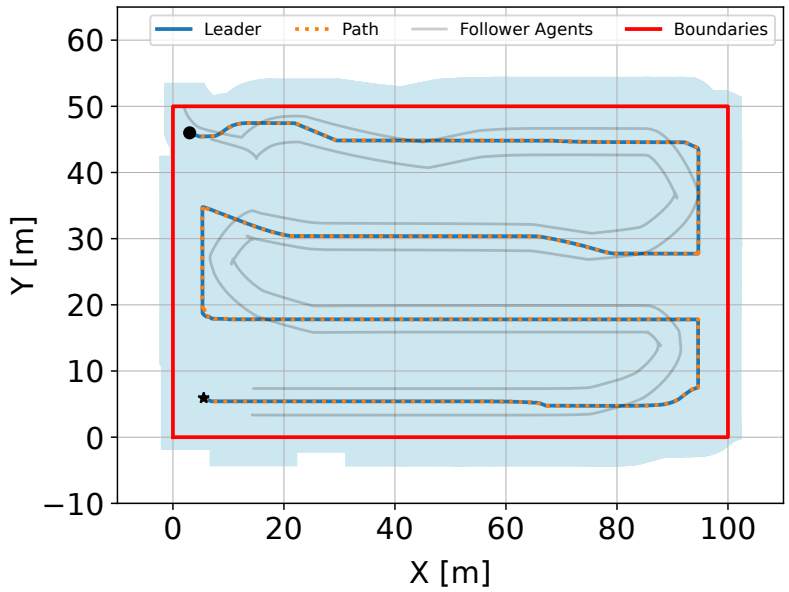


Figure 5.4.1: Drones' trajectories and area covered for $N = 3$.

Let us start by considering the case of $N = 3$ agents. Figure 5.4.1 shows the reference path provided by the path planner (dashed orange path in the figure), the drones' trajectories projected on the (x, y) plane (blue and gray paths), and the

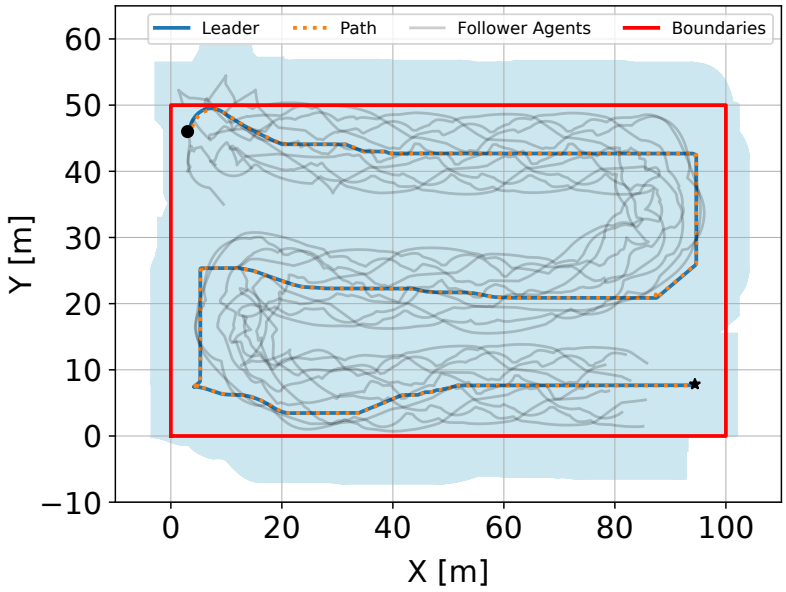


Figure 5.4.2: Drones' trajectories and area covered for $N = 10$.

overall area covered by their cameras (light blue area). The initial position of the leader (round marker) and of the other agents is in the top left corner of the given area. Looking at the drones' trajectories, one can observe that while the leader tracks the reference path until the end of the mission (star marker), the followers strive to catch up with the leader and stick together. It is also possible to inspect the adaptive nature of the proposed path planner, whose output trajectory tries to comply with the variable area scanned by each drone.

Figure 5.4.2 reports the case of $N = 10$ agents, where it is clear that, since the swarm can cover a considerably wider area around the reference path, it takes fewer turns for the path planner to complete the task. Intermediate results have been observed for the other numbers of agents, which you can find in Fig. 5.4.3 for $N = 6$ agents.

An interesting observation can be drawn from Figure 5.4.4, which shows how the percentage of the area covered by the swarm grows over time for each number of agents N considered for the simulations. The general trend is a decreasing time needed to complete the mission and a steeper curve as N increases.

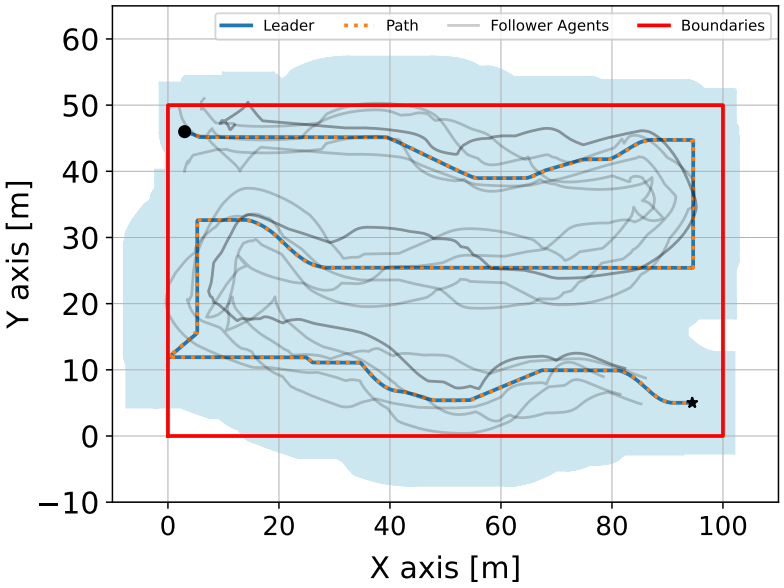


Figure 5.4.3: Drones' trajectories and area covered for $N = 6$.

Along the z-axis, the drones have to track a reference altitude based on their location and available network bandwidth, as computed in (4.19). Figure 5.4.5 shows the dynamics of the altitude tracking error for the case of $N = 6$ and $N = 10$ agents, respectively. All the agents start from the same initial altitude and then track their own reference altitude with satisfactory precision after a transient. At the beginning, there is a substantial error because the initial altitude of the agents is lower than the reference.

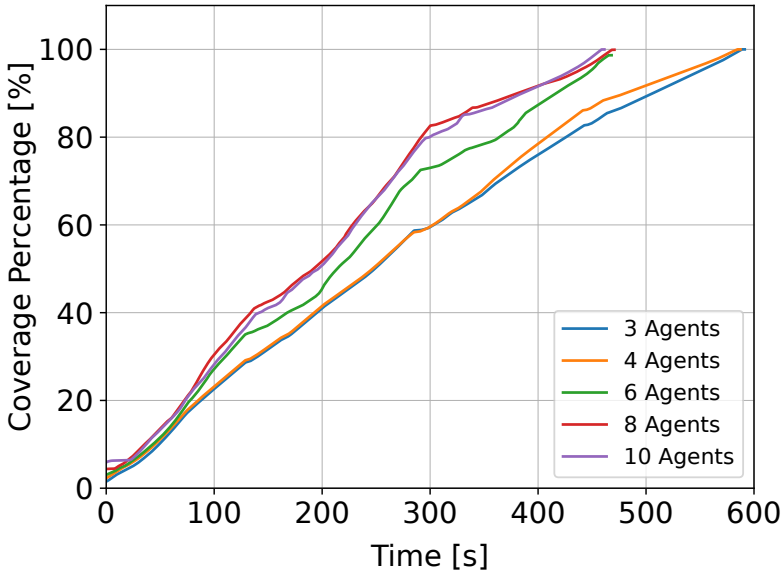
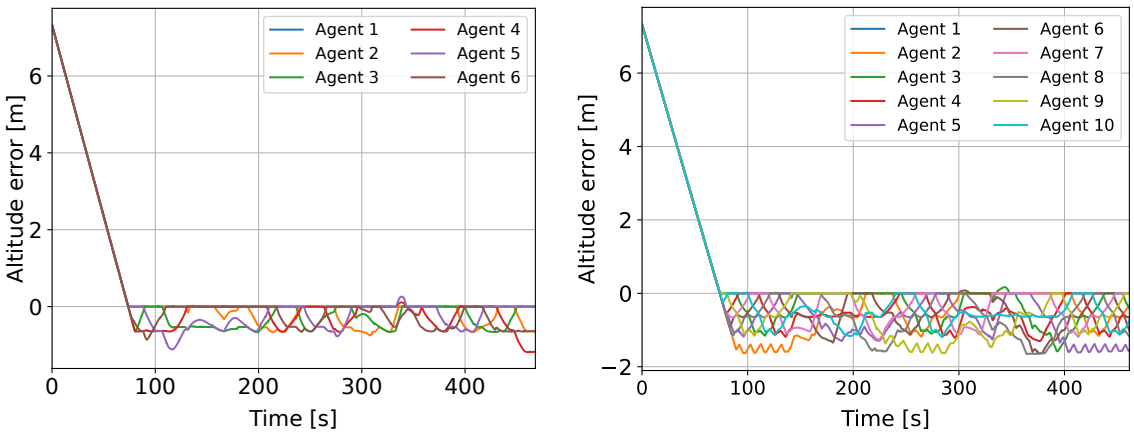


Figure 5.4.4: Percentage trend of the area covered over time.



(a) Tracking error of the altitude for $N = 6$ (b) Tracking error of the altitude for $N = 10$

Figure 5.4.5: Altitude tracking error.

Overall, the proposed path planner strives to provide full coverage and to adapt to the changing area scanned by the drones. In addition, it is scalable with respect to the number of drones since it runs only on the leader.

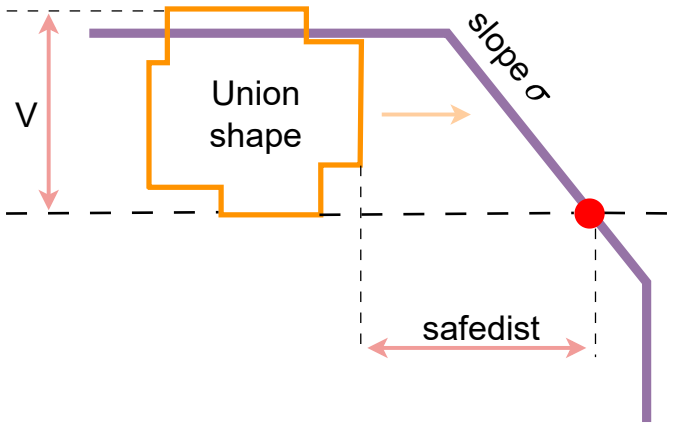


Figure 5.5.1: Path planning with sloped segments

5.5 Online Path Planning for Convex Areas

Let us now assume that the target area to be covered is no longer rectangular but has a general shape with the following characteristics: (i) the shape is polygonal and convex; (ii) the lowest segment is, without loss of generality, horizontal; (iii) the swarm always starts from the top left or, if there is only one, the top vertex.

In general, the algorithm is essentially the same as the one described in the case of a rectangular area. In particular, the lawnmower-like approach is preserved, thus Algorithm 1 still holds. However, when the boundary is made of sloped segments, determining when and how to perform the turning maneuver becomes less trivial and requires the choice of proper heuristics. In this work, at each time-step, assuming $flip$ is even (a similar rationale holds when $flip$ is odd), we compute the intersection point between the line passing through the lowest horizontal segment of the current union shape and the segment of the boundary on the right (see Figure 5.5.1). When the distance between this point and the vertex of the union shape with maximum x-coordinate ("safedist" in Figure 5.5.1) is less than a threshold, then the turning maneuver is triggered and works in the same way as explained above.

This time, the number of steps taken downward has to depend on the slope of the segment of the boundary that we have discussed. Also for this case, a certain overlap with the boundary has to be provided, therefore the component d_x is nonzero and follows the slope of the boundary. If the slope is not steep, then it is wise to take a larger number of turning steps before proceeding in the opposite direction in order to avoid a considerable overlap with the overall polygon when $flip$ changes, which would make the path more irregular and increase the probabilities of generating blind spots. Let σ be the slope, then the number of steps taken in the turning maneuver is:

$$\left\lfloor \gamma \cdot V + \rho \cdot \frac{1}{|\sigma|} \right\rfloor \tag{5.6}$$

where $\rho > 1$ is a tunable parameter that regulates the additional steps proportionally to the slope. Notice that if the segment is vertical, the rectangular case applies, i.e. the number of steps becomes $\lfloor \gamma \cdot V \rfloor$ as previously said.

The path planning algorithm keeps running until the swarm reaches the bottom of the area. In particular, the algorithm stops when the lowest segment of the boundary, which by assumption is horizontal, is fully covered.

5.6 Simulation Results for Online Path Planner for Convex Areas

In this section, we provide the simulation results obtained using MATLAB and the MPC Toolbox. The tuning of parameters has been carried out using Optuna in Python. Note that the formulation for the NMPC is exactly as the one in Section 5.4.

Our approach has been evaluated using swarms comprising different numbers of agents $N \in \{3, 4, 6, 8, 10\}$. The coverage mission has been repeated for various shapes of the area to be covered and under diverse bandwidth profiles, as will be

shown further on. Moreover, a sensitivity analysis has been carried out to discuss the impact of the weights of the cost functions on aspects such as coverage, energy, path length traversed by the agents, and simulation time.

Concerning the NMPC parameters, we have set $N_c = 2$, $N_p = 5$, $R_{\text{safe}} = 2m$, $r = 0.4$, and a sampling time of $T_s = 0.1s$. Upper and lower bounds have been imposed on the drones' altitude according to Eq. (4.19), specifically $z_{\min} = 6m$, and $z_{\max} = 15m$. The leader's velocity has been deliberately chosen to be lower than that of the follower agents to ensure that the latter ones can catch up with the leader. Accordingly, for the leader, $u_{x,\max} = u_{y,\max} = 0.7m/s$ in (4.6), whereas for the followers $u_{x,\max} = u_{y,\max} = 1m/s$. To prevent abrupt altitude variations, the vertical velocity limit has been set to $u_{z,\max} = 0.1m/s$ for all agents. Regarding the path planner, a desired overlap between 20% and 60% has been imposed, with maximum reference displacements of $dx_{\max} = 0.1m$ and $dy_{\max} = 0.1m$. The parameter γ related to the turning phase has been set to 15, whereas $\rho = 6$. Independently of the bandwidth profile adopted, we have chosen a minimum value of 500 kbit/s and a maximum value of 5000 kbit/s.

5.6.1 Results for Different Coverage Areas

The proposed path planner, along with the control framework, has been tested against different shapes of the given area to be covered. For the sake of brevity, we present the outcomes for two representative shapes. Similar trends are observed for other polygons. The bandwidth profile chosen for this analysis is a uniform random distribution, which, due to its unpredictability, reflects a realistic model.

In Figure 5.6.1, we show the results obtained in the case of a triangular shape and a swarm composed of 6 agents. Figure 5.6.2 displays a simulation where the given area is a pentagon and the coverage mission is assigned to a swarm of 8 agents. In both cases, the proposed online path planner guides the swarm across the given area while adapting to its shape and to the bandwidth conditions, as explained in

Section 5.5. It can be observed that our framework is able to achieve full coverage with rare and small unexplored spots. Results for other convex polygonal shapes exhibit similar coverage performance.

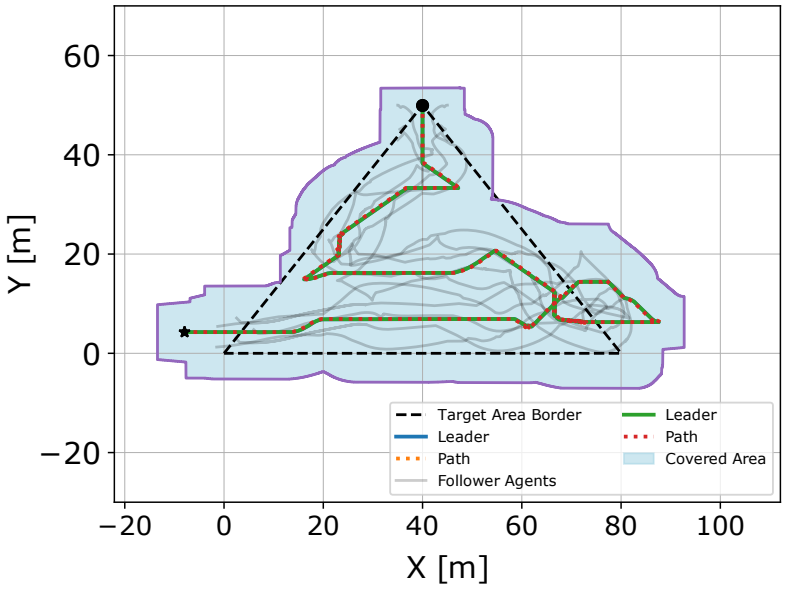


Figure 5.6.1: Triangular target area with 6 agents.

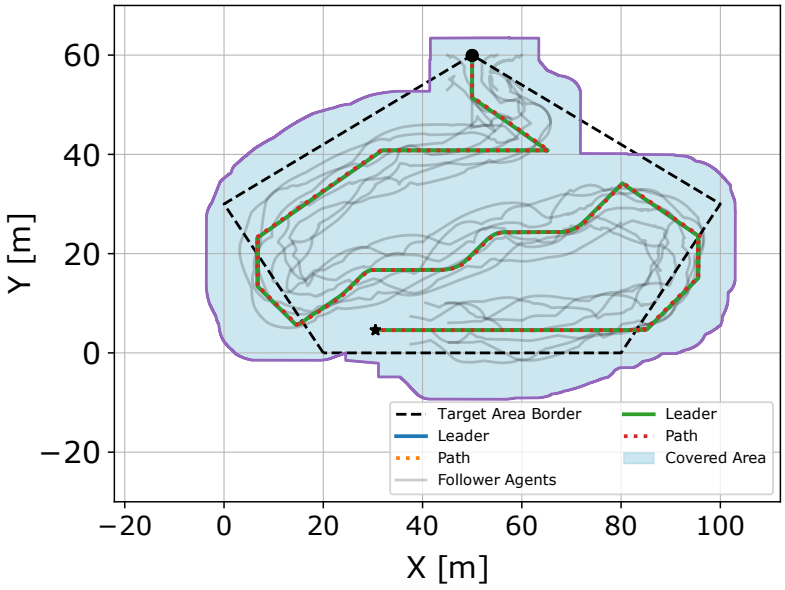


Figure 5.6.2: Pentagonal target area with 8 agents.

5.6.2 Altitude Tracking

So far, we have analyzed the performance obtained by the proposed approach on the (x, y)-plane. Let us now provide an overview of the results along the z-axis concerning altitude tracking. Such tracking is important to guarantee a high quality of the videos streamed to the GCS. However, due to the unpredictable nature of the bandwidth and its abrupt changes, the reference altitude computed through the mapping in (4.19) for each agent is characterized by sharp changes, too.

Therefore, each agent strives to track such a reference, as displayed in Figure 5.6.3 for the case of 8 agents covering a pentagonal area. From the figures, it can be observed that the agents struggle to perform perfect tracking of the reference altitudes because of the constraint on the control input (we have imposed $u_{z,\max} = 0.1m/s$) and the cost imposed on the variation of the control input in the NMPC cost function, i.e. $J_{\Delta u}$. Such limitations reflect those of a real-world UAV, which is not able to perfectly track a reference altitude with sharp changes but, as it can be surmised from Figure 5.6.3, the gradient of its altitude follows the gradient of the reference altitude. Thus, the trends obtained are satisfactory, allowing the agents, with limitations in velocity and acceleration, to stream videos at the best quality possible. To better understand the proposed framework, we provide a video of a simulation of 10 agents exploring a rectangular target area with a uniform random distribution of the bandwidth [1].

[1] https://www.youtube.com/watch?v=GaVvO07LFPY

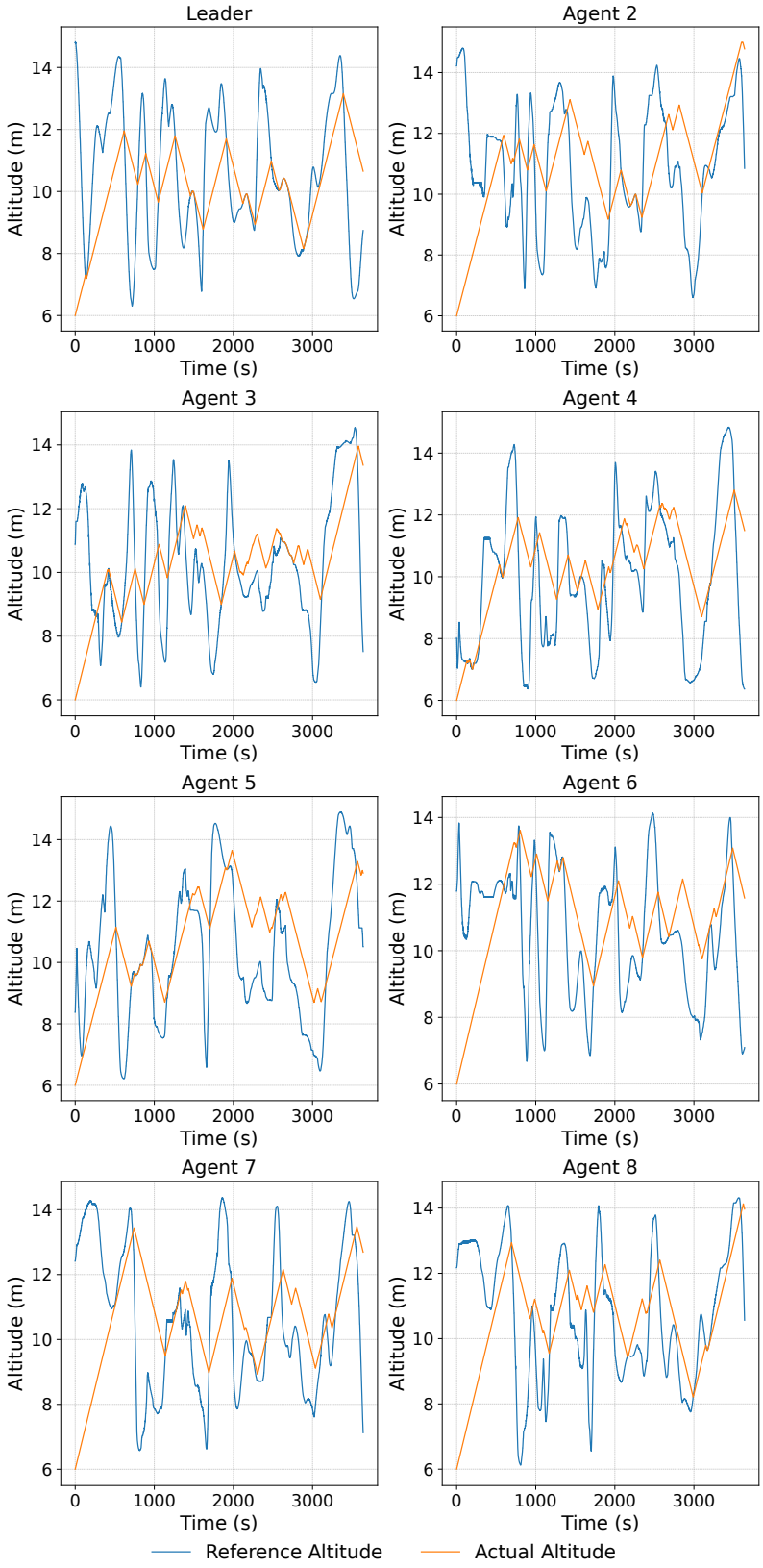


Figure 5.6.3: Altitude tracking for 8 agents with a pentagonal target area.

5.6.3 Simulations for Different Bandwidth Profiles

Let us now investigate the performance of the proposed framework–in particular the path planner–under different profiles of the available network bandwidth. In the first case, the proposed online path planner algorithm, which is based on the lawnmower approach, takes wider turns, but in a smaller number, to cover the whole area and reach the end with a resulting shorter path. Instead, in the second case, the path planner takes tighter turns, but in a greater number, to accomplish the mission. To better understand this concept, in the following we show some simulations under two different bandwidth distributions when, without loss of generality, the shape of the area to cover is a square (similar results hold for other polygonal shapes).

Case 1. We have considered an area characterized by a diagonal bandwidth profile that presents an abrupt change. In particular, the upper-left region exhibits the maximum available bandwidth considered in this work, i.e. 5000 kbit/s, whereas the lower-right region shows minimum bandwidth availability, i.e. 500 kbit/s. The simulation has been run for a swarm composed of $N = 6$ agents with the following weights: $w_l = 9$, $w_z = 10$, $w_u = 0.001$, $w_{\Delta_u} = 0.001$, $w_p = 6$, $w_f = 4$. Based on this setup, the union shape, i.e. the total area scanned at each time-step, is expected to be larger in the high-bandwidth region than in the low-bandwidth one. This behavior is illustrated in Figure 5.6.4, which also shows the leader's path (solid blue line) and the reference path (dashed orange line) going from the round marker to the starred marker. The figure also reports the FoVs of the swarm in five time-steps. In general, it can be observed that FoVs in the upper-left clear region, where the bandwidth is high, are larger than those in the lower-right dark region, where the bandwidth is low. However, when the swarm is in the initial point (round marker) or close to it, there are small FoVs despite a large availability of network bandwidth. This is due to the initial setting that we have imposed, according to which each agent starts the mission from the lowest altitude.

Finally, let us point out that the path planner takes larger turns in the high-

bandwidth area than in the low-bandwidth one, in accordance with our previous discussion and with the fact that, in the turning phase, the planner takes into account the width of the union shape according to Eq. (5.6).

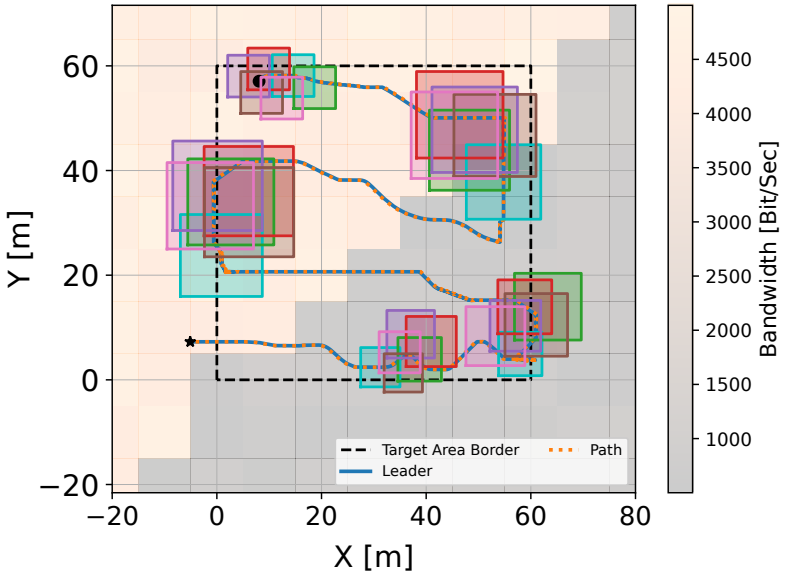


Figure 5.6.4: Diagonal bandwidth profile with 6 agents.

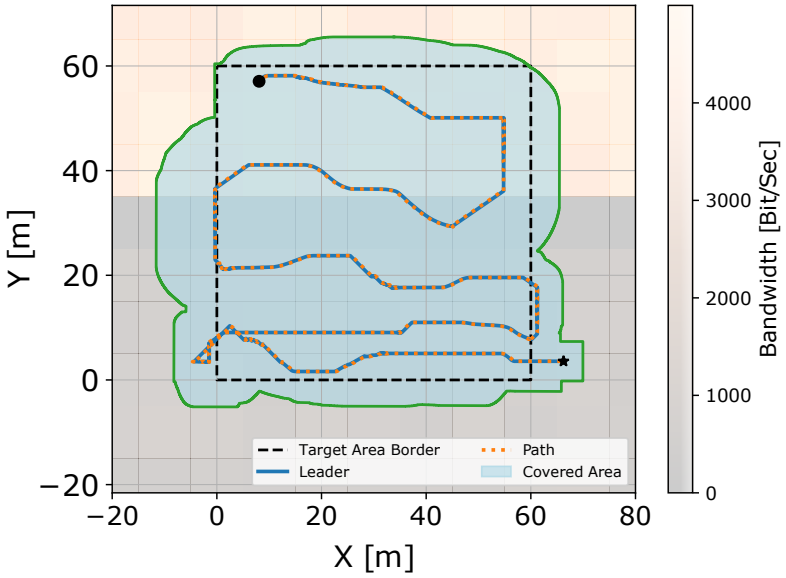


Figure 5.6.5: Horizontal bandwidth profile with 6 agents.

Case 2. Similarly to Case 1, the bandwidth distribution considered in this case

exhibits an abrupt change from the maximum value chosen for the bandwidth and the minimum one. This time, the square area that the swarm has to cover is divided horizontally into two parts: the upper part has maximum bandwidth, while the lower part has minimum bandwidth.

Figure 5.6.5 shows the results for a swarm of $N = 6$ agents and the same weights as in Case 1. The figure provides a clearer idea of the path planner's behavior in different bandwidth conditions. More precisely, the turns become tighter as the swarm moves downward due to the reduced area of the union shape. This way, the path planner prevents the formation of unexplored spots in the given area.

5.7 Simulation for Sensitivity Analysis

For a swarm with $N = 3$ agents, a sensitivity analysis has been conducted to show the effects of the single weights on the overall performance of the system. In particular, the weights related to altitude tracking (w_z), control input (w_u), control input variation rate ($w_{\Delta u}$), and leader tracking (w_l) have been chosen for this analysis.

We have evaluated the effect of each of the aforementioned weights by picking one and letting it range in an interval around the value chosen by Optuna, while all the other weights are kept unchanged. Specifically, we have focused on the effects on *Coverage Percentage*, i.e., the percentage of the target area covered by the swarm, *Mission Duration*, *Control Effort*, and *Path Length*. The Control Effort has been computed as $\sum_{i=1}^{N} \|u_i\|_2^2$, which is proportional to the energy consumed by the swarm. Similarly to what we have said for the cost function defined in Optuna, such performance metrics have been chosen based on the particular aspects we want to investigate. We have chosen to analyze the four aforementioned metrics because our goal is to cover a given area in the shortest time possible while having sufficient overlap between the agents' FoVs.

5.7.1 Parameter tuning

The tuning of the weights in the aforementioned cost functions has been performed using Optuna, a powerful hyperparameter optimization tool. Optuna is widely used in Machine Learning algorithms and, in general, in most optimization problems [65]. In this study, the weights of the cost functions have to be optimally set since they considerably impact the performance of the system.

Due to the complexity of the optimization problem (4.2)-(4.6), finding the best weights is not a trivial task. Thus, Optuna has been employed to identify an optimal combination of weights in our problem. To this end, the following cost function has been defined for Optuna to minimize.

$$\mathcal{J}_{\text{Optuna}} = 0.05 \cdot J_{\text{Coverage}} + 0.9 \cdot J_{\text{Detachment}} + 0.05 \cdot J_{\text{SimLength}} \tag{5.7}$$

where J_{Coverage} penalizes insufficient area coverage, $J_{\text{Detachment}}$ penalizes the number of potential detachment events between agents during the simulation, and $J_{\text{SimLength}}$ minimizes the total simulation time to complete the mission as quickly as possible. It is worth noting that alternative metrics can be selected depending on the specific performance objectives, thus leading to a different cost function $\mathcal{J}_{\text{Optuna}}$. The weights chosen in Eq. (5.7) reflect our specific preferences, as we prioritize maintaining team cohesion. For the case with $N = 3$ agents, Optuna suggested the following weights for the tracking and recovery states: $w_p = 1$, $w_f = 2$, $w_z = 10$, $w_f = 2$, $w_u = 0.4$, $w_{\Delta u} = 0.005$, and $w_l = 3$.

5.7.2 Weight for Control Input Variation ($w_{\Delta u}$)

As illustrated in Figure 5.7.1, the red marker indicates the value suggested by Optuna. This configuration exhibits both advantages and drawbacks when compared with the other weight values. In terms of coverage, the performance barely changes. However,

the Optuna-selected value achieves a reduction of more than 20 seconds in terms of mission duration. Despite higher weight values leading to a lower duration, the energy consumption in terms of control effort increases, and the agents' paths become longer. As expected, the final choice of this weight implies a trade-off between speed, energy, and profile of the path.

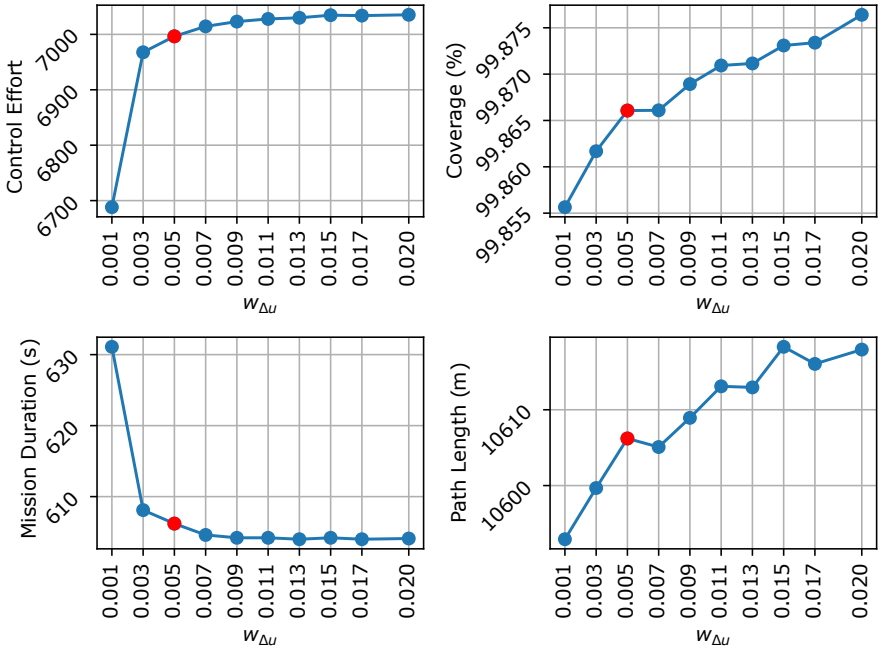


Figure 5.7.1: Sensitivity analysis for $w_{\Delta u}$

5.7.3 Weight for Control Input (w_u)

From Figure 5.7.2, it can be observed that when w_u is greater than 0.4, the agents tend to either stop or move significantly more slowly. This can be observed from the mission duration, which reaches the maximum limit imposed, i.e., 800 s, with a very low percentage of coverage and a short path length. This is quite expected since increasing this weight implies penalizing the control effort through the term J_u in the cost functions. On the other hand, for values less than 0.4, the performance metrics are approximately unchanged, although a slight increase in control effort and

path length is experienced.

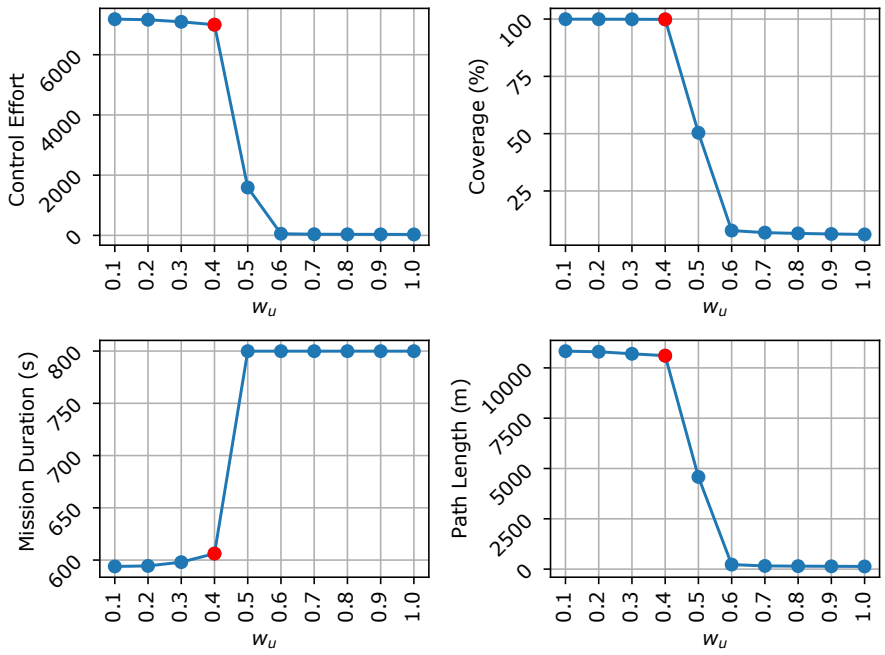


Figure 5.7.2: Sensitivity analysis for w_u

5.7.4 Weight for Altitude Tracking (w_z)

As Figure 5.7.3 shows, this weight only slightly affects the performance metrics, whose changes are negligible. In general, the altitude tracking weight in this framework does not have a distinctive effect on the investigated performance. This is due to the fact that UAVs have a limited vertical velocity that does not allow improvement in altitude tracking.

5.7.5 Weight for Leader Tracking (w_l)

Results in Figure 5.7.4 confirm the effectiveness of Optuna, which strove to find an optimal weight w_l such that the coverage is maximized and the mission duration is minimized while avoiding detachments, according to Eq. (5.7). However, while the selected weight $w_l = 3$ takes the mission duration to the minimum, the control effort

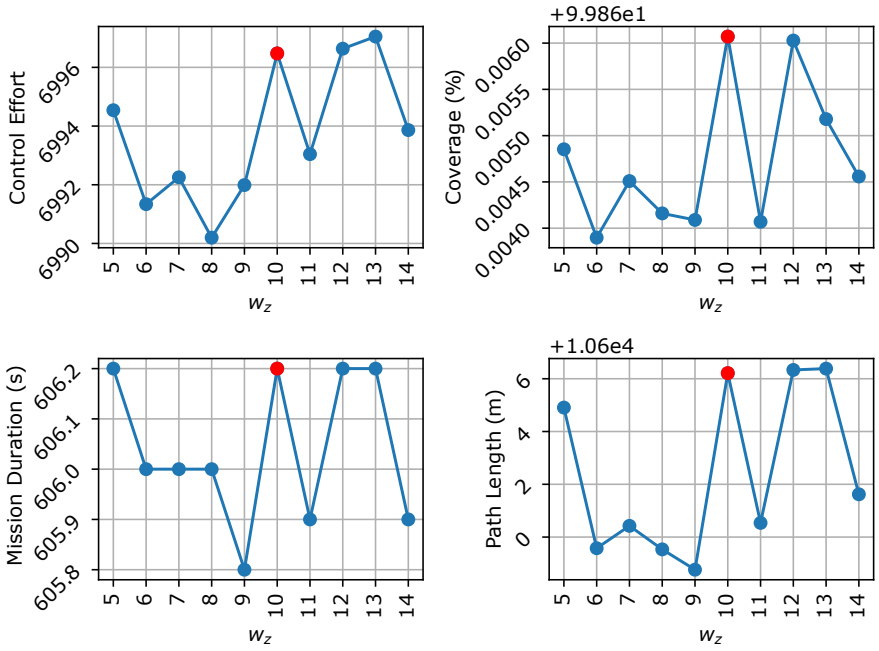


Figure 5.7.3: Sensitivity analysis for w_z

is at its peak. As expected, it takes a higher control effort to reduce the mission duration, and this is paid in terms of energy. Moreover, coverage reaches a practical optimum.

Notice that $w_l = 3$ is the best choice according to the cost function that we have defined for Optuna. Since in this cost function we have not included any cost related to energy or path length, the optimization only focused on obtaining a low mission duration with a high coverage percentage.

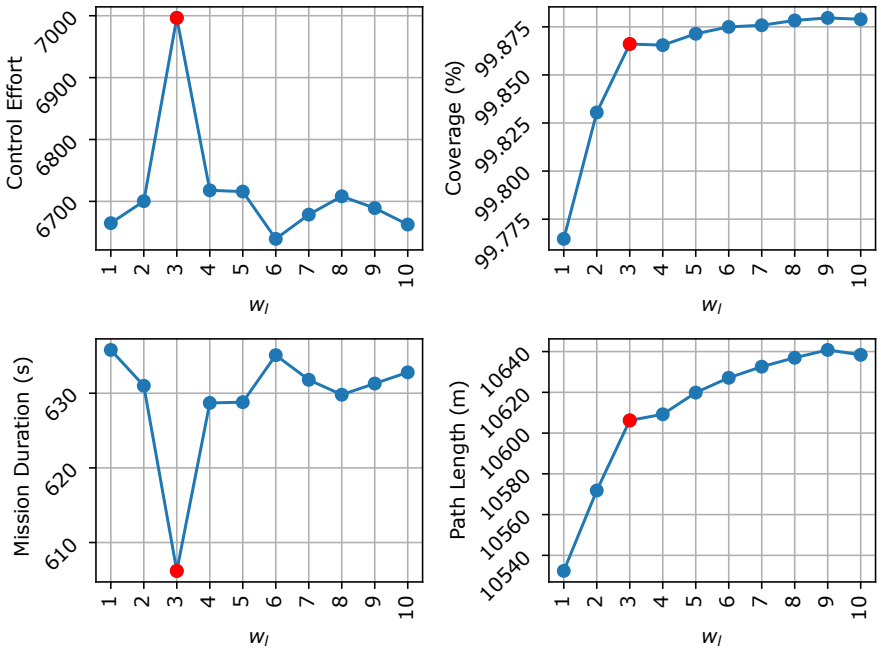


Figure 5.7.4: Sensitivity analysis for w_l

Chapter 6

Event-Triggered Communication

6.1 Event-triggered Communication Framework

Although UAV swarms are associated with many advantages including reduced costs, reduced task completion times, and scalability, the large communication traffic among the agents can lead to several issues, such as packet loss due to network congestion, that can lead to task failures. This is further exacerbated in live monitoring, where communicating live information is crucial to the task itself. To tackle these issues, an Event-Triggered Communication (ETC) scheme has been proposed to enable agents to share their own information only if a triggering condition is met, leading to a lower number of communications and reducing the load on the network, hence reducing the consumption of the available network bandwidth. This method decreases the amount of communications dramatically; however, when the triggering condition is not properly designed, zeno behaviors and asynchronous updates can occur leading to the mission failure.

In this scenario, we tackle the task of live video streaming with a swarm of drones, connected through LTE/5G networks, with the goal of maximizing the coverage for a given area. As mentioned before, each drone is required to transmit its video feed to a GCS while improving the overall QoE. The differences with respect to the

previously proposed methods are:

- A nonlinear Distributed Model Predictive Control (DMPC) controller for coverage maximization with UAV swarms that considers both the available network bandwidth for motion control, and an event-triggered communication scheme to save the bandwidth for live streaming.
- A novel event-triggering condition for the communication scheme based on the deviation of the agent's trajectory from the transmitted one, resulting in a much lower bandwidth consumption.
- An in-depth analysis of the triggering threshold ε to highlight the trade-offs at communication rates and formation accuracies, which provides a practical tool to identify the best value with respect to the task requirements.

For this scenario, another UAV model has been considered where it includes 6-state model capturing all the physical features of a real drone. A leader-follower structure has been used for the swarm where the leader has a knowledge of the area to be covered while the followers follow the leader keeping a relative distance to have sufficient overlapped videos.

6.2 Quadrotor Dynamics

We consider a swarm of N quadrotor UAVs. Each UAV $i \in \{1, \ldots, N\}$ is described by a 6-dimensional state vector combining position and velocity:

$$\mathbf{x}_i = [p_x, p_y, p_z, v_x, v_y, v_z]^T \quad \in \mathbb{R}^6 \tag{6.1}$$

where (p_x, p_y, p_z) are the position coordinates and (v_x, v_y, v_z) are the linear velocities in the inertial frame. The control input for each agent is a velocity vector as:

$$\mathbf{u}_i = [u_x, u_y, u_z]^T \quad \in \mathbb{R}^3 \tag{6.2}$$

The prediction model captures the closed-loop behavior of a low-velocity controller as a first-order lag:

$$\dot{\mathbf{v}}_i = \frac{1}{\tau}(\mathbf{u}_i - \mathbf{v}_i), \quad \dot{\mathbf{p}}_i = \mathbf{v}_i \tag{6.3}$$

where $\tau > 0$ is the velocity time constant that characterizes the closed-loop response. This model is compactly written as $\dot{\mathbf{x}}_i = f(\mathbf{x}_i, \mathbf{u}_i)$ and discretized with a fourth-order Runge-Kutta (RK) scheme at sampling time T_s:

$$\mathbf{x}_i(k+1) = F(\mathbf{x}_i(k), \mathbf{u}_i(k)) \tag{6.4}$$

Although each physical quadrotor has a 12-dimensional state, including attitude and angular rates, this 6-state model faithfully captures the velocity tracking behavior over the NMPC prediction horizon ($H_P T_S$) because a fast low-level controller ($>> 1/T_s$) ensures tracking of the control inputs on a timescale much shorter than T_s. The resulting model mismatch provides an implicit robustness test of the proposed event-triggered architecture.

6.3 Leader-Follower Architecture

The swarm is organized in a leader-follower topology. The UAV $i = 1$ is designated as the leader, which plans and follows the reference path in 2D, $\bar{\mathbf{p}}(t) = [\bar{p}_x(t), \bar{p}_y(t)]^T$ to maximize the area coverage. The remaining UAVs, $i = 2, \ldots, N$ are followers, tasked with maintaining a desired formation offset $\boldsymbol{\delta}_i \in \mathbb{R}^3$ relative to the leader:

$$\lim_{t \to \infty} |\mathbf{p}_i^{\text{des}}(t) - \mathbf{p}_1(t) - \boldsymbol{\delta}_i| = 0 \tag{6.5}$$

where $\mathbf{p}_i^{\text{des}}(t)$ is the desired 2D position of the ith agent. The underlying communication graph among agents is a fully connected one $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \ldots, N\}$ and $(i, j) \in \mathcal{E}$ means agent j receives predicted trajectory and state information from agent i.

6.4 Bandwidth Modelling

Although the drones can only measure bandwidth and cannot predict it, we assume the available bandwidth in an obstacle-free field is determined by hotspots distributed in the area: drones closer to the hotspots will measure a higher bandwidth than the ones flying further from the antennas. In particular, we model the bandwidth as a sum of Gaussian for the signal intensity [66] whose peaks are centered at the hotspots:

$$b(\mathbf{p}) = b_{\text{base}} + \sum_{j=1}^{N_h} b_j^{\text{peak}} \exp\left(-\frac{\|\mathbf{p}_{xy} - \mathbf{c}_j\|^2}{2\sigma_j^2}\right) \tag{6.6}$$

where the hotspot parameters $(\mathbf{c}_j, b_j^{\text{peak}}, \sigma_j)$ model a realistic coverage pattern. In order to optimize both formation tracking and bandwidth-driven altitude adaptation the NMPC is extended with a measurement of the local bandwidth b_i at its current position. Then, the reference altitude is computed via a linear mapping, as mentioned in (4.19). Note that, while in practice 5G/LTE antennas can provide a bandwidth ranging from hundreds of megabits per second to several gigabits per second, the experiment we model is a scaled-down version that serves the purpose of validating the methodology.

6.5 Event-Triggered Distributed MPC

6.5.1 Baseline: Time-Triggered DMPC

At each discrete time-step k, each agent i solves a local NMPC problem over a prediction horizon H_P:

$$\min_{\mathbf{u}_i(k:k+H_P-1)} \quad J_i \tag{6.7a}$$

$$\text{s.t.} \quad \mathbf{x}_i(k+l+1) = F(\mathbf{x}_i(k+l), \mathbf{u}_i(k+l)) \tag{6.7b}$$

$$\|\mathbf{p}_i^l - \mathbf{p}_j^l\|_2^2 \geq d_{\min}^2, \ \forall j \neq i \tag{6.7c}$$

$$\|\mathbf{u}_i(k+l)\|_\infty \leq v_{\max} \tag{6.7d}$$

$$p_{z,i}(k+l) \in [z_{\min}, z_{\max}] \tag{6.7e}$$

for $l = 0, \ldots, H_P - 1$, where Eq. (6.7c) enforces pairwise collision avoidance with inflated safety distance $d_{\min} = d_{\text{safe}} + d_\varepsilon$ (see Section 6.5.4), Eq. (6.7d) bounds the velocity commands, and Eq. (6.7e) constrains the altitude. In all experiments, the NMPC cost is formulated as a Nonlinear Least Squares (NLS) problem, enabling efficient Gauss-Newton Hessian approximations.

6.5.2 Standard NMPC

The stage cost residual is:

$$\mathbf{y}_i^l = \begin{bmatrix} \mathbf{x}_i^l \\ \mathbf{u}_i^l \\ \Delta\mathbf{u}_i^l \end{bmatrix} \in \mathbb{R}^{12}, \quad \Delta\mathbf{u}_i^l = \mathbf{u}_i^l - \mathbf{u}_i^{l-1} \tag{6.8}$$

and the cost function is:

$$J_i = \sum_{l=0}^{H_P-1} \|\mathbf{y}_i^l - \mathbf{y}_{\text{ref},i}^l\|_W^2 \ + \ \|\mathbf{x}_i^{H_P} - \mathbf{x}_{\text{ref},i}^{H_P}\|_{W_e}^2 \tag{6.9}$$

where $W = \text{diag}(Q_p, Q_v, R, S) \in \mathbb{R}^{12\times12}$ is the stage weight matrix with position tracking weight Q_p, velocity damping weight Q_v, control effort weight R, and control smoothness weight S. The terminal weight is $W_e = \gamma_e \cdot \text{diag}(Q_p, Q_v) \in \mathbb{R}^{6\times6}$ with terminal scaling factor γ_e, and the control input is $\mathbf{u}_i = [u_x, u_y, u_z]^T \in \mathbb{R}^3$.

Since the bandwidth-altitude constraint Eq. (6.11) may not always be strictly satisfied, we introduce a slack variable s_{bw} that augments the control input vector $\tilde{\mathbf{u}}_i = [u_x, u_y, u_z, s_{\text{bw}}]^T \in \mathbb{R}^4$ and heavily penalized in the cost ($w_{bw} = 1000$, allowing the solver to find a feasible solution at all times. The stage cost residual becomes 14-dimensional:

$$\mathbf{y}_i^l = [p_x,\ p_y,\ p_z,\ p_z^{cov},\ v_x,\ v_y,\ v_z,\ u_x,\ u_y,\ u_z,\ \Delta\mathbf{u}^T,\ s_{\text{bw}}]^T \tag{6.10}$$

The term p_z (weighted by w_z) tracks the bandwidth-derived reference altitude $\bar{z}_i = H(b_i)$ from (4.19), penalizing deviation from the altitude that the current bandwidth supports.

The term p_z^{cov} (weighted by w_{cov}) drives the drone toward the *coverage-optimal* altitude $z_{\max}^{\text{bw}}$, which maximizes the camera footprint. Note that this term is not influenced by the bandwidth and, therefore, always drives the drone towards higher altitudes. When bandwidth is low, the optimizer will compute the best trade-off between the two components.

The bandwidth altitude coupling is further enforced via a nonlinear constraint:

$$\alpha_b \cdot p_{z,i}^2 - b_i \leq s_{\text{bw},i}, \quad s_{\text{bw},i} \geq 0 \tag{6.11}$$

where α_b is the bitrate coupling coefficient and b_i is the locally measured bandwidth. This constraint encodes the physical relationship between altitude and required video bitrate (which scales with the ground area in the FoV): higher altitude demands more bandwidth. The slack s_{bw} is heavily penalized ($w_{\text{bw}} = 1000$) to ensure soft feasibility.

The terminal cost uses a 7-dimensional residual $[p_x, p_y, p_z, p_z^{cov}, v_x, v_y, v_z]^T$ with

weights scaled by γ_e. For the *standard formulation* (Experiments B and C), the cost uses the 12-dimensional residual Eq. (6.8) without altitude duplication or bandwidth coupling.

For the leader ($i = 1$), the reference is the planned path: $\mathbf{y}^l_{\text{ref},1} = [\bar{\mathbf{p}}^l;\ \mathbf{0}_3;\ \mathbf{0}_3;\ \mathbf{0}_3]$ (standard) or extended with $\bar{z}_1 = H(b_1)$ and $z^{\text{bw}}_{\max}$ targets (bandwidth-aware). For each follower ($i \geq 2$), the reference position is the leader's assumed position plus the formation offset: $\mathbf{p}^l_{\text{ref},i} = \hat{\mathbf{p}}^l_1 + \boldsymbol{\delta}_i$, with altitude overridden by $H(b_i)$ in the bandwidth-aware mode.

The collision avoidance constraint Eq. (6.7c) is implemented as a soft nonlinear constraint with L_1 penalty (linear slack cost z_l and quadratic slack cost Z_l), ensuring feasibility under tight formations:

$$\|\mathbf{p}^l_i - \mathbf{p}^l_j\|_2^2 \geq d^2_{\min} - s^l_{ij}, \quad s^l_{ij} \geq 0 \tag{6.12}$$

with penalty terms $z_l \cdot s^l_{ij} + Z_l \cdot (s^l_{ij})^2$ added to the cost.

In the Time-Triggered (TT) baseline, each agent i broadcasts its full planned trajectory $\mathcal{X}^{\text{plan}}_i(k) = \{\mathbf{x}_i(k+1|k), \ldots, \mathbf{x}_i(k+H_P|k)\}$ to all neighbors $j \in \mathcal{N}_i$ at every step k where each message contains $6 \times H_P$ floating-point values.

6.5.3 Event-Triggered Communication Protocol

The proposed Event-Triggered Distributed MPC (ET-DMPC) replaces the continuous broadcast with a selective communication scheme based on the principle of shared knowledge (dead reckoning).

6.5.3.1 Shared Knowledge and Forward-Shifting

When agent i broadcasts its trajectory $\mathcal{X}_i^{\text{plan}}$ at time k_b (the last broadcast time), both agent i and its neighbors store this trajectory as the last valid plan:

$$\hat{\mathcal{X}}_i(k_b) = \mathcal{X}_i^{\text{plan}}(k_b) = \{\mathbf{x}_i(k_b+1|k_b), \ldots, \mathbf{x}_i(k_b+H_P|k_b)\} \tag{6.13}$$

At a subsequent time $k > k_b$, the forward-shifted prediction of agent i's trajectory, as assumed by all agents, is obtained by shifting the stored plan by $\Delta k = k - k_b$ steps:

$$\hat{\mathcal{X}}_i^+(k) = \{\mathbf{x}_i(k+1|k_b), \ldots, \mathbf{x}_i(k+H_P|k_b)\} \tag{6.14}$$

where entries beyond the original horizon are padded by holding the last predicted state constant, i.e., $\mathbf{x}_i(k+l|k_b) = \mathbf{x}_i(k_b+H_P|k_b)$ for $l > H_P - \Delta k$. This is a reasonable assumption since the terminal state of the MPC prediction is typically near equilibrium.

6.5.3.2 Triggering Condition

At each step k, agent i solves its local NMPC and obtains the new optimal trajectory $\mathcal{X}_i^*(k)$. The normalized trajectory deviation is computed using only the *position* components:

$$e_i(k) = \frac{\|\mathcal{P}_i^*(k) - \hat{\mathcal{P}}_i^+(k)\|_F}{\|\hat{\mathcal{P}}_i^+(k)\|_F + \delta} \tag{6.15}$$

where $\mathcal{P}_i^*(k), \hat{\mathcal{P}}_i^+(k) \in \mathbb{R}^{H_P \times 3}$ are the position sub-matrices of the respective trajectory matrices, $\|\cdot\|_F$ denotes the Frobenius norm, and $\delta > 0$ is a small regularization constant preventing division by zero.

The communication decision rule is:

$$\text{Broadcast at step } k \iff e_i(k) > \varepsilon \;\text{ or }\; k - k_b > T_{\max} \tag{6.16}$$

where $\varepsilon \in [0, 1)$ is the event threshold (the primary design parameter) and $T_{\max}$ is a mandatory broadcast period that guarantees that the shared knowledge will not become more outdated than maximum allowed. Setting $\varepsilon = 0$ recovers the time-triggered baseline (broadcast at every step).

6.5.3.3 Protocol Operation

When the event is triggered, agent i broadcasts $\mathcal{X}_i^*(k)$ and updates the shared knowledge: $\hat{\mathcal{X}}_i \leftarrow \mathcal{X}_i^*(k)$, $k_b \leftarrow k$. When the event is not triggered, agent i remains silent, and all neighbors continue to use $\hat{\mathcal{X}}_i^+(k)$ as the assumed trajectory for agent i in their local NMPC computations. The complete procedure is summarized in Algorithm 2.

Algorithm 2 ET-DMPC for Agent i at Step k

1: Measure current state $\mathbf{x}_i(k)$
2: Receive new broadcasts from neighbors and update shared knowledge $\hat{\mathcal{X}}_j$
3: For silent neighbors j, compute $\hat{\mathcal{X}}_j^+(k)$ via (6.14)
4: Solve local NMPC (6.7) using neighbors' trajectories to obtain $\mathcal{X}_i^*(k)$
5: Compute deviation $e_i(k)$ via (6.15)
6: **if** $e_i(k) > \varepsilon$ **or** $k - k_b > T_{\max}$ **then**
7: Broadcast $\mathcal{X}_i^*(k)$ to neighbors
8: $\hat{\mathcal{X}}_i \leftarrow \mathcal{X}_i^*(k)$
9: $k_b \leftarrow k$
10: **else**
11: Remain silent
12: **end if**
13: Apply control input:

$$\mathbf{u}_i(k) = \mathbf{u}_i^*(k|k)$$

6.5.4 Safety and Feasibility Considerations

6.5.4.1 Collision Avoidance Under Stale Information

When a follower j uses the forward-shifted trajectory $\hat{\mathcal{X}}_i^+(k)$ instead of the true trajectory $\mathcal{X}_i^*(k)$ of a neighbor i, the actual position of agent i may deviate from

the assumed position by a margin proportional to ε. To ensure safety, we inflate the collision avoidance distance by a threshold-dependent margin:

$$d_{\min} = d_{\text{safe}} + d_\varepsilon, \qquad d_\varepsilon = \alpha \cdot \varepsilon \cdot L \tag{6.17}$$

where $\alpha > 0$ is a tunable safety factor and $L > 0$ is a scaling constant. When $\varepsilon = 0$ (time-triggered), $d_\varepsilon = 0$ and the original safety distance is recovered. The constraint is applied in squared form: $\|\mathbf{p}_i^l - \mathbf{p}_j^l\|_2^2 \geq d_{\min}^2$.

6.5.4.2 Recursive Feasibility

The forward-shifted plan $\hat{\mathcal{X}}_i^+(k)$ was dynamically feasible when originally computed at time k_b. Under the event-triggered protocol, if $e_i(k) \leq \varepsilon$, the actual trajectory $\mathcal{X}_i^*(k)$ is close to $\hat{\mathcal{X}}_i^+(k)$, ensuring that the neighbors' optimization landscape does not change drastically between communication events. The mandatory broadcast timer $T_{\max}$ provides an additional safeguard: even in the worst case, the shared knowledge is never older than $T_{\max}$ steps.

6.6 Simulation Results for Event-Triggered Communication

6.6.1 Simulation Environment

The framework is evaluated on a swarm of Bitcraze Crazyflie 2.x micro-quadrotors, simulated in NVIDIA IsaacSim[1]. The physical parameters are listed in Table 6.1. IsaacSim is a high-fidelity physics simulator that provides realistic rigid-body dynamics, sensor models, and rendering, at a physics rate of 120 Hz, with rendering at 60 Hz (two physics substeps per render frame). The simulation is integrated with

[1] https://developer.nvidia.com/isaac/sim

Table 6.1: Crazyflie 2.x Physical Parameters

Parameter	**Symbol**	**Value**
Mass	m	0.028 kg
Inertia (roll/pitch)	$J_{xx} = J_{yy}$	1.4×10^{-5} kg·m^2
Inertia (yaw)	J_{zz}	2.17×10^{-5} kg·m^2
Thrust-to-weight ratio	T/W	1.9
Moment scale	M_s	2×10^{-4} N·m

ROS2[2] (Humble), enabling a modular architecture where each UAV runs its own NMPC node. For example, such environment with $N = 8$ UAVs is depicted in Fig. 6.6.1. Each UAVs state $\mathbf{x}_i^{UAV} = [p_x, p_y, p_z, v_x, v_y, v_z]^T$ is published as JSON at the render rate, and velocity commands are received from NMPC nodes. Moreover, Fig. 6.6.2, illustrates trajectories and the covered area for $N = 5$ UAVs'.

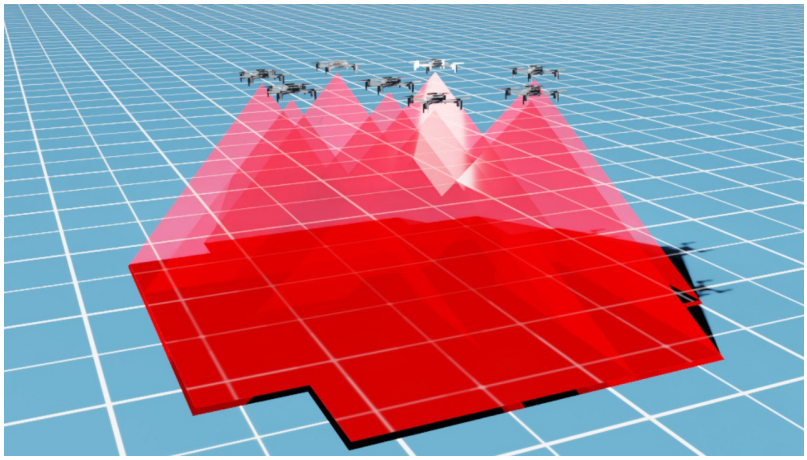


Figure 6.6.1: NVIDIA IsaacSim simulation environment with $N = 8$ UAVs performing a coverage task in leader-follower formation.

6.6.2 Hierarchical Control Architecture

A two-level control architecture, which is standard practice for multi-rotor systems [67] [68], bridges the gap between the NMPC prediction model (Section IV) and the physics of the UAV:

[2]https://docs.ros.org/en/humble/index.html

1. **High-level NMPC** ($1/T_s = 10$ Hz): computes velocity commands $\mathbf{u}_i = [u_x \quad u_y \quad u_z]$ using the prediction model (Eq. 6.3). The NMPC receives the measurements via ROS2, and publishes the computed velocity command. The optimization problem is solved using Acados [69] (SQP-RTI solver with HPIPM as the QP backend and ERK integration).

2. **Low-level PID controller** (120 Hz): a cascaded PI-PD controller converts the velocity command into normalized wrench actions $\mathbf{a} \in \mathbb{R}^4$ where each value inside $\mathbf{a}$ is ranged between -1 and 1, mapping to collective thrust and body-axis moment via a three-stage pipeline: (i) vertical velocity $\rightarrow$ thrust (PI with gravity and tilt compensation); (ii) horizontal velocity $\rightarrow$ desired roll/pitch (PI, clamped to ± 15); (iii) attitude $\rightarrow$ moments (PD at 3 Hz bandwidth, a design choice to prevent bang-bang oscillation given the Crazyflie's small moment scale M_s).

6.6.3 Experiments

In order to validate the proposed methodology, three different experiments have been designed. To obtain a reliable representation of the performance, we use the Monte Carlo approach. The event threshold is swept over a total of 8 values ($\varepsilon \in \{0, 0.01, 0.02, 0.05, 0.10, 0.15, 0.20, 0.30\}$), and the swarm size is tested for 3 values ($N \in \{3, 5, 8\}$). For each combination ($\varepsilon, N,$ experiment), we perform $K = 50$ Monte Carlo runs with randomized initial positions within ± 1m of the nominal formation, for a total of 1200 runs for each experiment.

Experiment A – Bandwidth-aware coverage: Experiments A is meant to evaluate the performance of the proposed methodology: the leader follows a customized Lawnmower path over a 20×20m rectangular area. Each UAV measures the local bandwidth at every step and computes a reference altitude via the mapping $H(b_i)$ in Eq. (4.19). In low-bandwidth regions, drones lower their altitude to

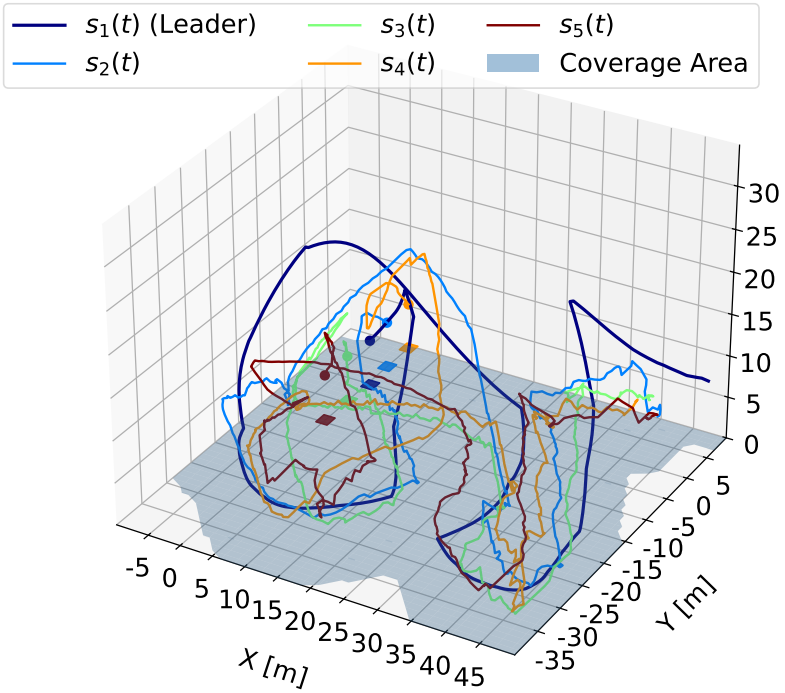


Figure 6.6.2: UAVs' trajectories and the covered area for $N = 5$.

improve video pixel density, meanwhile, in high-bandwidth regions, they go up for greater coverage. The formation spacing adapts dynamically via the FoV geometry: $d_{\text{fov}}(z) = 2z\tan(\theta_{\text{fov}}/2)(1 - o/100)$, where θ_{fov} is the camera half-angle and o the overlap percentage.

Experiment B – Standard coverage baseline: This experiment shares the same $20m \times 20m$ Lawnmower mission and the leader's path as in Experiment A, but uses the standard NMPC without bandwidth adaptation: drones fly at a constant reference altitude with fixed formation spacing, isolating the contribution of the bandwidth-aware formulation. Note that the event-trigger is still applied.

Experiment C – Aggressive maneuvering: The leader tracks a 3D figure-eight trajectory with $v_{\max} = 1.0m/s$ (vs. $v^L_{\max} = 0.7m/s$ in coverage). In this case, no coverage objective is active: the goal of this experiment is to stress the event-triggered logic under rapidly changing leader trajectories.

Table 6.2 summarizes the NMPC and event-trigger parameters for experiments A, B, and C that have been used for the simulation purposes.

Table 6.2: NMPC and Event-Trigger Parameters

Parameter	**Symbol**	**Value**
NMPC		
Prediction horizon	H_p	20
Sampling time	T_s	0.1s
Velocity time constant	τ	0.3s
Max. velocity	$v_{\max}$	1.0m/s
Leader velocity	$v^L_{\max}$	0.7m/s
Safety distance	d_{safe}	2.0m
Safety factor / scaling	$\alpha,\ L$	0.5, 0.5
Cost weights — Experiments B & C		
Position / velocity / input	$Q_p,\ Q_v,\ R$	5, 2, 1
Smoothness / terminal	$S,\ \gamma_e$	5, 5.0
Cost weights — Experiment A		
XY / Z tracking	$w_{xy},\ w_z$	10, 2
Coverage altitude / BW slack	$w_{\text{cov}},\ w_{\text{bw}}$	5, 1000
Bitrate coupling	α_b	0.05
BW altitude range	$[z^{\text{bw}}_{\min}, z^{\text{bw}}_{\max}]$	$[0.5, 5.0]$m
Event trigger		
Mandatory broadcast	$T_{\max}$	20 steps (2s)
Regularization	δ	10^{-6}

6.6.4 Metrics

We evaluate the performance of the proposed approach through the following metrics:

- **Communication rate** ρ: the ratio between number of broadcast messages and the total number of possible message exchanges (all agents at every step), averaged over all Monte Carlo runs. $\rho = 1$ corresponds to time-triggered DMPC.

- **Minimum inter-agent distance**:

  $d_{min} = \min_{k,i\neq j} ||\mathbf{p}_i(k) - \mathbf{p}_j(k)||_2$, over the entire simulation, to verify collision safety.

- **Formation Root Mean Square Error (RMSE)**:

  $\text{RMSE} = \sqrt{\frac{1}{(N-1)T}\sum_{i=2}^{N}\sum_{k=1}^{T}\|\mathbf{p}_i(k) - \mathbf{p}_i^{\text{des}}(k)\|^2}$, measuring the average position error of all followers relative to their desired formation position.

6.6.5 Communication Reduction

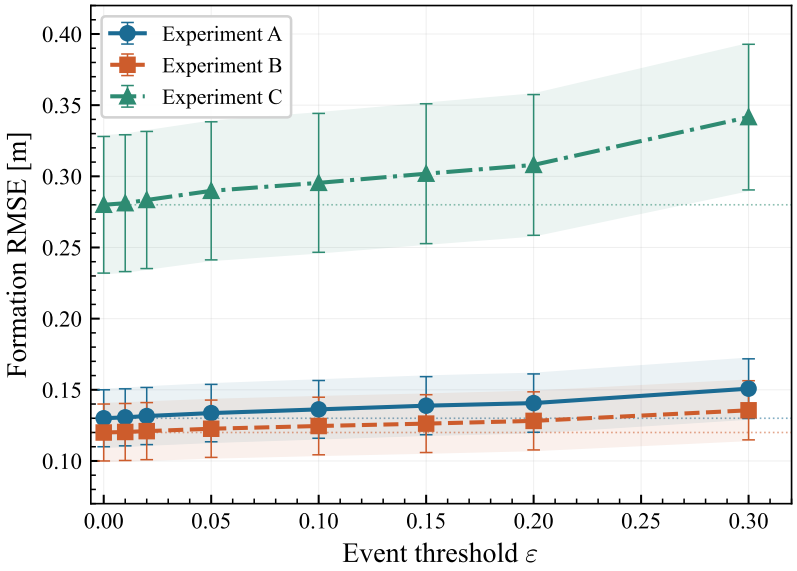


Figure 6.6.3: Communication rate ρ as a function of event threshold ε for $N = 5$ across the three experiments. Error bars indicate ± 1 standard deviation over $K = 50$ Monte Carlo runs.

Fig. 6.6.3 presents the communication rate ρ as a function of the event threshold

ε for $N = 5$ UAVs. The results demonstrate a consistent and significant reduction in communication across all experiments. In Experiment A, the communication rate drops from $\rho = 1.0$ at $\varepsilon = 0$ to approximately $\rho = 0.30$ at $\varepsilon = 0.05$ and $\rho = 0.15$ at $\varepsilon = 0.20$. Despite the additional complexity of altitude adaptation, which causes time-varying formation that could trigger more broadcasts, the long straight segments of the lawnmower path remain highly predictable, and the trigger fires mainly during 180° turns.

Experiment B shows slightly better communication reduction than A, reaching $\rho \approx 0.12$ at $\varepsilon = 0.20$, since the constant altitude avoids the broadcast bursts caused by altitude transitions. This confirms that bandwidth-aware altitude adaptation causes some communication overhead, but the event-triggered mechanism effectively dampens it.

Experiment C exhibits a slower decline, reaching $\rho \approx 0.45$ at $\varepsilon = 0.20$. The continuously changing leader trajectory causes more frequent triggering events. Nevertheless, even in this challenging experiment, over 50% of the communications are eliminated.

6.6.6 Formation tracking accuracy

Fig. 6.6.4 shows the formation RMSE as a function of ε. Across all experiments, the RMSE remains remarkably stable for threshold values up to $\varepsilon = 0.10$, with less than 3% degradation relative to the time-triggered baseline. At $\varepsilon = 0.20$, the RMSE increase is approximately 5–8%, which remains acceptable for most practical applications. Beyond $\varepsilon = 0.25$, the degradation accelerates as the followers increasingly rely on old trajectory information.

The absolute RMSE values at $\varepsilon = 0$ (baseline) are 0.13 ± 0.02m (Experiment A), 0.12 ± 0.02m (Experiment B), and 0.28 ± 0.05m (Experiment C). Experiment A has slightly higher baseline RMSE than B due to the dynamic altitude changes and time-varying formation geometry introduced by bandwidth adaptation. Experiment

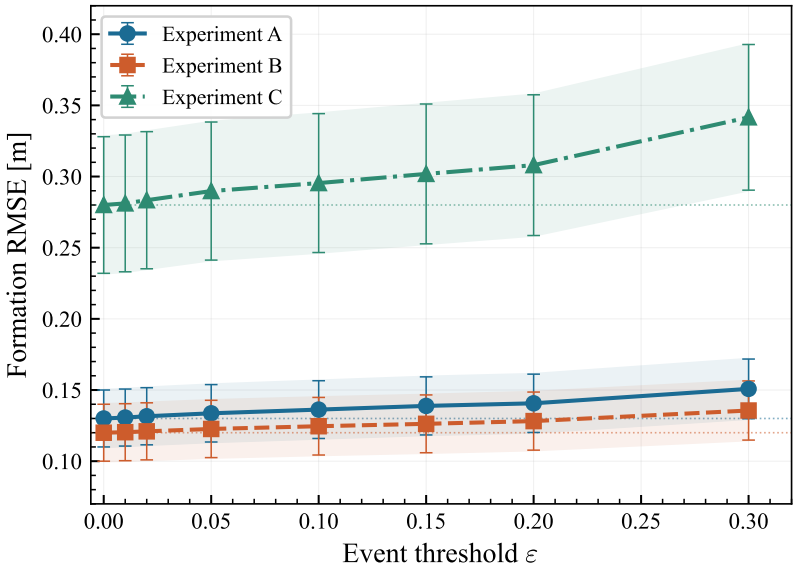


Figure 6.6.4: Formation RMSE as a function of event threshold ε for $N = 5$. The dashed horizontal line indicates the baseline TT-DMPC RMSE.

C has the highest RMSE due to the aggressive trajectory, as expected. All values are well within acceptable bounds for multi-UAV formation missions where safety distances are on the order of 2–3m.

6.6.7 Communication vs accuracy

The Pareto front in Fig. 6.6.5 shows a clear "knee" visible in all experiments around $\rho \approx 0.20$–0.30, corresponding to $\varepsilon \approx 0.05$–0.10. At this operating point, the communication rate is reduced by 70–80% while the formation RMSE increases by less than 5%. This highlights that most trajectory exchanges in standard DMPC are redundant.

6.6.8 Comparison with periodic baselines

Fig. 6.6.6 compares ET-DMPC against fixed-rate baselines that broadcast every $k \in \{5, 10, 20\}$ steps. At a comparable communication rate ($\rho \approx 0.20$), the periodic baseline with $k = 5$ achieves a formation RMSE of 0.24 ± 0.04m, while the ET-

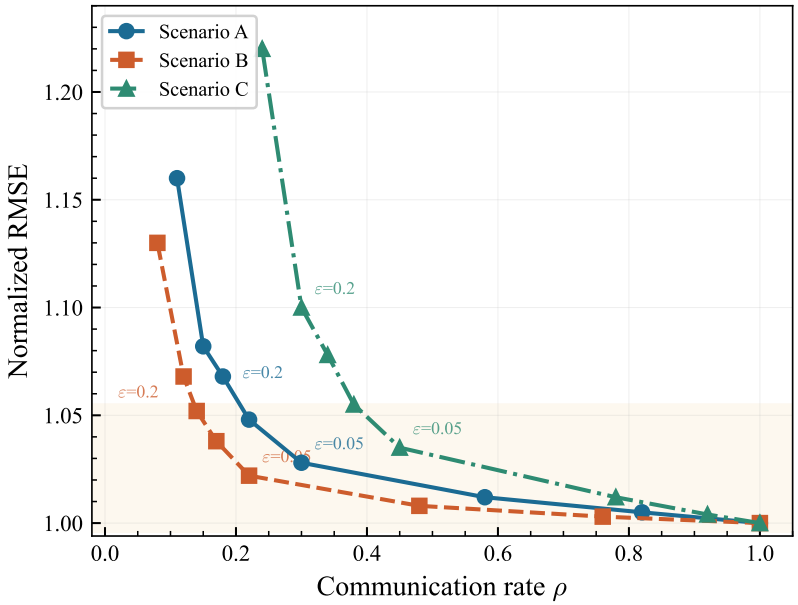
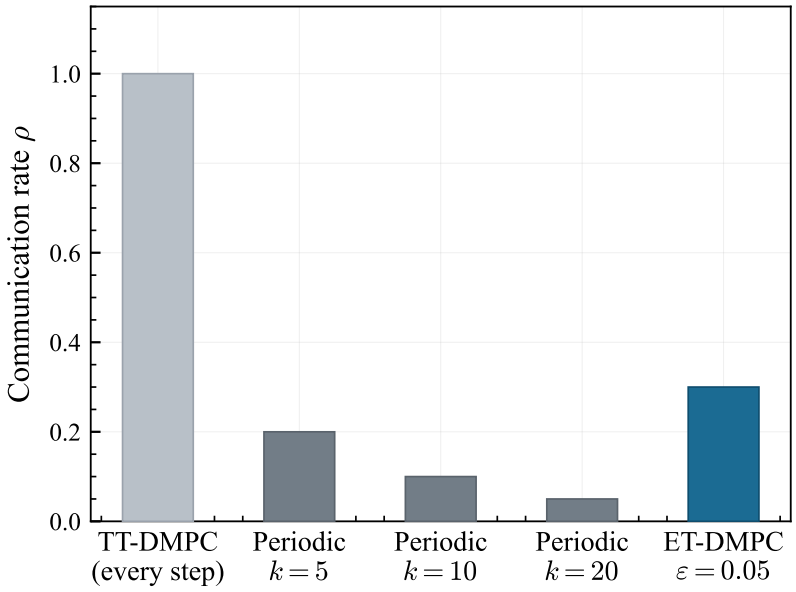


Figure 6.6.5: Pareto front: normalized formation RMSE vs. communication rate ρ for $N = 5$. The "knee" region (shaded) indicates the recommended operating regime with 70–80% communication reduction and $< 5\%$ RMSE degradation.

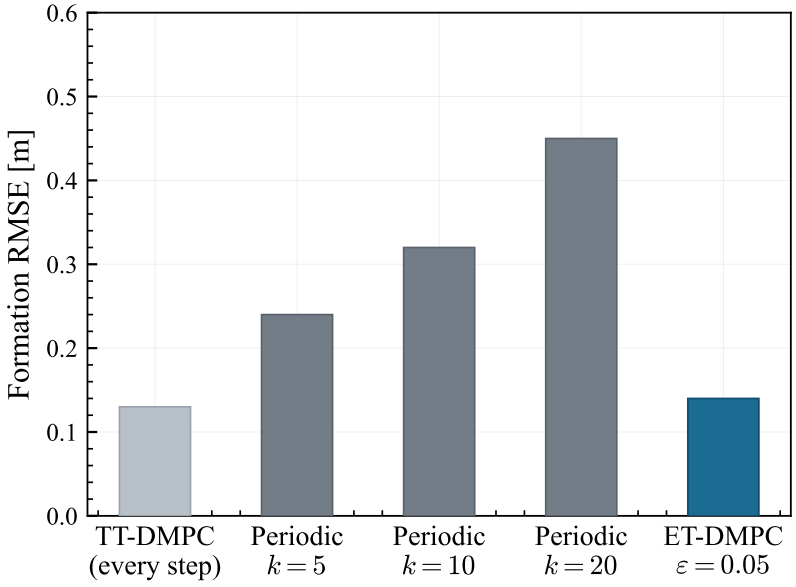
DMPC at $\varepsilon = 0.05$ achieves 0.14 ± 0.02m, a 42% improvement. This advantage is especially visible in the bandwidth-aware experiment (A): the event trigger adapts to both horizontal turns *and* altitude transitions, broadcasting during rapid changes and staying silent during stable cruise. The periodic scheme, in contrast, wastes bandwidth during predictable segments and provides stale information during critical maneuvers.

6.6.9 Collision Safety

Fig. 6.6.7 shows the minimum inter-agent distance for the worst-case experiment (Experiment C, $N = 8$, $\varepsilon = 0.10$). Across all 50 Monte Carlo runs, the minimum distance never falls below $d_{\text{safe}} = 2m$. The closest approach observed is $d_{\min} = 2.31m$, providing a margin of $0.31m$ above the safety constraint. For higher thresholds ($\varepsilon = 0.20$), the margin decreases to approximately $0.18m$, confirming that the conservative safety distance is essential.



(a) Communication Rate



(b) Formation Tracking Error

Figure 6.6.6: Comparison of ET-DMPC ($\varepsilon = 0.05$) against periodic communication baselines and TT-DMPC for Experiment A with $N = 5$.

6.6.10 Trigger events analysis

Figure 6.6.8 illustrates the binary distribution of trigger events for a typical Experiment A execution. While communication remains minimal during linear segments,

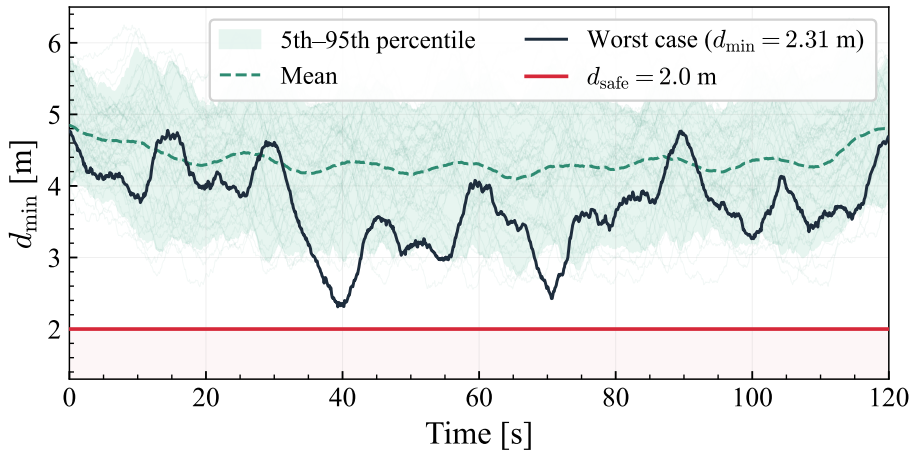


Figure 6.6.7: Minimum inter-agent distance over time for Experiment C with $N = 8$ and $\varepsilon = 0.10$. All $K = 50$ Monte Carlo runs maintain $d_{min} > d_{safe}$.

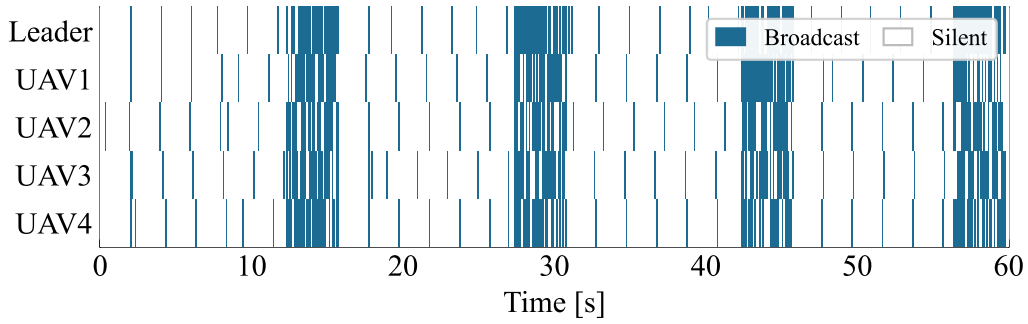
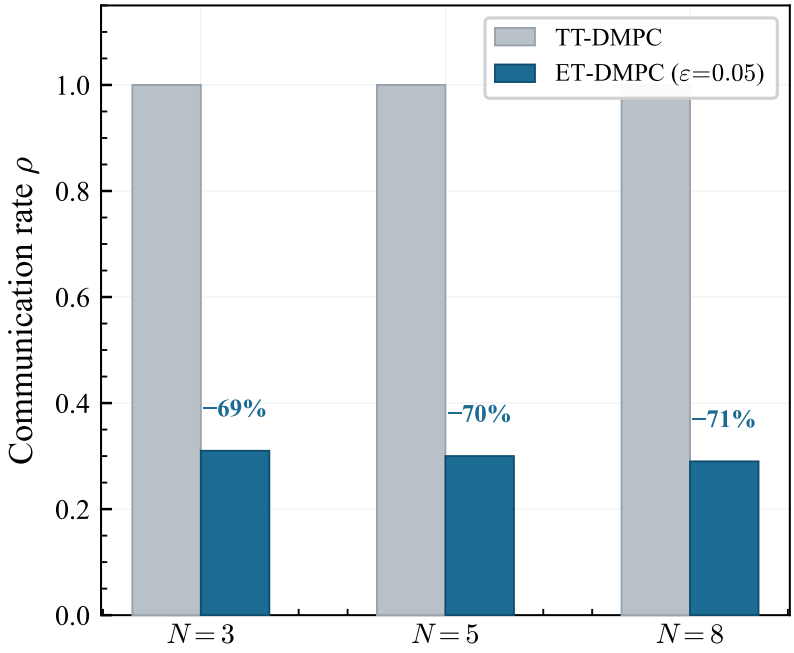
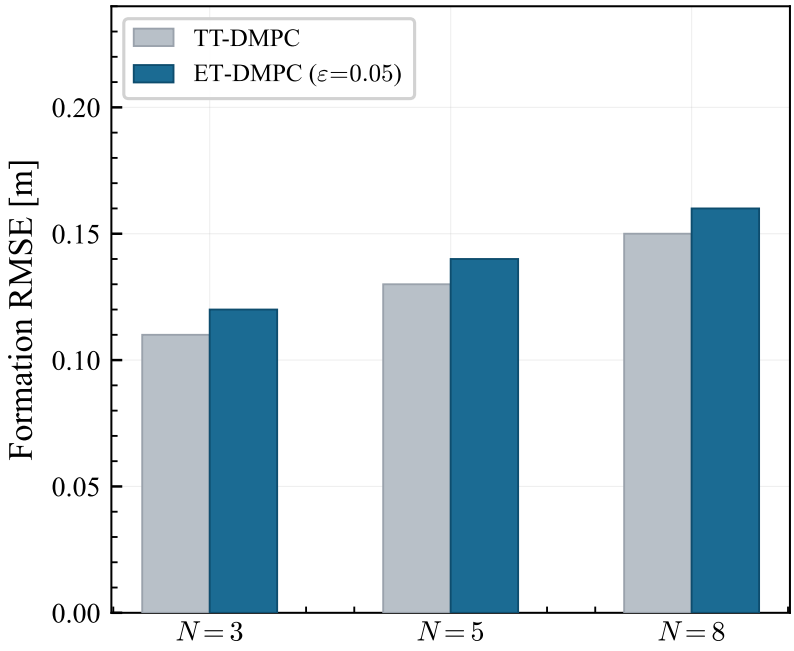


Figure 6.6.8: Event trigger heatmap for Experiment A with $N = 5$ and $\varepsilon = 0.05$. Blue cells indicate broadcast events.

driven largely by the mandatory $T_{\max}$ limit, triggering becomes nearly universal across the swarm during directional and vertical transitions. The disproportionate trigger frequency of the leader suggests that a heterogeneous thresholding approach could further enhance communication efficiency. Furthermore, while bandwidth-constrained altitude adjustments generate distinct trigger clusters not seen in the baseline (Experiment B), their density remains secondary to those triggered by maneuvers.



(a) Communication Rate



(b) Formation Tracking Error

Figure 6.6.9: Scalability analysis: communication rate and formation RMSE for $N \in \{3, 5, 8\}$ in Experiment A.

Table 6.3: Average NMPC solve time per agent (ms)

	$N = 3$	$N = 5$	$N = 8$
NMPC solve (Acados SQP-RTI)	1.2 ± 0.8	1.7 ± 1.2	2.1 ± 1.4
Trigger evaluation	< 0.1	< 0.1	< 0.1
Total	1.3	1.8	2.2

6.6.11 Scalability

Fig. 6.6.9 examines scalability. For fixed $\varepsilon = 0.05$, the communication rate ρ remains approximately constant across $N \in \{3, 5, 8\}$, at around 0.30 ± 0.03. The absolute number of avoided transmissions grows linearly with N. The formation RMSE shows a modest increase from $0.12 \pm 0.02m$ at $N = 3$ to $0.16 \pm 0.03m$ at $N = 8$, traceable to increased formation complexity. The same trend is observed in the Time-Triggered Distributed Model Predictive Control (TT-DMPC) baseline.

6.6.12 Computational Time

Table 6.3 reports the average NMPC solve time per agent. The event-triggered logic introduces negligible computational overhead ($< 0.1ms$), while the NMPC solve time remains within the real-time bound of $T_s = 100ms$ for all tested configurations. The low-level controller adds negligible overhead ($< 0.01ms$ per call at 120 Hz).

6.6.13 Summary of Results

Table 6.4: Summary of results at $\varepsilon = 0.05$, $N = 5$

	Exp. A	Exp. B	Exp. C
Comm. Rate ρ	0.30	0.22	0.45
Comm. Reduction	70%	78%	55%
RMSE (m)	0.14	0.13	0.30
RMSE vs. baseline	+4.1%	+2.8%	+3.5%
$d_{\min}$ (m)	2.42	2.48	2.31
Safety margin (m)	0.42	0.48	0.31

Table 6.4 summarizes the key results at the recommended operating point $\varepsilon = 0.05$.

Chapter 7

Grid-Based Coverage

7.1 Grid-based Coverage Methodology

In this section, another approach for the same general scenario has been proposed aiming at maximizing the coverage area of the drones, utilizing a second order dynamic, and providing a mathematical proof for the provided algorithm. The motivation for this approach is to increase the coverage area while keeping a polygon shaped form for the movement of the agents. A nonlinear controller has been chosen as a candidate for each drone to steer them in a way that the coverage gets maximized without any detachments. Again, the leader-follower approach has been proposed where the leader agent follows a predefined path and the other agents keep a safe distance from it and the other drones to not have collision as well as providing the required overlapped videos. A wide range of applications can be defined based on the proposed framework, where a specific parameter of the system needs to track a reference signal due to a disturbance which forces this reference signal to the system. A key feature of the proposed framework is that the altitude of each agent can vary according to a measurable local disturbance that depends both on position and time. Such disturbances include: network bandwidth for data or video streaming applications, heat and smoke levels for fire monitoring missions [70], or

wind strength for stability considerations. In each case, adjusting the altitude allows agents to maintain mission effectiveness while adapting to local environmental conditions. The main contributions of this work are: (i) the definition of an *admissible grid formation* that formalizes the spatial patterns and neighbor relationships described above; (ii) decentralized reference generation rules that compute each agent's target position to maintain a desired overlap with neighbors, even when footprint sizes vary; (iii) a formal proof of velocity-bounded asymptotic tracking for the proposed nonlinear controller.

7.2 Problem Formulation - Nonlinear Controller

In the first place, consider N agents associated with a variable FoV whose shape, as already explained, is assumed to be a square such that the position of the agent is exactly at the center. Such an area could represent the portion of the ground framed by a camera mounted on a drone, the portion of the seabed captured by a sonar mounted on a Unmanned Underwater Vehicle (UUV), etc. For simplicity we assume that each agent is modelled as a double integrator in Eq. (7.1).

$$\dot{\mathbf{s}}_1 = \mathbf{s}_2, \quad \dot{\mathbf{s}}_2 = \mathbf{u} \tag{7.1}$$

where $\mathbf{s}_1 = (x, y, z)^T$ is the vector of the position coordinates of the agent, $\mathbf{s}_2 = (v_x, v_y, v_z)$ is the vector of velocities, and $\mathbf{u} = (u_x, u_y, u_z)^T$ is the control input (acceleration) for the agent. The goal we want to achieve is the following: a certain number N of agents, starting at time $t = 0$ from an initial configuration in the space with the same initial altitude, i.e. the same initial scanned area, is required to move forward and cover an area along a given reference path. The agents are required to move together as a swarm while avoiding detachments between their scanned areas to avoid uncovered spots. To this purpose, we consider a leader-follower setup for the agents, where the leader is placed without loss of generality at the center of the

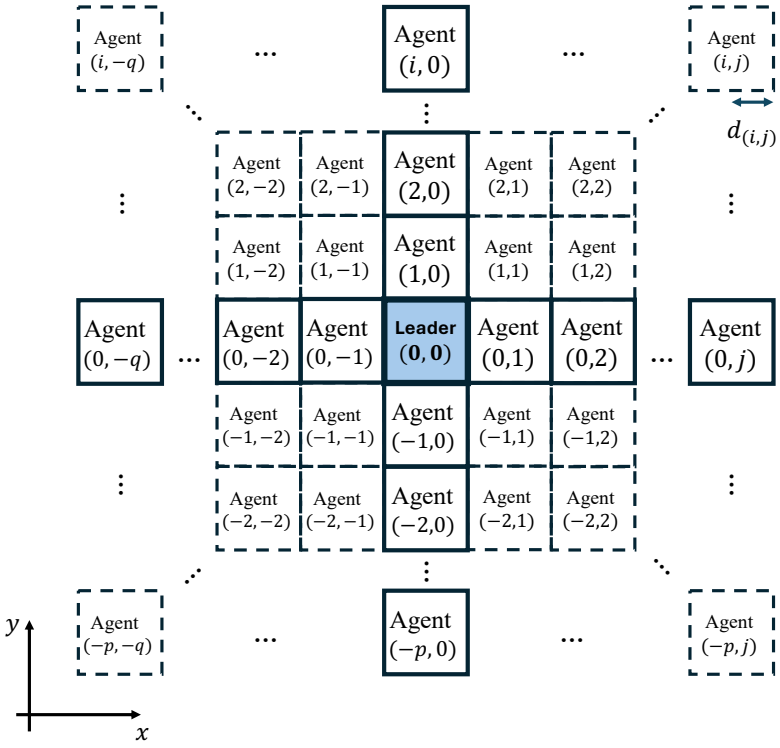


Figure 7.2.1: General grid configuration

swarm. The other agents are distributed around the leader in a specific initial formation such that there are no detachments and that the square FoVs are arranged in a grid-like fashion.

Figure 7.2.1 shows the most general initial configuration, along with the notation employed, where the leader (blue tile) is in the center, whereas the other agents are the followers. Such agents are identified by a couple indicating the row and the column. In particular, agents (i, j) with $i, j \geq 0$ are located in the top-right quadrant, agents $(i, -q)$ with $i, q \geq 0$ in the top-left quadrant, agents $(-p, -q)$ with $p, q \geq 0$ in the bottom-left quadrant, and agents $(-p, j)$ with $p, j \geq 0$ in the bottom-right quadrant. Finally, we denote with $d_{(i,j)}$ the variable length of half the side of the square FoV associated to agent (i, j). The relationship between $d_{(i,j)}$ and the altitude of the drone $z_{(i,j)}$ is defined as follows [43]:

$$d_{(i,j)} = \tan\left(\frac{\phi}{2}\right) z_{(i,j)} \tag{7.2}$$

where ϕ is the observation camera angle.

The configuration in Fig. 7.2.1 will be referred to as *general grid configuration*. All the other admissible configurations are subsets of this one. In particular, a configuration is considered to be admissible for the proposed approach if it is built according to the following rules.

- It is possible to freely add agents of the type $(0, j), (0, -q), (i, 0), (-p, 0)$, i.e. along the horizontal or vertical axis of the leader.
- Agent $(i, j), i, j > 0$ can be added as long as agents $(i - 1, j)$ and $(i, j - 1)$ are already present (first quadrant).
- Agent $(i, -q), i, q > 0$ can be added as long as agents $(i-1, -q)$ and $(i, -q+1)$ are present (second quadrant).
- Agent $(-p, -q), p, q > 0$ can be added as long as agents $(-p + 1, -q)$ and $(-p, -q + 1)$ are present (third quadrant).
- Agent $(-p, j), p, j > 0$ can be added as long as agents $(-p+1, j)$ and $(-p, j+1)$ are present (fourth quadrant).

Definition 1. An *admissible initial grid configuration* is a set of agents positioned according to the aforementioned rules.

Based on this definition, two important observations can be drawn. In the first place, the configuration does not have to be symmetric with respect to the number of agents. Secondly, although not mandatory, it is preferable to equally distribute the followers around the leader in order to obtain a wider area coverage around the the path tracked by the leader.

Notice that the general grid configuration in Figure 7.2.1 depicts the case of no overlap between the agents. Since the amount of overlap is a tunable parameter, if it is different from zero, then the general or admissible grid configuration built according to Definition 1 would show overlapped FoVs.

In order for the swarm to track a desired path while keeping the initial grid-like admissible configuration at each time-step, it is important for each agent to compute a proper reference position in the space to be reached at the next time-step. To this purpose, let us start by showing how such reference points are computed at each agent and then we design proper controllers that allow the agents to reach the reference points. In particular, we prove that the proposed control laws guarantee convergence of the agents to the desired reference points. It is important to remark that trajectories and reference points associated with each agent refer to the center of its FoV, which is assumed to be a square, where the agent is located. Moreover, recall that the area of the square FoV of each agent changes *independently* according to a disturbance (f.i., the available bandwidth) measured in that specific location. Thus, as the agents move to follow the leader, the altitude, along with the FoVs of the agents, change and the grid configuration varies over time while striving to avoid detachments. Notice that, although the shape of the scanned area of the swarm changes, the designed control technique keeps a constant distribution of the agents around the leader. In other words, the row and column identifying the location of an agent in the swarm remain unchanged, as explained further on. Finally, we make the realistic assumption that all the agents are equipped with a mobile network connection, thus they can communicate with each other.

For each agent, the reference altitude $z^{\mathbf{r}}$ on the z-axis depends on the available network bandwidth and is provided by a specific mapping that has been discussed in Eq. (4.19). Let us now show how to compute the references on the (x, y)-plane to be tracked by the agents.

The leader's reference point $(x_{\mathrm{L}}, y_{\mathrm{L}})$ belongs to the given desired path. Notice

that the design of an online path planner that computes the reference point for the leader at each time-step is orthogonal to this work. Regarding agents $(0, j), j > 0$, the reference point at the next time-step depends only on the agent on its left, i.e. agent $(0, j-1)$. Therefore, the reference $(x^{\mathbf{r}}_{(0,j)}, y^{\mathbf{r}}_{(0,j)})$ for an agent $(0, j), j > 0$ is:

$$x^{\mathbf{r}}_{(0,j)} = x_{(0,j-1)} + d_{(0,j-1)} + d_{(0,j)} - \gamma \tag{7.3}$$

$$y^{\mathbf{r}}_{(0,j)} = y_{\mathrm{L}} \tag{7.4}$$

where $(x^{\mathbf{r}}_{(0,j)}, y^{\mathbf{r}}_{(0,j)})$ is the desired position, i.e. the center of the square FoV, of agent $(0, j)$ at the next time-step, $d_{(0,j-1)} + d_{(0,j)}$ is the distance between agent $(0, j-1)$ and $(0, j)$ in the x direction, which was previously defined, $\gamma \geq 0$ is a positive real parameter that denotes the desired overlap ($\gamma = 0$ means no overlap). y_{L} is the y-coordinate of the leader at the next time-step, which is shared with the leader's neighbors as soon as it is available.

Regarding agents $(i, 0), i > 0$, the reference point depends only on agent $(i-1, 0)$. Thus, the reference $(x^{\mathbf{r}}_{(i,0)}, y^{\mathbf{r}}_{(i,0)})$ for an agent $(i, 0), i > 0$ is:

$$x^{\mathbf{r}}_{(i,0)} = x_{\mathrm{L}} \tag{7.5}$$

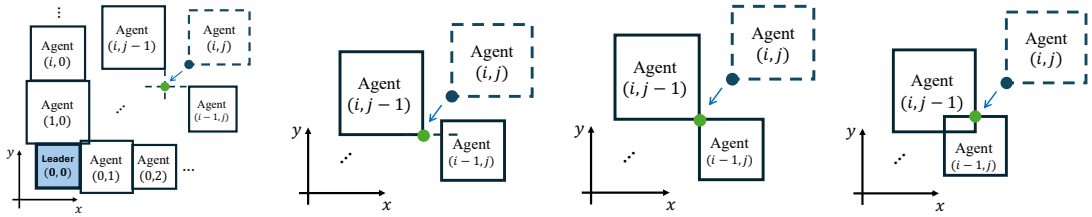
$$y^{\mathbf{r}}_{(i,0)} = y_{(i-1,0)} + d_{(i-1,0)} + d_{(i,0)} - \gamma \tag{7.6}$$

Analogously, for agent $(0, -q), q > 0$, the reference point is given by:

$$x^{\mathbf{r}}_{(0,-q)} = x_{(0,-q+1)} - d_{(0,-q+1)} - d_{(0,-q)} + \gamma \tag{7.7}$$

$$y^{\mathbf{r}}_{(0,-q)} = y_{\mathrm{L}} \tag{7.8}$$

In other words, to compute the reference point, this time we take into account the neighboring agent on the right of agent $(0, -q)$, i.e. agent $(0, -q+1)$. Finally, given



(a) Case 1 (b) Case 2 (c) Case 3 (d) Case 4

Figure 7.2.2: Cases for reference points

agent $(-p, 0), p > 0$, the reference point is:

$$x^{\mathbf{r}}_{(-p,0)} = x_{\mathrm{L}} \tag{7.9}$$

$$y^{\mathbf{r}}_{(-p,0)} = y_{(-p+1,0)} - d_{(-p+1,0)} - d_{(-p,0)} + \gamma \tag{7.10}$$

For a generic agent $(i, j), i, j > 0$, the computation of the reference point depends on the location of agents $(i-1, j)$, i.e. the neighbor below, and $(i, j-1)$, i.e. the neighbor on its left. In particular, different cases may occur. The first case is shown in Figure 7.2.2(a). It would be desirable to set the reference point, which refers to the center of the square, such that the lower-left corner of the agent's FoV is on the green dot. Figures 7.2.2(b)-7.2.2(d) show the other possible cases that may occur where the green dot represents the point where to drive the agent's lower-left corner to avoid detachments in the overall coverage. Notice that these figures show the case where γ is set to 0. As γ grows, the green dot is pushed down and left.

The reference point of agent $(i, j), i, j > 0$, for the cases depicted in Figures 7.2.2(a)-7.2.2(c) is:

$$x^{\mathbf{r}}_{(i,j)} = \min\left(x_{(i,j-1)} + d_{(i,j-1)},\ x_{(i-1,j)} - d_{(i-1,j)}\right) + d_{(i,j)} - \gamma \tag{7.11}$$

$$y^{\mathbf{r}}_{(i,j)} = \min\left(y_{(i,j-1)} - d_{(i,j-1)},\, y_{(i-1,j)} + d_{(i-1,j)}\right) + d_{(i,j)} - \gamma \tag{7.12}$$

For the case in Figure 7.2.2(d), the computation changes as follows:

$$x^{\mathbf{r}}_{(i,j)} = \max\left(x_{(i,j-1)} + d_{(i,j-1)},\, x_{(i-1,j)} - d_{(i-1,j)}\right) + d_{(i,j)} - \gamma \tag{7.13}$$

$$y^{\mathbf{r}}_{(i,j)} = \max\left(y_{(i,j-1)} - d_{(i,j-1)},\, y_{(i-1,j)} + d_{(i-1,j)}\right) + d_{(i,j)} - \gamma \tag{7.14}$$

Notice that $d_{(i,j)}$ is added to pass from the coordinates of the green dot to the coordinates of the center of the FoV associated with agent (i, j). In the same way, the computation of the reference point for a generic agent $(i, -q)$, $(-p, -q)$, or $(-p, j)$, with $i, q, p > 0$, follows a similar logic. This formulation of the reference points avoids the formation of blind spots and imposes an overlap between the scanned areas that is proportional to γ. In the following, the reference points for the remaining agents with respect to their neighbors are computed. For agents $(i, -q)$, with $i, q > 0$, and without overlap between the neighboring agents $(i, -q + 1)$ and $(i - 1, -q)$, the reference point is the following:

$$\begin{aligned} x^{\mathbf{r}}_{(i,-q)} = \max(&x_{(i,-q+1)} - d_{(i,-q+1)}, \\ &x_{(i-1,-q)} + d_{(i-1,-q)}) - d_{(i,-q)} + \gamma \end{aligned} \tag{7.15}$$

$$\begin{aligned} y^{\mathbf{r}}_{(i,-q)} = \min(&y_{(i,-q+1)} - d_{(i,-q+1)}, \\ &y_{(i-1,-q)} + d_{(i-1,-q)}) + d_{(i,-q)} - \gamma \end{aligned} \tag{7.16}$$

If there is overlap between the neighboring agents, it results:

$$x^{\mathbf{r}}_{(i,-q)} = \min(x_{(i,-q+1)} - d_{(i,-q+1)}, \\ x_{(i-1,-q)} + d_{(i-1,-q)}) - d_{(i,-q)} + \gamma \tag{7.17}$$

$$y^{\mathbf{r}}_{(i,-q)} = \max(y_{(i,-q+1)} - d_{(i,-q+1)}, \\ y_{(i-1,-q)} + d_{(i-1,-q)}) + d_{(i,-q)} - \gamma \tag{7.18}$$

For agents $(-p,-q)$ with $p,q > 0$, and without overlap between the neighboring agents $(-p,-q+1)$ and $(-p+1,-q)$ the reference point is:

$$x^{\mathbf{r}}_{(-p,-q)} = \max(x_{(-p,-q+1)} - d_{(-p,-q+1)}, \\ x_{(-p+1,-q)} + d_{(-p+1,-q)}) - d_{(-p,-q)} + \gamma \tag{7.19}$$

$$y^{\mathbf{r}}_{(-p,-q)} = \max(y_{(-p,-q+1)} - d_{(-p,-q+1)}, \\ y_{(-p+1,-q)} + d_{(-p+1,-q)}) + d_{(-p+1,-q)} - \gamma \tag{7.20}$$

When agents $(-p,-q+1)$ and $(-p+1,-q)$ overlap, it results:

$$x^{\mathbf{r}}_{(-p,-q)} = \min(x_{(-p,-q+1)} - d_{(-p,-q+1)}, \\ x_{(-p+1,-q)} + d_{(-p+1,-q)}) - d_{(-p,-q)} + \gamma \tag{7.21}$$

$$y^{\mathbf{r}}_{(-p,-q)} = \min(y_{(-p,-q+1)} - d_{(-p,-q+1)}, \\ y_{(-p+1,-q)} + d_{(-p+1,-q)}) + d_{(-p+1,-q)} - \gamma \tag{7.22}$$

Finally, for agents $(-p, j)$ when agents $(-p, j-1)$ and $(-p+1, j)$ do not overlap, we obtain:

$$x^{\mathbf{r}}_{(-p,j)} = \min(x_{(-p,j-1)} - d_{(-p,j-1)}, \\ x_{(-p+1,j)} + d_{(-p+1,j)}) + d_{(-p,j)} - \gamma \tag{7.23}$$

$$y^{\mathbf{r}}_{(-p,j)} = \max(y_{(-p,j-1)} - d_{(-p,j-1)}, \\ y_{(-p+1,j)} + d_{(-p+1,j)}) - d_{(-p,j)} + \gamma \tag{7.24}$$

If agents $(-p, j-1)$ and $(-p+1, j)$ overlap, we impose:

$$x^{\mathbf{r}}_{(-p,j)} = \max(x_{(-p,j-1)} - d_{(-p,j-1)}, \\ x_{(-p+1,j)} + d_{(-p+1,j)}) + d_{(-p,j)} - \gamma \tag{7.25}$$

$$y^{\mathbf{r}}_{(-p,j)} = \min(y_{(-p,j-1)} - d_{(-p,j-1)}, \\ y_{(-p+1,j)} + d_{(-p+1,j)}) - d_{(-p,j)} + \gamma \tag{7.26}$$

7.3 Nonlinear Controller Design

Now that the reference points for the agents have been proposed in all the dimensions, the nonlinear controller can be designed based on the given references to steer the agents toward the given points to follow the given path, and change the altitude accordingly. To assume bounded velocities for a more practical scenario, a saturation on such velocities are imposed through a state space transform. To this purpose, let us provide the following theorem.

Theorem. Consider a swarm of agents in $\mathbb{R}^3$, whose dynamics is as in Eq. (7.1) and

whose initial formation is any *admissible grid configuration*. Suppose that each agent is required to reach a desired reference point $(x^{\mathbf{r}}, y^{\mathbf{r}}, z^{\mathbf{r}})$ in the space with bounded velocities, i.e. such that $|v_x| < M$, $|v_y| < M$ and $|v_z| < M$, with $M \in \mathbb{R}^+$. Then, for any $K_d, K_p > 0$, the following controllers for $c = x, y$, and z:

$$\begin{aligned} u_c = &- K_d \frac{2M}{\pi} \sin\left(\frac{\pi}{2M} v_c\right) \cos\left(\frac{\pi}{2M} v_c\right) \\ &- \frac{2M}{\pi} K_p \cos^2\left(\frac{\pi}{2M} v_c\right) \frac{v_c}{\tan\left(\frac{\pi}{2M} v_c\right)} (c - c^{\mathbf{r}}) \end{aligned} \tag{7.27}$$

guarantee asymptotic convergence of the agent's position to $(x^{\mathbf{r}}, y^{\mathbf{r}}, z^{\mathbf{r}})$ with bounded velocities.

Proof. We prove the theorem for u_x, i.e. the first element of the control vector $\mathbf{u}$. The proof for u_y and u_z can be obtained in the same way.

We want to prove that, under the control u_x, (x, v_x) converge to the equilibrium point $(x^{\mathbf{r}}, 0)$. From Eq. (7.1), it can be seen that the system for the x coordinate has two states, i.e. x and v_x. While the first state has no constraints, the second one is bounded by M, i.e. $|v_x| < M$ with $M \in \mathbb{R}^+$. Let us now consider the following change of coordinates, which represents a mapping from the original state space to $\mathbb{R}^2$:

$$\xi_1 = x, \quad \xi_2 = \tan\left(\frac{\pi}{2M} v_x\right) \tag{7.28}$$

Hence, it follows that:

$$x = \xi_1, \quad v_x = \frac{2M}{\pi} \tan^{-1}(\xi_2) \tag{7.29}$$

From Eq. (7.29), it is evident that this change of coordinates keeps the velocity bounded as $|\frac{2M}{\pi} \tan^{-1}(\xi_2)| < M$.

The resulting model in the new coordinates is:

$$\dot{\xi}_1 = v_x = \frac{2M}{\pi}\tan^{-1}(\xi_2), \quad \dot{\xi}_2 = \frac{\pi/2M}{\cos^2(v_x \frac{\pi}{2M})}u_x \tag{7.30}$$

Using Eq. (7.28) and trigonometric identities, one can rewrite Eq. (7.30) as follows:

$$\dot{\xi}_1 = \frac{2M}{\pi}\tan^{-1}(\xi_2), \quad \dot{\xi}_2 = \frac{\pi}{2M}(1+\xi_2^2)u_x \coloneqq \nu \tag{7.31}$$

To prove asymptotic convergence to the desired reference position, consider the following positive definite candidate Lyapunov functional:

$$V(\boldsymbol{\xi}) = \frac{1}{2}\xi_2^2 + \frac{K_p}{2}(\xi_1 - \xi_1^{\mathbf{r}})^2 \tag{7.32}$$

The time derivative of $V(\boldsymbol{\xi})$ is equal to:

$$\begin{aligned} \dot{V}(\boldsymbol{\xi}) &= \xi_2\dot{\xi}_2 + K_p\dot{\xi}_1(\xi_1 - \xi_1^{\mathbf{r}}) \\ &= \xi_2\nu + \frac{2MK_p}{\pi}\tan^{-1}(\xi_2)(\xi_1 - \xi_1^{\mathbf{r}}) \end{aligned} \tag{7.33}$$

Let:

$$\nu = -K_d\xi_2 - \frac{2MK_p}{\pi}\frac{\tan^{-1}(\xi_2)}{\xi_2}(\xi_1 - \xi_1^{\mathbf{r}}) \tag{7.34}$$

Then:

$$\dot{V}(\boldsymbol{\xi}) = -K_d\xi_2^2 \leq 0 \tag{7.35}$$

which implies that $V(\boldsymbol{\xi})$ is negative semi-definite and is bounded, thus it has a finite limit as $t \to +\infty$. From Eq. (7.32), boundedness of $V(\boldsymbol{\xi})$ implies boundedness of ξ_2 and $\xi_1 - \xi_1^{\mathbf{r}}$, which, in turn, implies boundedness of ν in Eq. (7.34) and therefore of $\dot{\xi}_2$.

At this point, consider the second derivative of $V(\boldsymbol{\xi})$:

$$\ddot{V}(\boldsymbol{\xi}) = -2K_d\xi_2\dot{\xi}_2 = -2K_d\xi_2\nu \tag{7.36}$$

Since ξ_2 and ν are bounded, then $\ddot{V}(\boldsymbol{\xi})$ is bounded, i.e. $\dot{V}(\boldsymbol{\xi})$ is uniformly continuous. Moreover, remembering that $V(\boldsymbol{\xi})$ has a finite limit as $t \to +\infty$, by Barbalat's lemma it results that $\dot{V}(\boldsymbol{\xi}) \to 0, t \to +\infty$. This implies that $\xi_2 \to 0$, i.e. $v_x \to 0$ as desired. Now we have to prove that $\xi_1 \to \xi_1^{\mathbf{r}}$. To do so, consider:

$$\dot{\nu} = -K_d\dot{\xi}_2 - \frac{2MK_p}{\pi}\left[\frac{d}{d\xi_2}\left(\frac{\tan^{-1}(\xi_2)}{\xi_2}\right)(\xi_1 - \xi_1^{\mathbf{r}}) + \frac{\tan^{-1}(\xi_2)}{\xi_2}\dot{\xi}_1\right] \tag{7.37}$$

Since $\xi_2 \to 0$, then it is easy to show that

$$\frac{d}{d\xi_2}\left(\frac{\tan^{-1}(\xi_2)}{\xi_2}\right) \to 0 \tag{7.38}$$

Hence, $\dot{\nu} = \ddot{\xi}_2$ is bounded. Then, by Barbalat's lemma, it results that $\dot{\xi}_2 = \nu \to 0$, which implies $\xi_1 \to \xi_1^{\mathbf{r}}$, i.e. $x \to x^{\mathbf{r}}$. To conclude, by equating (7.31) and (7.34) it is possible to obtain u_x, i.e. the x-component of the control input (7.27).

□

Concerning the parameter γ regulating the overlap between two scanned areas, it is important to point out that its value should be chosen in such a way that potential detachments are avoided. Since detachments are caused by abrupt changes in speed on the $(x, y)-$plane or by abrupt changes in the width of the scanned area, which translates in sudden changes in the speed along the z-axis, the parameter γ should depend proportionally on the maximum value M that bounds the agent's velocities along the three axes. A formal analysis on such a dependency and a mathematical formulation of this parameter is left as a future work. In the following, γ will be set empirically.

7.4 Simulation Results for Grid-Based Coverage

This section presents the simulation results for the proposed control problem. We have considered a sine-shaped reference path for the leader to track whereas the other agents follow the leader while avoiding detachments.

We show the results for 9 agents[1] starting from the same altitude with the following admissible grid configuration: leader $(0, 0)$ and agents $(0, 1)$, $(0, -1)$, $(1, 0)$, $(-1, 0)$, $(1, 1)$, $(1, -1)$, $(-1, -1)$, $(-1, 1)$. Notice that guiding the agents of the swarm to form the initial admissible grid configuration used to start the coverage mission is the well-known standard rendez-vous problem. Each agent uses the controller provided in (7.27), where we have set $K_p = K_d = 5$. Notice that the controller works for any K_p and K_d as long as they are positive. The velocity limit is $M = 1\,m/s$ and the initial altitude for the agents is $6\,m$. For this simulation, $z_{\min}$ and $z_{\max}$ in (4.19) have been set to $6m$ and $15m$, respectively, whereas $b_{\min}$ and $b_{\max}$ have been set to $500\,\text{kbit/s}$ and $5000\,\text{kbit/s}$, respectively. Moreover, the bandwidth is modeled as a uniform random distribution over the area due to its unpredictability, the desired overlap γ has been set to 2, and finally an animation of the simulation has been produced[2].

7.4.1 Coverage and Trajectory Tracking

Let us now provide the results in terms of reference path tracking and coverage. To this end, we will consider only the projections on the (x, y)-plane. Figure 7.4.1 shows the case of 9 agents where the leader has to track a sine-shaped path (orange dashed line), as previously said. The starting point is marked with a black circle and the end point with a black star. It is possible to observe that the leader's trajectory (blue line) perfectly tracks the reference path. The other agents follow the leader, ensuring no detachments and maintaining a continuous coverage area (light blue area) without

[1] results for a different number of agents are similar

[2] https://www.youtube.com/watch?v=Wuikhf3PtWA

gaps. Notice that the followers' trajectories (shown in gray) are irregular because of the variable FoV depending on the altitude. This takes them to get closer or further from each other along the path to avoid detachments or collisions and to obtain overlap between their FoVs. The figure also shows snapshots of the swarm's area covered at $t = 10.4s$ and $t = 150s$, respectively. The '+' marker denotes the leader's position on the plane.

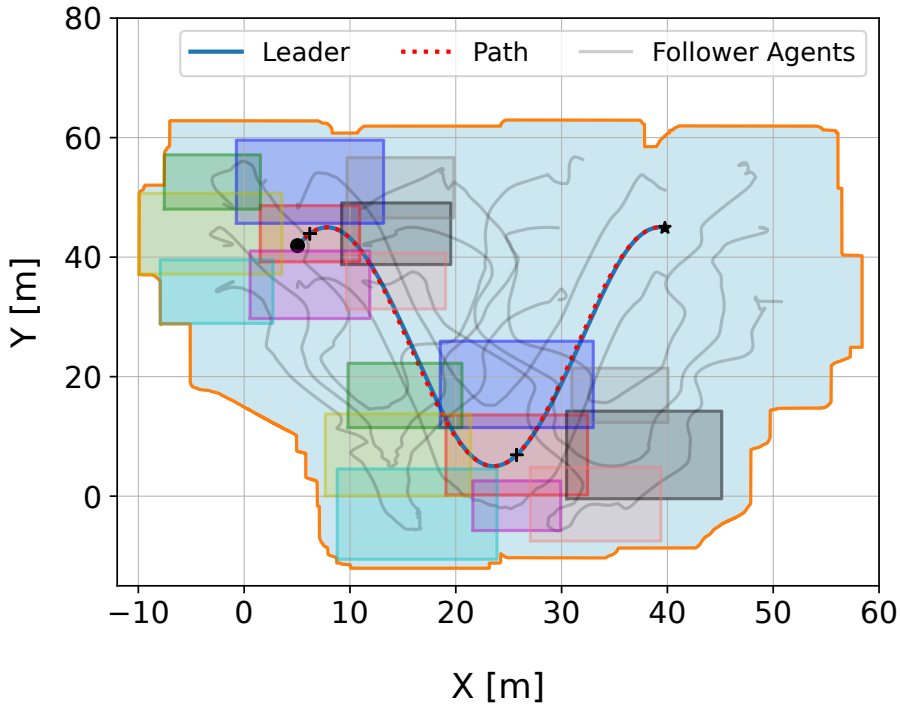


Figure 7.4.1: Coverage and trajectories of the swarm on the plane

7.4.2 Altitude Tracking and Control variables

Each agent is required to track reference altitude points computed on the basis of the available network bandwidth in their current position, as discussed in Eq. (4.19). Figure 7.4.2 shows the reference altitude trajectories are tracked accurately for each of the 9 agents.

The control inputs u_x, u_y, and u_z, i.e. the acceleration components, are shown in Figure 7.4.3 for all the agents. The smoothness of the control inputs can be regulated

by properly setting K_p and K_d. An important observation arises from Figure 7.4.4, which depicts the velocity components v_x, v_y, and v_z of the leader. It is possible to state that the three velocities are inside the bound set to $\pm M$ as desired. The same holds for the velocities associated with the other agents.

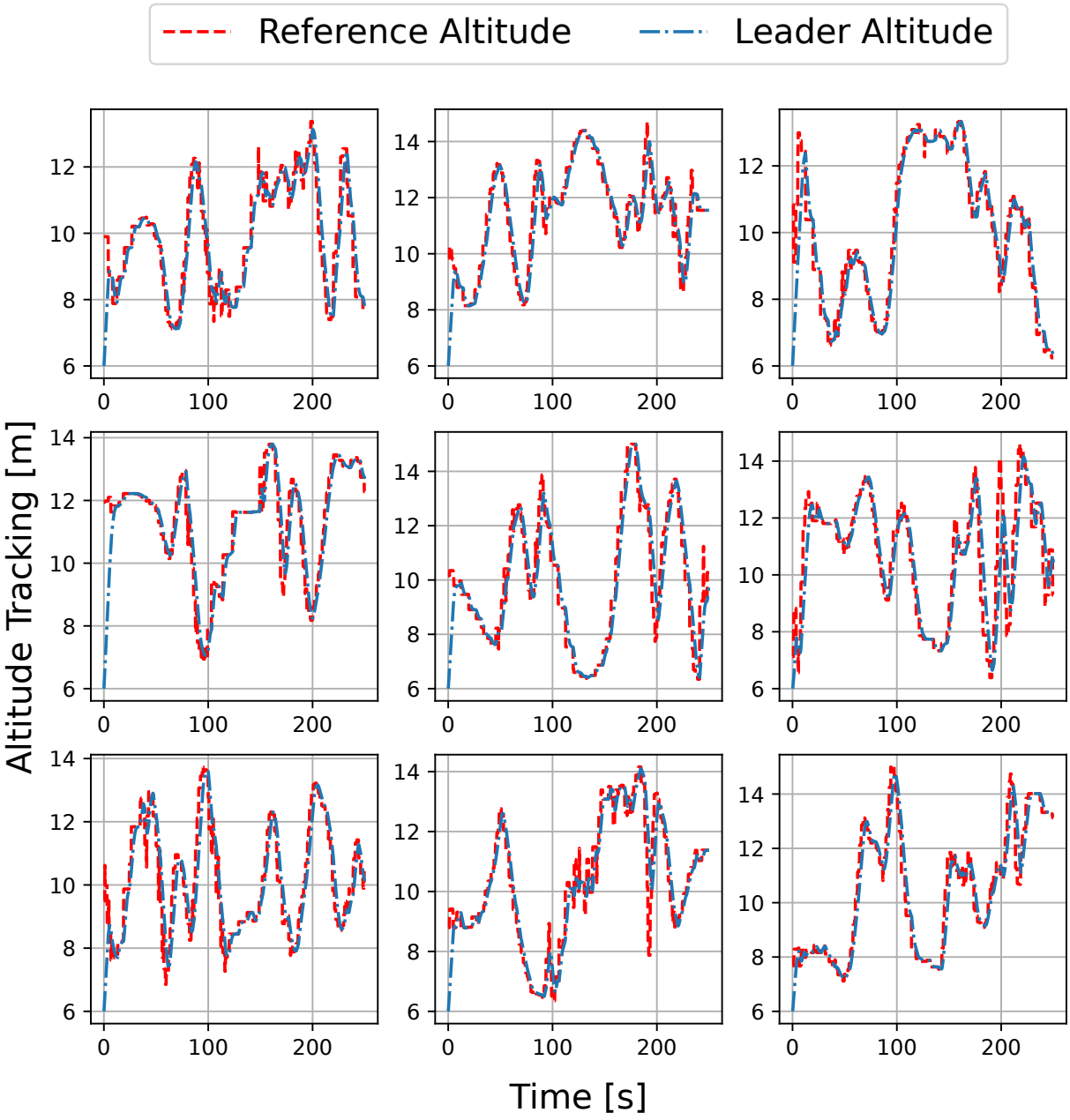


Figure 7.4.2: Altitude trajectory tracking for each agent

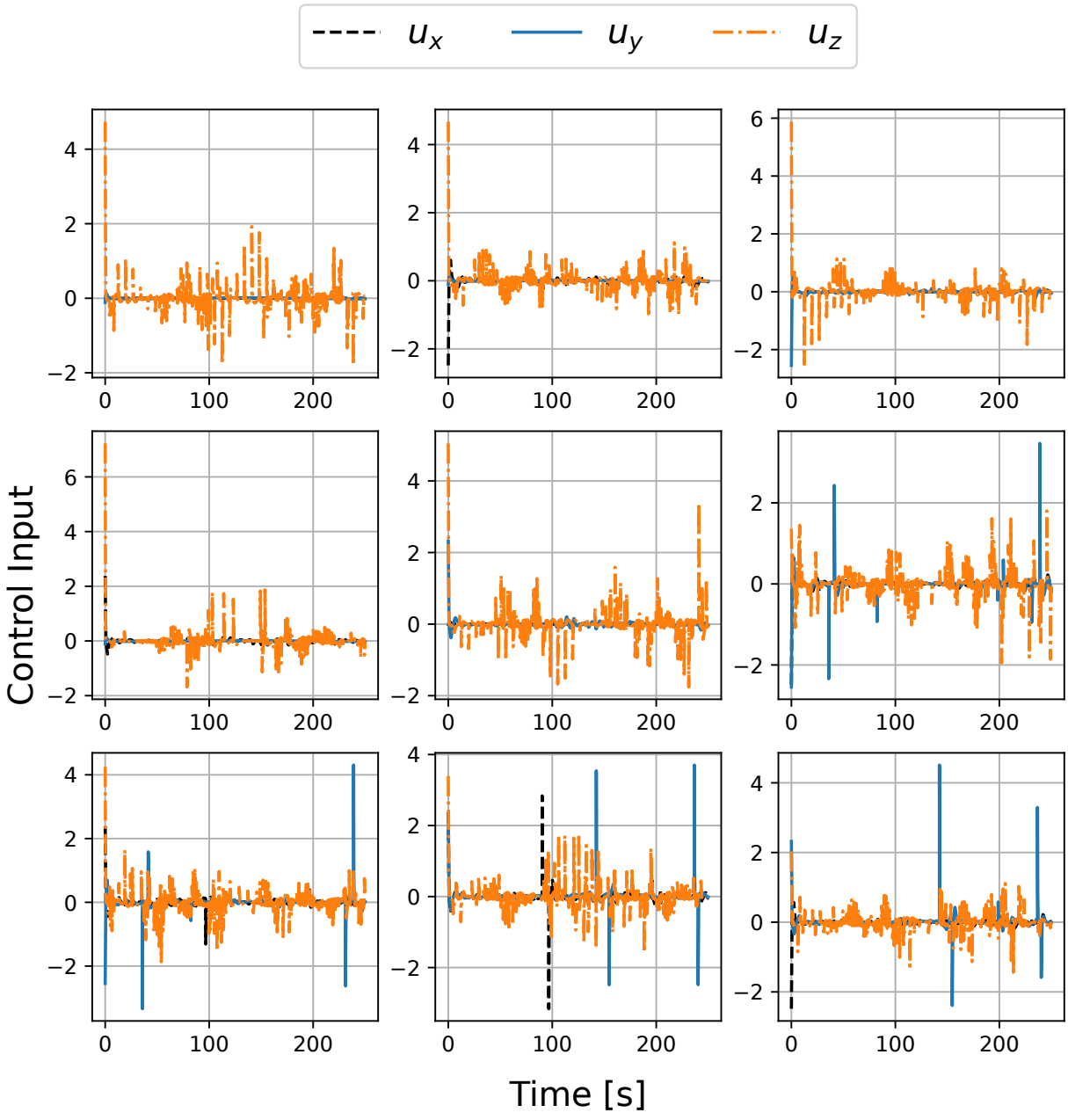


Figure 7.4.3: Control inputs for each agent

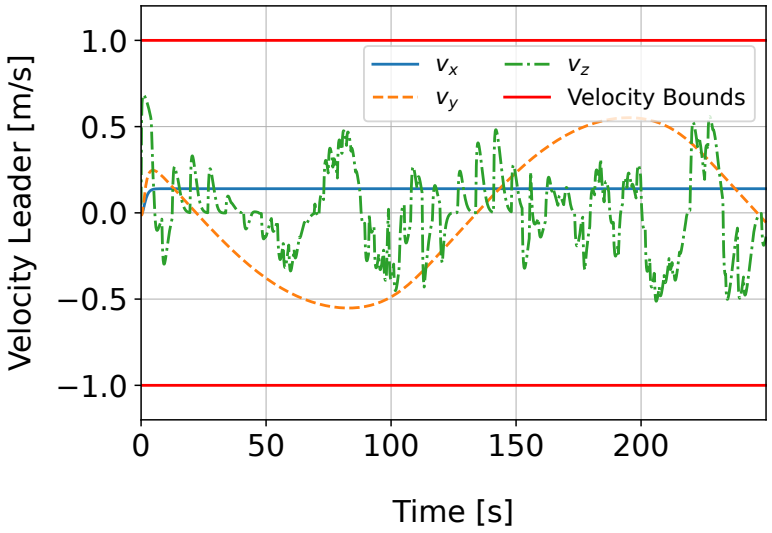


Figure 7.4.4: Velocity components of the leader

7.4.3 Comparison

In this section, a comparison with the methodology in 4.2 has been carried out, while considering 9 agents in the swarm. This comparison includes the total area coverage given the same path, and the circularity parameter (will be discussed in section 7.4.5), which shows how regularly the coverage area at each time-step is distributed around the current point on the reference path. Notice that, for a fair comparison, the available network bandwidth distribution is the same in both cases.

7.4.4 Coverage

As already said, the same Sine-like path considered in 4.2 is used for the proposed method in this paper for a fair comparison. Figure 7.4.5 shows the total areas covered employing the two aforementioned approaches. It can be observed that the area covered with the proposed method (orange border) is substantially greater than the coverage achieved with the approach in section 4.2 (light blue border), thus proving the effectiveness of the proposed method.

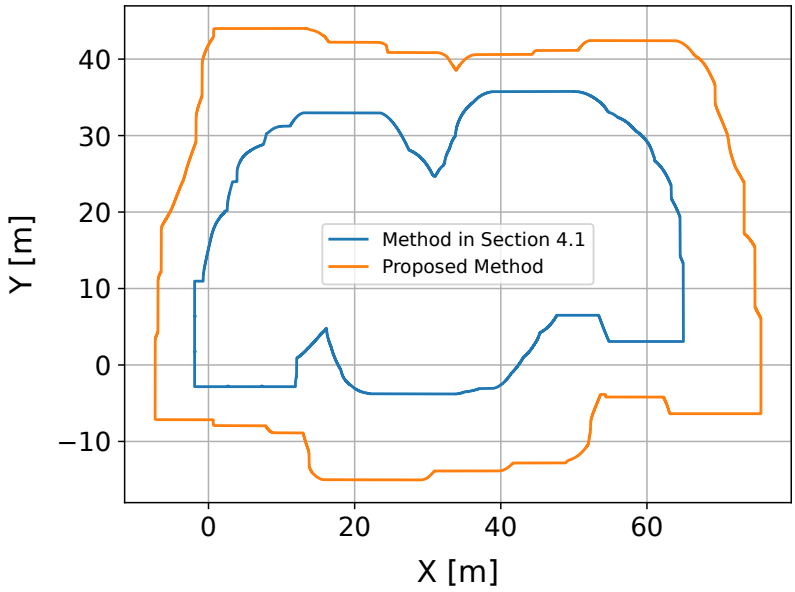


Figure 7.4.5: The comparison for the coverage area of the proposed method and the method in section 4.2.

7.4.5 Circularity

At each time-step, when the leader reaches a certain reference point along the given reference path, beside coverage, it is preferable that the followers are regularly distributed around it to provide satisfactory coverage of the surrounding area. To evaluate this aspect, let us define the circularity parameter as follows:

$$CP(t) = \frac{\min_{p \in E(t)}(\|x_L(t) - p\|)}{\max_{p \in E(t)}(\|x_L(t) - p\|)} \tag{7.39}$$

where $CP(t)$ is the circularity parameter, $x_L(t)$ is the leader's position, and $E(t)$ is the set of the points belonging to the boundary of the covered area at time t. Notice that this parameter changes over time. Based on Eq. (7.39), it clearly results that $0 < CP(t) \leq 1$. Moreover, if $CP(t) = 1$, it means that the area covered at time t is a perfect circle. On the other hand, if such a parameter decreases, it means that the swarm is monitoring an area around the leader that is irregularly distributed.

Therefore, the goal is to obtain high values of the circularity parameter during the mission.

Figure 7.4.6 displays the values of the circularity parameter over a 200s mission both for the proposed approach and the technique proposed in section 4.2, which does not aim at keeping a constant configuration but rather leverages an optimization problem to maximize coverage.

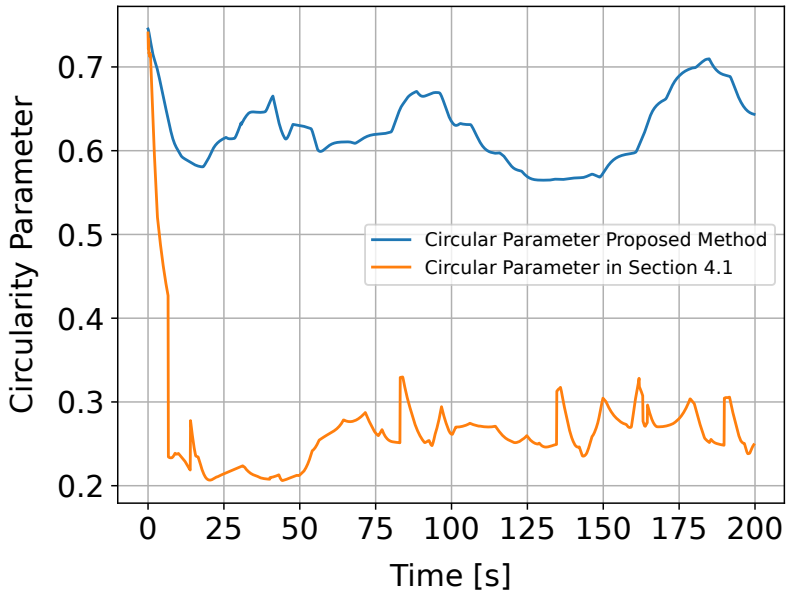


Figure 7.4.6: The comparison of the circular parameter.

It is possible to point out that the proposed approach achieves higher circularity with respect to the other method, thus implying a more regular distribution and coverage around the leader.

Part II

DC Islanded Grids

Chapter 8

Consensus Problem of Islanded Grids

8.1 Consensus in DC Islanded Grids

This section introduces the consensus algorithm used in this work, the microgrid's possible operating modes, and some useful remarks highlighting key aspects of the problem.

8.1.1 Consensus Algorithm

Consensus is a widely used branch of MAS in which the states of the agents reach the same value [31]. This state can represent a variety of concepts, including currents and voltages [58], the position of mobile robots [71], the position of drones [66], or even the human behavior of a society [72]. For a continuous-time system, the consensus algorithm is defined as follows.

Definition 2. A set of agents of a multi-agent system reaches consensus in their states if and only if:

$$\lim_{t\to\infty}(x_i(t) - x_j(t)) = 0. \tag{8.1}$$

The necessary and sufficient condition for consensus to occur is that the communication graph among the agents is connected. Based on this, we assume that the communication network between the batteries is connected, meaning that they can share information among themselves.

To model the dynamics of each of the batteries, one can simply use a single integrator in the discrete-time domain, as in (8.2):

$$x_i(k+1) = x_i(k) + T_s u_i(k) \tag{8.2}$$

Here, T_s is the sampling time, and u_i is the control input. To achieve consensus among the agents, the control input can be chosen as follows:

$$u_i(k) = g \sum_{j \in \mathcal{N}_i} a_{ij}(x_j(k) - x_i(k)) \tag{8.3}$$

where a_{ij} are the elements of the adjacency matrix A, and $g \in \mathbb{R}$ is a positive control gain that must satisfy the following condition:

$$0 < g \leq \frac{1}{D_{\max}} \tag{8.4}$$

to lead to consensus [21]. The quantity $D_{\max} > 0$ denotes the maximum degree of a node in the communication graph.

8.1.2 Dynamics and Operating Modes

The state of charge of a battery in an islanded grid can be defined based on the current that charges or discharges it, the time-step, the capacity of the battery, and the initial state of charge, which is known as the Coulomb Counting Method (CCM) [73]. For a discrete-time dynamical model, one can assume (8.5) as the evolution dynamics of the SoC:

$$SoC_i(k+1) = SoC_i(k) - \frac{\Delta T \cdot I_i(k)}{C_i \times 3600}, \quad i = 1, \ldots, N, \tag{8.5}$$

where N is the number of batteries in the islanded grid, ΔT is the time interval, and SoC_i, C_i, and I_i are the state of charge, capacity (Ampere-hours, Ah), and current of battery i, respectively. Based on this formulation, if I_i is positive, the battery discharges and adds current to the system, meaning that the SoC of the battery decreases. In contrast, a negative I_i means that the battery is charging, which leads to an increase in SoC. As mentioned before, the batteries share current to reach consensus in SoC, based on their capacities.

Four different operating modes are possible based on the amount of generated energy and consumption.

Mode 1: When the demanded power is equal to the power generated by the RES units, there is neither surplus energy to charge the batteries nor a deficit that requires battery support. Therefore, the BESSs remain out of service in this operating condition.

Mode 2: When the demanded power exceeds the total power generated by the MPPT-based RES units, the batteries must compensate for the deficit to meet the load demand. In this case, the batteries discharge and contribute to energy provision according to their capacities and SoC levels.

Mode 3: When the demanded power is lower than the power generated by the MPPT-based RES units, excess energy is produced. If some batteries are not fully charged, they are activated in the charging mode to act as additional loads and absorb this surplus energy. Not to mention that batteries will be charged in proportion to their capacities and SoCs.

Mode 4: Once the BESSs are fully charged, they can no longer operate as additional loads. If generation still exceeds the demand under this condition, the voltage of the common bus begins to rise. To prevent this, the RESs exit the MPPT mode and switch to load-sharing control, thereby reducing their power generation and regulating the common bus voltage.

Note that in Mode 4, the currents of the RES units will be decreased to meet the

exact demanded energy; therefore, they are no longer in MPPT mode. The division of the currents is based on the maximum currents that each of the RES units can supply and the demanded current. Thus, the provided current by each unit is given in (8.9).

Aside from Mode 1, which is the ideal mode, the remaining working modes can happen in two different cases, as follows.

Case 1. The SoCs are the same for the batteries.

For this case, there is no need to run a consensus algorithm for the batteries, and the charging/discharging will be based on their capacities. Of course, this process should be carried out in a way that keeps the SoCs the same for the remainder of the operation time. Based on Eq. (8.5), we have:

$$Soc_i(k) - \frac{\Delta T I_i(k)}{C_i \times 3600} = Soc_j(k) - \frac{\Delta T I_j(k)}{C_j \times 3600} \tag{8.6}$$

Since SoCs are equal, one can write Eq. (8.6) as follows:

$$\frac{I_i(k)}{C_i} = \frac{I_j(k)}{C_j} \tag{8.7}$$

Considering and extending Eq. (8.7), one can write the following formula for the reference currents when the SoCs are the same:

$$I_i(k) = \frac{I_{error}(k) C_i}{\sum_{j=1}^{N} C_j}, \tag{8.8}$$

where $I_{\text{error}} = I_{\text{demand}} - \sum_{i=1}^{M} I_{\text{RES}_i}$ (M is the number of RES units). Using this simple formula for the currents of the batteries, the SoCs will remain the same for all the batteries. Note that I_{error} can be negative or positive, which causes the batteries

to charge or discharge, respectively.

$$I_{RES_i}(k) = \frac{I_{\text{demand}}(k) I_{RES_{max_i}}}{\sum_{j=1}^{M} I_{RES_{max_j}}}, \tag{8.9}$$

Here, I_{demand} is the required current that must be provided by the sources (i.e., RESs or both RESs and BESSs, depending on the microgrid's working mode) to preserve the common bus voltage at its nominal value. This value can be determined using the DC bus signaling technique [53, 54], which is described in the following subsection.

Case 2. The SoCs are not the same for the batteries.

In this case, the consensus algorithm must be run for the batteries to reach the same values of SoC, as well as to discharge or charge the batteries with a suitable proportion of the currents to respect the capacities of the batteries. Considering the consensus formula for the case of SoCs (8.10):

$$Soc_i(k+1) = Soc_i(k) + g \sum_{j \in N_i} a_{ij}(Soc_j(k) - Soc_i(k)) \tag{8.10}$$

Based on Eq. (8.5) and Eq. (8.10), one can compute the values of the currents that discharge/charge the batteries to reach consensus on the SoCs. In Eq. (8.10), the control input is the difference between SoCs, which needs to be translated into the required current from the battery. This can be done using Eq. (8.5) and Eq. (8.10) as follows.

$$\begin{aligned} g \sum_{j \in N_i} a_{ij}\big(Soc_j(k) - Soc_i(k)\big) &= -\frac{\Delta T \cdot I_{Soc_i}(k)}{C_i \times 3600} \\ I_{Soc_i}(k) &= \Big(-\frac{C_i \times 3600}{\Delta T}\Big)\Big(g \sum_{j \in N_i} a_{ij}\big(Soc_j(k) - Soc_i(k)\big)\Big) \end{aligned} \tag{8.11}$$

where I_{SoC_i} is the current needed to compensate for the difference in the SoC. I_{SoC_i}

and I_{error} will be used to compute the reference current of each battery, I_i. Note that I_{SoC_i} can be a large current because of the nature of the relationship between the current and the SoC of a battery. Let us define $\Delta_i = g\sum_{j\in\mathcal{N}_i} a_{ij}\big(SoC_j(k) - SoC_i(k)\big)$. From (8.11), one can notice the large value of $C_i \cdot 3600$, which is natural since the SoC of a battery changes slowly while charging or discharging. Therefore, what is suggested in this thesis is to compute the reference current for each battery as in Eq. (8.12).

$$I_i(k) = \begin{cases} \dfrac{I_{Soc_i}(k)}{\sum_{j=1}^{N} I_{Soc_j}(k)} I_{error} & I_{error}\cdot\Delta_i < 0, \\ 0 & else. \end{cases} \tag{8.12}$$

In Eq. (8.12), the first condition refers to the fact that the signs of I_{error} and Δ_i must be different. For instance, when I_{error} is positive (negative), it means that the batteries should send (receive) current; however, only those batteries should participate that have a negative (positive) Δ_i, which means that they have a greater (smaller) SoC with respect to the other batteries in the grid.

Remark 1. Considering Eq. (8.8) and Eq. (8.12), one can notice that, to compute the reference current for each battery, the information of all the batteries, the RES units, and I_{demand} is needed. Therefore, a fully connected network of the elements in this islanded grid is required.

Remark 2. Using the normal consensus formula, the batteries would send energy to each other to make the consensus happen; however, since this consensus applies to a practical problem, the charging of the batteries is desired to be carried out by the RES units, not by the batteries themselves. Therefore, Eq. (8.12) has been used to compute the needed current for the batteries to reach a consensus on SoCs while ensuring that the batteries do not charge each other.

Remark 3. Due to high demand (generation), it is possible that the batteries reach $SoC = 0$ ($SoC = 1$). In this case, the batteries stop discharging (charging) since

Table 8.1: Characteristics of the tested DC-MGs.

Parameters	**Symbol**	**Value**
Nominal Voltage	V_{nom}	48 V
Max. Current (PV_1 & PV_2)	I_{max}	10 A
Max. Current ($Wind\ Turbine$)	I_{max}	5 A
Common Bus Capacitance	C_{bus}	20 mF

they are empty (full). Excessive consumption may cause the batteries to become fully depleted (i.e., $SoC = 0$), which is a fundamental challenge in islanded grid design and threatens continuous electrification. In such a situation, load shedding becomes the only unavoidable option. On the other side, during excessive generation, the case with $SoC = 1$ of all batteries leads to changing the working mode of the RES units from MPPT phase to load-sharing phase, as mentioned in Mode 4.

8.1.3 DC Bus signaling

The DC common-bus voltage (i.e., V_{bus}) reflects the power balance condition of the microgrid. This is because a generation deficit discharges the equivalent DC-bus capacitor, causing a voltage drop, whereas surplus generation charges it, resulting in a voltage rise. This correlation between the common-bus equivalent capacitance and the voltage deviation provides an appropriate criterion for determining the proper value of I_{error}, the dominant parameter in the proposed control approach that specifies the surplus or deficit current to be compensated by the batteries. Irrespective of the operating mode of the microgrid, the common-bus voltage can be represented by the following general formula:

$$V_{bus} = \frac{1}{C_{bus}} \int_{t_0}^{t} \left(I_{\text{demand}} - \sum_{i=1}^{M} I_{RES_i} \right) dt \tag{8.13}$$

where C_{bus} is the common bus capacitance. Assuming an ideal controller, an ideal battery (with negligible terminal-voltage variation across different SoC levels), and

considering the demonstration in [74] that line resistances can be designed to ensure negligible voltage drops, the dynamic behavior of V_{bus} can be obtained by solving the previous differential equation, given by:

$$V_{\text{bus}}(t) = V_{\text{bus}}(t_0) + \frac{I_{\text{demand}} - \Sigma_{i=1}^{M} I_{RES_i}}{C_{\text{bus}}} (t - t_0), \tag{8.14}$$

Thus, V_{bus} changes linearly with time, with the slope of $(I_{\text{demand}} - \Sigma_{i=1}^{M} I_{RES_i})/C_{\text{bus}}$. Under normal conditions, $V_{\text{bus}}(t_0) = V_{nom}$. In steady-states (i.e., $t : t \to \infty$), the voltage deviation from its nominal value (i.e., $\Delta V = \lim_{t\to\infty} V_{bus}(t) - V_{nom}$) is zero provided that the supply satisfies the demand, denoted as $I_{error} = 0$. Otherwise, I_{error} can be computed from the resulting voltage deviation at the common bus as well as the sum of measured terminal currents of i-th RES (I_{RES_i}), as shown in Eq. (8.14).

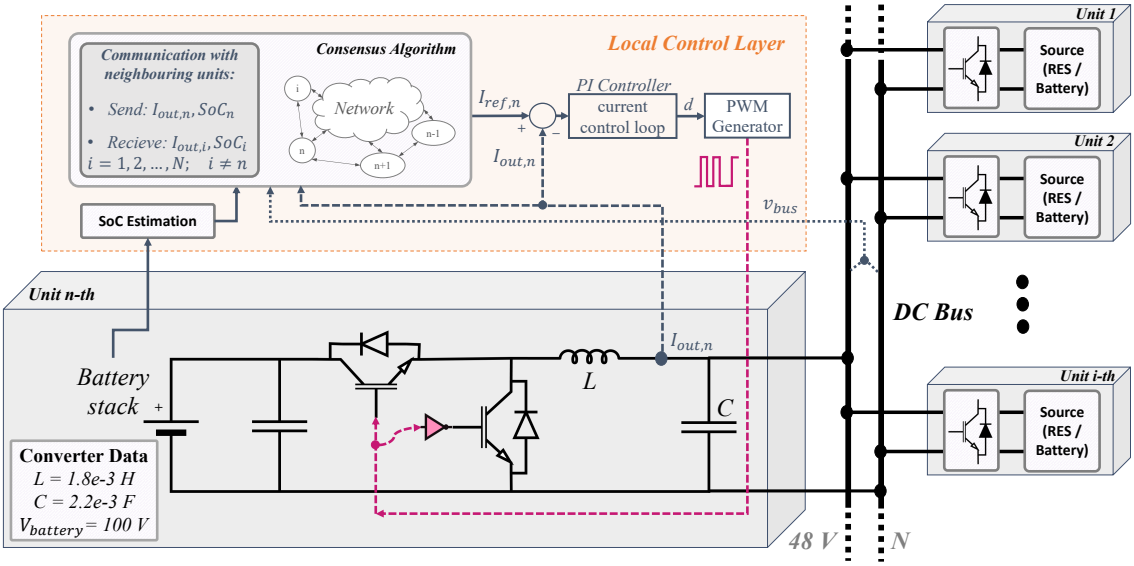


Figure 8.1.1: Proposed converter and control diagram for batteries.

8.1.4 Converter and Controller Diagram

As shown in Fig. 8.1.1, the local controller layer receives the information regarding SoCs, currents, and bus voltage to compute the reference current for the SoC consensus among the storage units. Using a PI controller, the error between the battery

and reference current is used to provide the needed duty cycle. This duty cycle will then be applied to a Pulse Width Modulation (PWM) generator to control the battery current by manipulating the corresponding converter, and eventually, leading to voltage regulation and SoC consensus among the batteries. The converter transfer function based on this elaboration can be determined as follows:

$$G_{Converter} = \frac{\frac{I_{Battery}}{DutyCycle}}{\frac{1 + r_C C s}{LCs^2 + (\frac{L}{R} + (r_L + r_C)C)s + (1 + \frac{r_L}{R})}} \tag{8.15}$$

where C and L represent the capacitance and inductance of the converter output filter, respectively, and R is the total load connected to the bus. The parameters r_C and r_L stand for Equivalent Series Resistance (ESR) of the filter capacitor and the inductor, respectively which are included to model the converter more realistically [75].

8.1.5 Simulation Results for Consensus of SoCs

In this section, the simulation results for the consensus among the SoCs, charge/discharge currents of the batteries, demanded current, and produced current are illustrated. Three different examples are provided to show different operation modes of the islanded grid.

8.1.5.1 Example 1

In this example, Modes 1, 2, and 3 have been simulated. For the initial parameters, we considered $N = 5$ batteries with initial SoCs of $SoC = [0.9, 0.5, 0.7, 0.8, 0.6]$, zero initial currents from the batteries, capacities as $Cap = [50, 100, 150, 200, 250]Ah$, $g = 0.2$, and $\Delta T = 1\,\text{s}$.

The simulation runtime is 86400 seconds (a full day) to effectively show the results. The demanded energy from the grid and the produced energy from PV units and

the wind turbine are time-varying to show the nature of the problem of balancing energy among producers and users. Due to changes in sunlight and wind, RESs do not produce constant energy all the time, and consumers also do not use the same amount of energy during a day.

Fig. 8.1.2 shows I_{demand}, I_{RES}, and the difference between them, which is I_{error}. This represents a logical operating mode for a full day in which I_{demand} and I_{RES} vary due to the available sunlight, wind, and consumption of the users.

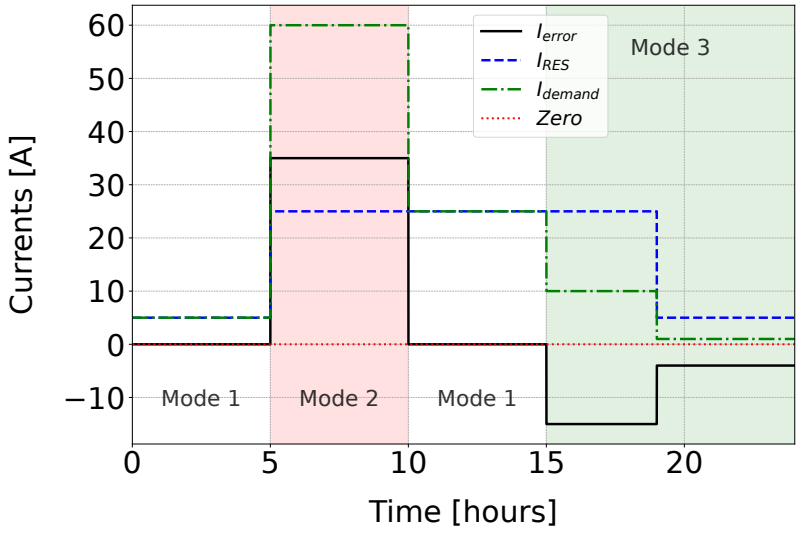


Figure 8.1.2: Demanded and produced currents in a day.

Fig. 8.1.3 shows the consensus of the SoCs for the batteries. Starting from the initial SoCs, they reach consensus and then continue to discharge/charge based on the demanded and produced energy. As you can notice, battery 2 does not participate in the consensus until 8 hours due to the positive I_{error} and its negative Δ_i as in Eq. (8.12). Therefore, as an example, battery 2 is charged not by other batteries but by the produced energy from the RES units.

Fig. 8.1.4 shows the coordinated charge/discharge currents for the $N = 5$ batteries based on the demanded and produced energy. From Figs. 8.1.3 and 8.1.4, it can be seen that at 08:00, consensus among the batteries is achieved. After this time, the batteries discharge based on their capacities, since this ensures that their consensus will not be violated, and consequently, the full contribution of all the batteries

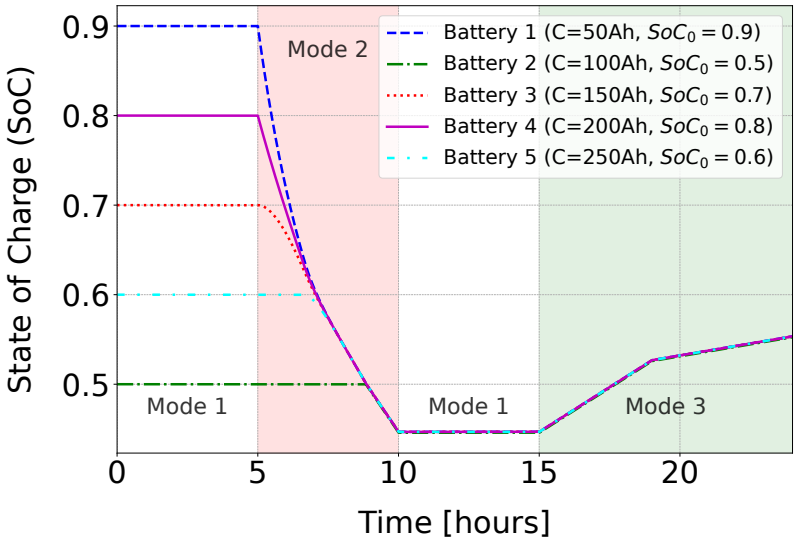


Figure 8.1.3: Consensus of the SoCs for $N = 5$ batteries.

prolongs their lifespan.

8.1.5.2 Example 2

Operation Mode 4 is shown in this example. Consider that the amount of produced energy is far more than the demanded one; thus, the SoCs of the batteries become 1, and they cannot absorb the excess energy. In this case, as mentioned in Mode 4, the RES units reduce their production in order to meet exactly the demanded energy. Initial conditions are the same except for $Cap = [50, 100, 150, 200, 250]Ah$. Fig. 8.1.5 shows the SoCs of the batteries, in which all of them have reached 1 while they are in Mode 3.

Fig. 8.1.6 shows the currents of the batteries in Modes 3 and 4. In Mode 3, they absorb the excess energy, and as soon as their SoC reaches one, the RES units change their mode to 4. In this mode, the batteries do not absorb any current since they are full.

The produced energy in Mode 4 must exactly meet the demanded energy, which can be seen in Fig. 8.1.7. With a changing I_{demand}, the sum of the RES units follows the demanded energy.

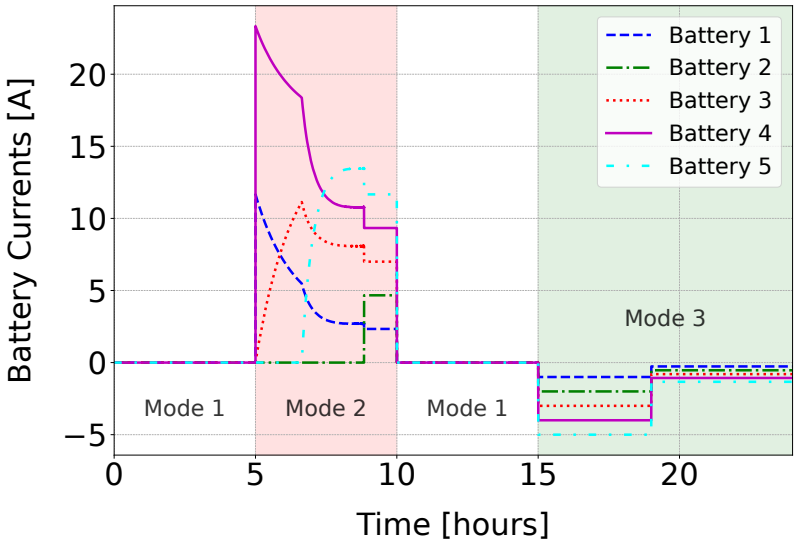


Figure 8.1.4: Battery currents for $N = 5$ batteries.

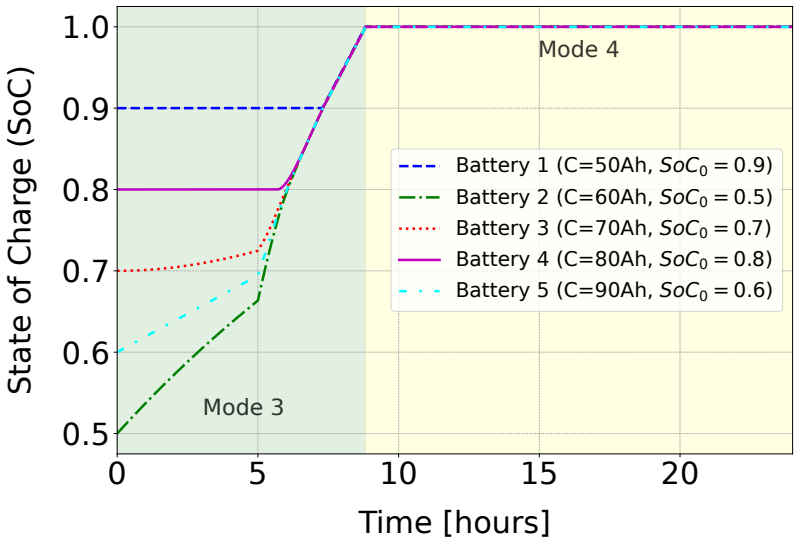


Figure 8.1.5: SoC consensus in Mode 4 for $N = 5$ batteries.

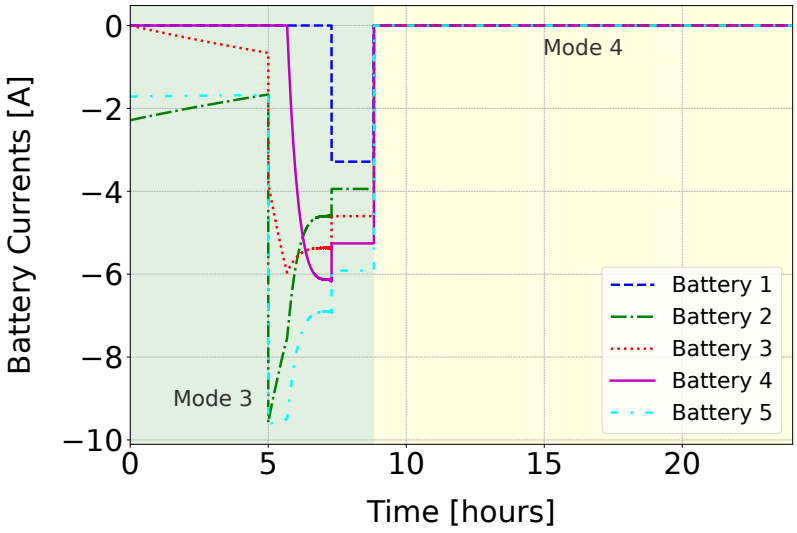


Figure 8.1.6: Battery currents in Mode 4 for $N = 5$ batteries.

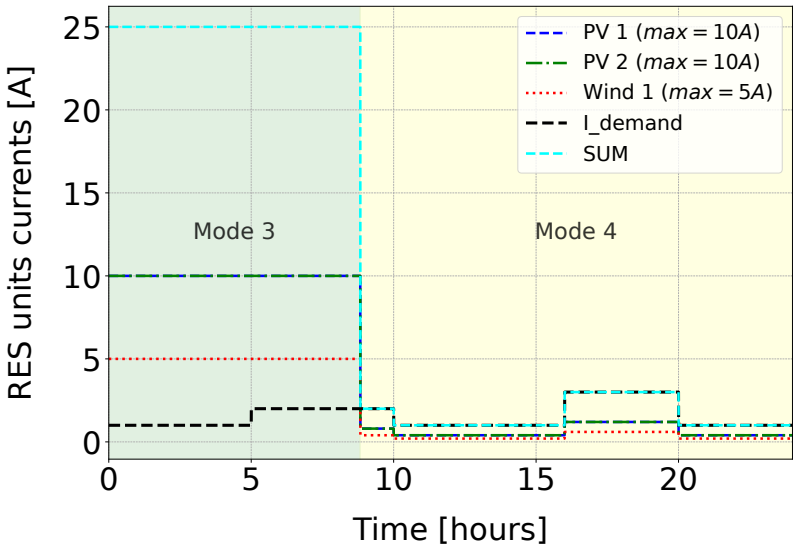


Figure 8.1.7: Demanded and RES units' currents in Mode 4 for $M = 3$ RES units.

8.1.5.3 Example 3

This example illustrates the implementation of the proposed algorithm applied to a real converter transfer function. This model incorporates the details of a real converter model where its input is the duty cycle and its output is the battery current. Note that for this simulation, we considered sampling time $T_s = 0.001s$, total duration of $T = 3s$, initial $SoC = [0.5, 0.8, 0.3]$, battery capacity of $Cap =$

$[0.01, 0.02, 0.05]$, and $N = 3$ batteries. The reason to choose small capacities is due to avoid long simulation time to show the convergence of SoCs. As mentioned before, a PI controller is being used to compensate the current error which its parameters have been chosen as $k_P = 0.001$, and $k_I = 40$ which leads to a perfect tracking of the reference current with a transition time equal to 5 samples ($0.005s$). Fig. 8.1.8 demonstrates the currents that the batteries shared as demanded by the converters. As it is obvious from the fig, the tracking of the current is almost perfect which shows that the algorithm is working in real-time with a real sampling time ($T_s = 0.001$). Two different working modes have been considered that has been mentioned previously as Mode 2 and Mode 3.

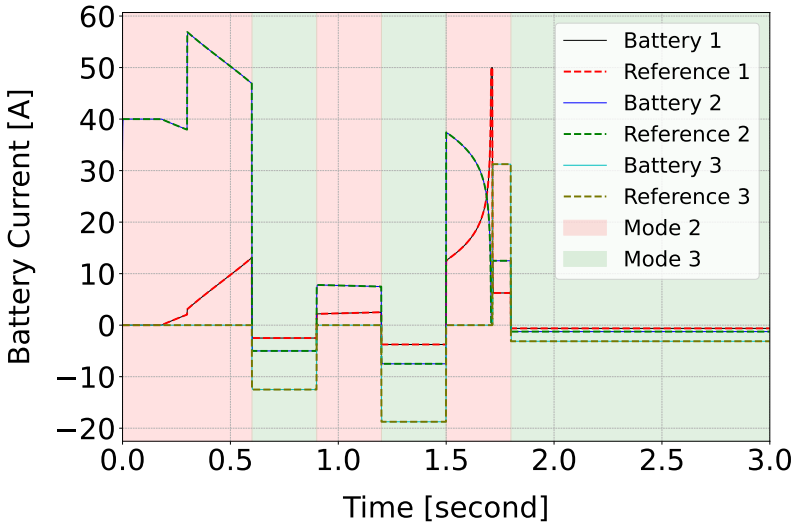


Figure 8.1.8: The battery currents provided by the converters.

Fig. 8.1.9 demonstrates the convergence of the SoCs for the batteries while working in two different modes. As can be seen, circular current is avoided and the charging and discharging of the batteries respects the SoCs and the capacities of the batteries.

Based on the Fig 8.1.8 and Fig 8.1.9, the algorithm shows a perfect tracking of the reference current in a short time while respecting the real-time constraints. Note that this simulation is only for 3 seconds to avoid long simulations and simplicity.

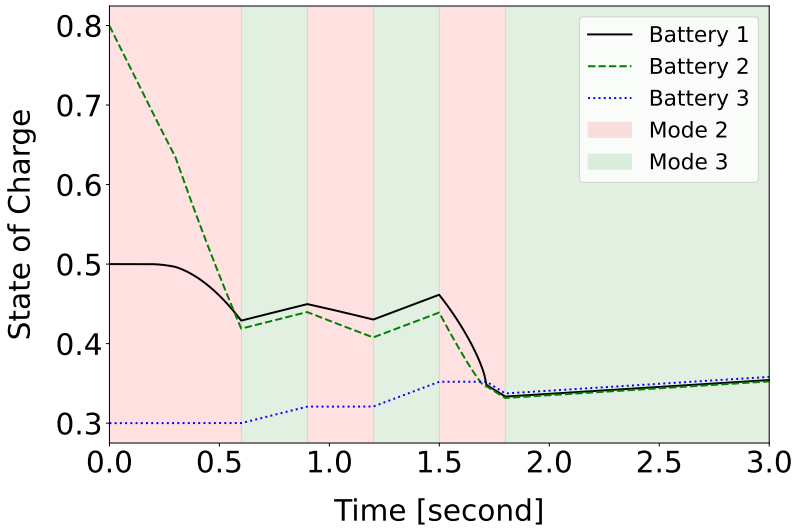


Figure 8.1.9: The *SoC* convergence for the batteries working in Mode 2 and Mode 3.

Chapter 9

Conclusions and Future Work

9.1 Conclusions

This thesis addresses two key problems within the domain of multi-agent systems and distributed control: the coordinated motion control of drone swarms for efficient area coverage and video acquisition, and the consensus-based coordination of heterogeneous backup batteries in DC islanded grids. In both cases, the focus was on developing control strategies that are scalable, adaptive, and capable of operating under practical constraints.

In the first part of the thesis, a leader–follower framework based on MPC was developed for the coordination of drone swarms. The proposed approach enabled the agents to maintain a distance-based formation while following a predefined trajectory, ensuring collision avoidance and the capture of overlapping video streams. To enhance the performance of the system, an online path planning strategy was introduced, allowing the leader agent to dynamically identify uncovered areas and guide the swarm accordingly. This resulted in improved area coverage while adapting to the available network bandwidth, which directly influenced the achievable video quality. Also, the online path planner works for any convex area, which enhances the flexibility of the proposed method in dealing with various target areas. Later, a

comprehensive sensitivity analysis was performed to identify the effectiveness of each of the weighting factors participating in the optimization of the MPC controller. Due to the limitations of the available network bandwidth and to mitigate the scalability of the problem, an event-triggered communication framework has been added to the problem avoiding unnecessary communications, leading network bandwidth overload. Within this approach, a more complicated drone model has been investigated to have a more realistic scenario for the swarm control of the drones.

Furthermore, a grid-based coverage method was proposed as an alternative strategy to improve coverage efficiency. By positioning the leader at the center of the formation and organizing the follower agents into a polygonal structure, the swarm was able to sweep the environment more effectively compared to the other previously mentioned methods. A prescribed nonlinear controller was designed to ensure accurate tracking of the reference positions while respecting the velocity constraints, and a mathematical convergence proof was provided to support the stability and correctness of the proposed control law.

In the second part of the thesis, the consensus problem for backup batteries in DC islanded grids was investigated. A consensus-based control strategy was developed to achieve balanced power sharing among heterogeneous batteries with different initial states of charge and capacities. Unlike simplified models often used in the literature, a realistic converter transfer function was incorporated to capture the relationship between the duty cycle and the battery current. The results demonstrated that the proposed approach is capable of ensuring coordinated operation and fair load distribution under realistic system dynamics.

Overall, the contributions of this thesis demonstrate the effectiveness of combining MPC and consensus-based strategies for solving complex coordination problems in multi-agent systems. The proposed methods provide practical solutions for improving performance in both autonomous aerial systems and smart energy networks, while maintaining scalability and robustness.

9.2 Future Work

Despite the promising results obtained in this thesis, several directions remain open for further investigation.

Experimental validation of the proposed framework for the swarm control represents an important next step to better understand its practical limitations and potential challenges. Starting with a small number of drones, the communication protocols and control strategies should be tested to evaluate their performance and robustness in real-world scenarios.

In the current work, an online path planning method and an improvement to it have been proposed to enhance area coverage, where one is limited to rectangular environments and the improved version is applicable to convex regions. A natural extension of this work is the development of a more general path planning algorithm capable of handling obstacles and non-convex environments.

In the grid-based coverage approach, the initial conditions of the agents must satisfy specific requirements imposed by the method. While this assumption is common in the literature, enhancing the robustness of the algorithm by relaxing these constraints is of interest. In particular, the implementation of a rendezvous strategy based on an optimal transport formulation could enable an energy-efficient reconfiguration of the agents toward the desired initial formation.

The proposed consensus-based control strategy for backup batteries in DC islanded grids can be further validated through hardware-in-the-loop experiments to assess its performance under realistic operating conditions. In addition, incorporating voltage regulation into the control framework would provide a more comprehensive and practical solution, and is therefore identified as an important direction for future work.

These directions can further improve the robustness, scalability, and real-world

applicability of the proposed methods and open the path for future research in distributed control of complex systems.

Bibliography

[1] W. B. Carlson, "A history of control engineering, 1800–1930 by stuart bennett," *Technology and Culture*, vol. 23, no. 4, pp. 657–658, 1982.

[2] S. Bennett, *A history of control engineering, 1930-1955*. No. 47, IET, 1993.

[3] M. Moradi, M. Najafi, Z. Lin, and H. Pan, "Advancing sensor network optimization and data analytics for smart wastewater and stormwater collection systems: An overview," *Pipelines 2025*, pp. 463–472, 2025.

[4] R. Sun, "Cognitive science meets multi-agent systems: A prolegomenon," *Philosophical psychology*, vol. 14, no. 1, pp. 5–28, 2001.

[5] M. A. Rezaei, P. Bagheri, and F. Hashemzadeh, "Consensus tracking of multi-agent systems in the presence of byzantine agents using model predictive control," in *2022 8th International Conference on Control, Instrumentation and Automation (ICCIA)*, pp. 1–5, 2022.

[6] F. Mehdifar, C. P. Bechlioulis, F. Hashemzadeh, and M. Baradarannia, "Prescribed performance distance-based formation control of multi-agent systems," *Automatica*, vol. 119, p. 109086, 2020.

[7] X. Li and Y. Xi, "Flocking of multi-agent dynamic systems with guaranteed group connectivity," *Journal of Systems Science and Complexity*, vol. 21, no. 3, pp. 337–346, 2008.

[8] S. Shoja, M. Baradarannia, F. Hashemzadeh, M. Badamchizadeh, and P. Bagheri, "Surrounding control of nonlinear multi-agent systems with non-identical agents," *ISA transactions*, vol. 70, pp. 219–227, 2017.

[9] A. T. Koru, A. Ramírez, S. B. Sarsılmaz, R. Sipahi, T. Yucelen, and K. M. Dogan, "Containment control of multi-human multi-agent systems under time delays," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 54, no. 6, pp. 3344–3356, 2024.

[10] M. Shaw, B. Harrow, and S. Herman, "Distributed artificial intelligence for multi-agent problem solving and group learning," in *Proceedings of the Twenty-Fourth Annual Hawaii International Conference on System Sciences*, vol. 4, pp. 13–26, IEEE, 1991.

[11] F. H. Panahi, F. H. Panahi, and T. Ohtsuki, "An intelligent path planning mechanism for firefighting in wireless sensor and actor networks," *IEEE Internet of Things Journal*, vol. 10, no. 11, pp. 9646–9661, 2023.

[12] M. Orisatoki, W. Sheng, E. Pinar, A. Rasoulzadeh, M. Amouzadi, and A. M Dizqah, "Informative path planning for autonomous mapping of unknown non-convex environments: design, benchmarking, and validation," 2026.

[13] Y. Wu, S. Wu, and X. Hu, "Cooperative path planning of uavs & ugvs for a persistent surveillance task in urban environments," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4906–4919, 2020.

[14] G. Lau and H. H. Liu, "Real-time path planning algorithm for autonomous border patrol: design, simulation, and experimentation," *Journal of intelligent & robotic systems*, vol. 75, no. 3, pp. 517–539, 2014.

[15] J. Hu and G. Feng, "Distributed tracking control of leader–follower multi-agent systems under noisy measurement," *Automatica*, vol. 46, no. 8, pp. 1382–1387, 2010.

[16] X. Li and J. Wang, "Fully distributed fault-tolerant leaderless consensus of multi-agent systems under directed communication graph," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 2, pp. 1049–1059, 2022.

[17] F. Gul, A. Mir, I. Mir, S. Mir, T. U. Islaam, L. Abualigah, and A. Forestiero, "A centralized strategy for multi-agent exploration," *IEEE Access*, vol. 10, pp. 126871–126884, 2022.

[18] W. Liu, W. Gu, W. Sheng, X. Meng, Z. Wu, and W. Chen, "Decentralized multi-agent system-based cooperative frequency control for autonomous microgrids with communication constraints," *IEEE Transactions on Sustainable Energy*, vol. 5, no. 2, pp. 446–456, 2014.

[19] F. Zarei and B. Shafai, "Consensus of multi-agent singular systems by using an algebraic transformation," in *2024 32nd Mediterranean conference on control and automation (MED)*, pp. 682–687, IEEE, 2024.

[20] H. Kim, H. Shim, and J. H. Seo, "Output consensus of heterogeneous uncertain linear multi-agent systems," *IEEE Transactions on Automatic Control*, vol. 56, no. 1, pp. 200–206, 2010.

[21] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[22] J. J. Roldán-Gómez, E. González-Gironda, and A. Barrientos, "A survey on robotic technologies for forest firefighting: Applying drone swarms to improve firefighters' efficiency and safety," *Applied sciences*, vol. 11, no. 1, p. 363, 2021.

[23] L. Tang and G. Shao, "Drone remote sensing for forestry research and practices," *Journal of forestry research*, vol. 26, no. 4, pp. 791–797, 2015.

[24] A. Albanese, V. Sciancalepore, and X. Costa-Pérez, "Sardo: An automated search-and-rescue drone-based solution for victims localization," *IEEE Transactions on Mobile Computing*, vol. 21, no. 9, pp. 3312–3325, 2021.

[25] B. Mishra, D. Garg, P. Narang, and V. Mishra, "Drone-surveillance for search and rescue in natural disaster," *Computer Communications*, vol. 156, pp. 1–10, 2020.

[26] W. Budiharto, A. A. Gunawan, J. S. Suroso, A. Chowanda, A. Patrik, and G. Utama, "Fast object detection for quadcopter drone using deep learning," in *2018 3rd international conference on computer and communication systems (ICCCS)*, pp. 192–195, IEEE, 2018.

[27] M.-Q. Dao, J. S. Berrio, V. Frémont, M. Shan, E. Héry, and S. Worrall, "Practical collaborative perception: A framework for asynchronous and multi-agent 3d object detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 9, pp. 12163–12175, 2024.

[28] W. Brescia, P. Gomes, L. Toni, S. Mascolo, and L. De Cicco, "Millinoise: a millimeter-wave radar sparse point cloud dataset in indoor scenarios," in *Proceedings of the 15th ACM Multimedia Systems Conference*, pp. 422–428, 2024.

[29] N. Barone, W. Brescia, S. Mascolo, and L. De Cicco, "Apeiron: a multimodal drone dataset bridging perception and network data in outdoor environments," in *Proceedings of the 15th ACM Multimedia Systems Conference*, pp. 401–407, 2024.

[30] U. Dah-Achinanon, S. E. Marjani Bajestani, P.-Y. Lajoie, and G. Beltrame, "Search and rescue with sparsely connected swarms," *Autonomous Robots*, vol. 47, no. 7, pp. 849–863, 2023.

[31] M. A. Rezaei, P. Bagheri, and F. Hashemzadeh, "Predictive consensus tracking of multi-agent systems in the presence of byzantine agents and connection loss

of reference signals," *Optimal Control Applications and Methods*, vol. 45, no. 2, pp. 842–854, 2024.

[32] B. D. Anderson, B. Fidan, C. Yu, and D. Walle, "Uav formation control: Theory and application," in *Recent advances in learning and control*, pp. 15–33, Springer, 2008.

[33] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Transactions on automatic control*, vol. 51, no. 3, pp. 401–420, 2006.

[34] A. Salih, M. Moghavvemi, and H. A. Mohamed, "Flight pid controller design for a uav quadrotor.," *Scientific Research and Essays (SRE)*, vol. 5, no. 23, pp. 3660–3667, 2010.

[35] G. Muñoz, C. Barrado, E. Çetin, and E. Salami, "Deep reinforcement learning for drone delivery," *Drones*, vol. 3, no. 3, p. 72, 2019.

[36] M. A. S. Teixeira, F. Neves-Jr, A. Koubâa, L. V. R. De Arruda, and A. S. De Oliveira, "A quadral-fuzzy control approach to flight formation by a fleet of unmanned aerial vehicles," *IEEE access*, vol. 8, pp. 64366–64381, 2020.

[37] H. Liu, S. Suzuki, W. Wang, H. Liu, and Q. Wang, "Robust control strategy for quadrotor drone using reference model-based deep deterministic policy gradient," *Drones*, vol. 6, no. 9, p. 251, 2022.

[38] F. Gazzola and E. M. Marchini, "A minimal time optimal control for a drone landing problem," *ESAIM: Control, Optimisation and Calculus of Variations*, vol. 27, p. 99, 2021.

[39] I. Medeiros, A. Boukerche, and E. Cerqueira, "Swarm-based and energy-aware unmanned aerial vehicle system for video delivery of mobile objects," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 1, pp. 766–779, 2021.

[40] H. Munawar, A. Hammad, and S. Waller, "Disaster region coverage using drones: Maximum area coverage and minimum resource utilisation. drones 2022, 6, 96," 2022.

[41] A. Bist and C. Singhal, "Efficient immersive surveillance of inaccessible regions using uav network," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 1–6, IEEE, 2021.

[42] U. Choi and S. Lee, "Bandwidth-aware coverage path planning for swarm of uavs with aerial base station," in *2023 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 360–365, IEEE, 2023.

[43] H. Huang, A. V. Savkin, and C. Huang, "Decentralized autonomous navigation of a uav network for road traffic monitoring," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 57, no. 4, pp. 2558–2564, 2021.

[44] R. Zahinos, H. Abaunza, J. Murillo, M. Trujillo, and A. Viguria, "Cooperative multi-uav system for surveillance and search&rescue operations over a mobile 5g node," in *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1016–1024, IEEE, 2022.

[45] M. Schwager, B. J. Julian, M. Angermann, and D. Rus, "Eyes in the sky: Decentralized control for the deployment of robotic camera networks," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1541–1561, 2011.

[46] Y. Chen, H. Zhang, and Y. Hu, "Optimal power and bandwidth allocation for multiuser video streaming in uav relay networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 6644–6655, 2020.

[47] S. Colonnese, A. Carlesimo, L. Brigato, and F. Cuomo, "Qoe-aware uav flight path design for mobile video streaming in hetnet," in *2018 IEEE 10th Sensor Array and Multichannel Signal Processing Workshop (SAM)*, pp. 301–305, IEEE, 2018.

[48] R. Shivgan and Z. Dong, "Energy-efficient drone coverage path planning using genetic algorithm," in *2020 IEEE 21st International Conference on High Performance Switching and Routing (HPSR)*, pp. 1–6, IEEE, 2020.

[49] S. W. Cho, H. J. Park, H. Lee, D. H. Shim, and S.-Y. Kim, "Coverage path planning for multiple unmanned aerial vehicles in maritime search and rescue operations," *Computers & Industrial Engineering*, vol. 161, p. 107612, 2021.

[50] K. Zhou, H. Yang, and F. Blaabjerg, "A review of control strategies for islanded microgrids with renewable energy sources," *Renewable and Sustainable Energy Reviews*, vol. 158, p. 112086, 2022.

[51] M. M. Islam *et al.*, "Improving reliability and stability of the power systems: A comprehensive review on the role of energy storage systems to enhance flexibility," *IEEE Access*, vol. 12, pp. 152738–152765, 2024.

[52] S. Ali, M. Iqbal, and M. Jamil, "Model predictive control of consensus-based energy management for dres and bess," *PLOS ONE*, vol. 18, no. 3, p. e0282224, 2023.

[53] M. R. Nasab *et al.*, "Stand-alone dc microgrids for rural areas: A decentralized energy management and voltage regulation approach," in *2024 IEEE Int. Human. Tech. Conf. (IHTC)*, pp. 1–7, IEEE, 2024.

[54] T. Dragičević, J. M. Guerrero, J. C. Vasquez, and D. Škrlec, "Supervisory control of an adaptive-droop regulated dc microgrid with battery management capability," *IEEE Transactions on power Electronics*, vol. 29, no. 2, pp. 695–706, 2013.

[55] M. R. Nasab, S. Bruno, M. Liserre, and M. La Scala, "Adaptive droop control for stability enhancement in islanded dc microgrids with cpls," in *2025 IEEE Kiel PowerTech*, pp. 1–6, IEEE, 2025.

[56] T. Dragičević, J. M. Guerrero, J. C. Vasquez, and D. Škrlec, "Supervisory control of an adaptive-droop regulated dc microgrid with battery management capability," *IEEE Transactions on power Electronics*, vol. 29, no. 2, pp. 695–706, 2013.

[57] E. M. Attia, H. A. Abdelsalam, and E. E. M. Rashad, "Energy management and soc balancing of distributed batteries in ac microgrids using consensus tracking control," *Sustainable Energy, Grids and Networks*, vol. 38, p. 101345, 2024.

[58] M. Cucuzzella, S. Trip, C. De Persis, X. Cheng, A. Ferrara, and A. van der Schaft, "A robust consensus algorithm for current sharing and voltage regulation in dc microgrids," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 4, pp. 1583–1595, 2018.

[59] J. Li, H. Zhang, and S. Wang, "A cooperative control strategy for balancing soc and power in distributed microgrids," *IET Generation, Transmission & Distribution*, vol. 18, no. 2, pp. 210–222, 2024.

[60] C. Li, E. A. A. Coelho, T. Dragicevic, J. M. Guerrero, and J. C. Vasquez, "Multiagent-based distributed state of charge balancing control for distributed energy storage units in ac microgrids," *IEEE Transactions on Industry Applications*, vol. 53, no. 3, pp. 2369–2381, 2016.

[61] L. Wei, Z. Zhong, C. Lang, and Z. Yi, "A survey on image and video stitching," *Virtual Reality & Intelligent Hardware*, vol. 1, no. 1, pp. 55–83, 2019.

[62] X. Meng, W. Wang, and B. Leong, "Skystitch: A cooperative multi-uav-based real-time video surveillance system with stitching," in *Proceedings of the 23rd ACM international conference on Multimedia*, pp. 261–270, 2015.

[63] G. Carlucci, L. De Cicco, S. Holmer, and S. Mascolo, "Congestion control for web real-time communication," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 2629–2642, 2017.

[64] J. Frey, K. Kovach, S. Stemmler, and B. Koch, "Uav photogrammetry of forests as a vulnerable process. a sensitivity analysis for a structure from motion rgb-image pipeline," *Remote Sensing*, vol. 10, no. 6, p. 912, 2018.

[65] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631, 2019.

[66] M. A. Rezaei, G. Manfredi, V. A. Racanelli, L. De Cicco, and S. Mascolo, "Decentralized control of uav swarms for bandwidth-aware video surveillance using nmpc," in *2024 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 947–954, IEEE, 2024.

[67] M. Kamel, T. Stastny, K. Alexis, and R. Siegwart, "Model predictive control for trajectory tracking of unmanned aerial vehicles using robot operating system," in *Robot Operating System (ROS) The Complete Reference (Volume 2)*, pp. 3–39, Springer, 2017.

[68] G. Torrente, E. Kaufmann, P. Föhn, and D. Scaramuzza, "Data-driven mpc for quadrotors," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3769–3776, 2021.

[69] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. v. Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados—a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, vol. 14, no. 1, pp. 147–183, 2022.

[70] F. El Tin, I. Sharf, and M. Nahon, "Guidance of unmanned aerial gliders for wildfire surveillance," in *AIAA Scitech 2020 Forum*, p. 1718, 2020.

[71] Z. Peng, G. Wen, A. Rahmani, and Y. Yu, "Distributed consensus-based formation control for multiple nonholonomic mobile robots with a specified reference

trajectory," *International Journal of Systems Science*, vol. 46, no. 8, pp. 1447–1457, 2015.

[72] I. Palomares, L. Martinez, and F. Herrera, "A consensus model to detect and manage noncooperative behaviors in large-scale group decision making," *IEEE Transactions on Fuzzy Systems*, vol. 22, no. 3, pp. 516–530, 2013.

[73] W.-Y. Chang, "The state of charge estimating methods for battery: A review," *International Scholarly Research Notices*, vol. 2013, no. 1, p. 953792, 2013.

[74] X. Lu, K. Sun, J. M. Guerrero, J. C. Vasquez, and L. Huang, "Double-quadrant state-of-charge-based droop control method for distributed energy storage systems in autonomous dc microgrids," *IEEE transactions on smart grid*, vol. 6, no. 1, pp. 147–157, 2014.

[75] R. W. Erickson and D. Maksimovic, *Fundamentals of power electronics*. Springer Science & Business Media, 2007.