UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

University of Modena and Reggio Emilia

Department of Sciences and Methods for Engineering

Doctoral school of Industrial Innovation Engineering

XXXVIII cycle

Autonomous Control and Cooperation in a Multi-Robot System: Decentralized Strategies under Limited Sensing and Uncertainty

---

Author:
Mattia Catellani

Supervisor:
Lorenzo Sabattini

PhD Coordinator:
Franco Zambonelli

March 2026

Reggio Emilia, Italy, March 2026

Every journey is better when shared.

Acknowledgements

Questo lavoro rappresenta la conclusione di un percorso lungo e impegnativo, che non avrei mai potuto affrontare senza il supporto della mia famiglia. A loro va il mio più sincero ringraziamento, per avermi sostenuto e non avermi mai fatto mancare niente durante tutta la durata dei miei studi.

Il dottorato è stato per me un'esperienza estremamente formativa e stimolante: mi ha dato l'opportunità di viaggiare, di conoscere persone provenienti da tutto il mondo e di lavorare su temi che mi appassionano profondamente. Per questo, desidero ringraziare in modo particolare il mio advisor, Prof. Lorenzo Sabattini, per il supporto costante, la disponibilità e la fiducia dimostratemi nel corso di questi anni.

Tra i ricordi più preziosi di questo percorso ci sono senza dubbio le giornate passate in laboratorio con i miei colleghi del gruppo ARSControl, che con il tempo si sono trasformati in veri e propri amici. Con loro ho condiviso non solo risultati e difficoltà, ma anche momenti di crescita e confronto quotidiano, oltre a tante risate e divertimento. Un ringraziamento speciale va poi a Mattia e Marta, con cui ho avuto il privilegio di lavorare più a stretto contatto. Grazie per le giornate passate all' "aeroporto", per il casinò e le maschere a Singapore e per tutti quei momenti che hanno reso speciale questo ultimo periodo.

Un pensiero particolare va a Veronica, per la presenza costante, il supporto dal momento in cui ho deciso di iniziare questo percorso, la pazienza e la comprensione dimostrati in ogni fase di questo viaggio.

Infine, un caloroso ringraziamento a Kobe, per ricordarmi che, oltre al piacere del viaggio, c'è sempre qualcosa di speciale anche nel tornare a casa.

Abstract

Multi-Robot Systems (MRS) are gaining increasing attention in the research community due to their broad range of potential applications and the numerous advantages they offer over single-robot systems. These advantages include improved scalability, robustness to individual failures, and faster task completion. However, the deployment of MRS in real-world environments is considerably more challenging, as it requires effective coordination and cooperation among multiple agents to achieve common objectives. These challenges become even more demanding under non-ideal conditions, such as limited communication, sensing uncertainties, and dynamic environments, where truly decentralized and reliable solutions are still lacking.

In this thesis, we propose novel control strategies aimed at advancing Multi-Robot Systems toward practical real-world deployment and widespread adoption, with a particular focus on operating under uncertainties and incomplete information. First, we focus on the decentralized control of a team of robots equipped with limited-range sensing capabilities. Building upon traditional coverage control frameworks, we develop strategies for environment exploration and monitoring that enhance the robots' autonomy and efficiency when operating in large-scale environments. Furthermore, we extend the coverage control paradigm to human-robot interaction, enabling robots to adapt their motion based on human intentions and to safely navigate environments shared with humans, also exploring mixed reality solutions to facilitate this interaction.

Taking a step further, we investigate even more challenging operating conditions, focusing on scenarios in which robots are equipped solely with camera-like sensors featuring a limited angular field of view for perceiving their teammates. We first consider strategies that maintain other robots within each agent's visual field to ensure continuous information exchange. Building on this idea, we then propose alternative approaches suited for larger teams, where maintaining visual contact with all other robots is no longer feasible. Among these, we develop an exploration–exploitation strategy that balances neighbor seeking and task execution, as well as a clustering framework that partitions the team into subgroups and coordinates each subgroup as a single agent.

Finally, we explore learning-based solutions, which offer significant advantages by enabling robots to autonomously adapt to complex, uncertain, and dynamic environments without requiring explicit modeling of all system dynamics or interactions. In particular, we investigate decentralized coverage control using both reinforcement learning and neural networks, allowing each robot to learn an effective control policy based solely on locally available information. Furthermore, we employ Gaussian Processes to reconstruct spatial phenomena under limited sensing conditions, enabling the robots to collectively estimate and monitor environmental distributions with improved accuracy. In conclusion, we address the problem of online learning of uncertainties in a robot's motion model, with the goal of extending this capability to multi-robot scenarios such as close-formation control under disturbances (e.g., downwash), cooperative transportation, and other collaborative tasks.

Sommario

I Sistemi Multi-Robot (MRS) stanno attirando un'attenzione crescente nella comunità scientifica grazie alla loro vasta gamma di potenziali applicazioni e ai numerosi vantaggi che offrono rispetto ai sistemi a singolo robot. Tra questi vantaggi si annoverano una maggiore scalabilità, una migliore robustezza ai guasti dei singoli agenti e una più rapida esecuzione dei compiti. Tuttavia, l'impiego dei MRS in ambienti reali risulta notevolmente più complesso, poiché richiede un'efficace coordinazione e cooperazione tra molteplici agenti per raggiungere obiettivi comuni. Queste sfide diventano ancora più impegnative in condizioni non ideali, come comunicazione limitata, incertezze sensoriali e ambienti dinamici, dove soluzioni realmente decentralizzate e affidabili sono ancora carenti.

In questa tesi, proponiamo nuove strategie di controllo volte a favorire l'avanzamento dei Sistemi Multi-Robot verso un impiego pratico nel mondo reale e una diffusione su larga scala, con particolare attenzione al funzionamento in presenza di incertezze e informazioni incomplete. Inizialmente, ci concentriamo sul controllo decentralizzato di un team di robot dotati di sensori a raggio limitato. Basandoci sui tradizionali framework di coverage control, sviluppiamo strategie di esplorazione e monitoraggio dell'ambiente che migliorano l'autonomia e l'efficienza dei robot durante il funzionamento in ambienti di grande scala. Inoltre, estendiamo il paradigma del coverage control all'interazione uomo-robot, permettendo ai robot di adattare il proprio movimento in base alle intenzioni umane e di navigare in sicurezza in ambienti condivisi con le persone, esplorando anche soluzioni di Mixed Reality per facilitare tale interazione.

Successivamente, affrontiamo condizioni operative ancora più complesse, concentrandoci su scenari in cui i robot sono equipaggiati esclusivamente con sensori di tipo visivo, come ad esempio telecamere, con un campo visivo angolare limitato per percepire i propri compagni. In un primo momento, consideriamo strategie che mantengano gli altri robot all'interno del campo visivo di ciascun agente, garantendo così uno scambio continuo di informazioni. Partendo da questa idea, proponiamo poi approcci alternativi adatti a team di dimensioni maggiori, nei quali mantenere il contatto visivo con tutti gli altri

robot non è più fattibile. Tra questi, sviluppiamo una strategia di exploration–exploitation che bilancia la ricerca dei vicini con l'esecuzione della missione, oltre a un framework di clustering che suddivide il team in sottogruppi e coordina ciascun sottogruppo come un singolo agente.

Infine, esploriamo soluzioni basate sull'apprendimento, che offrono vantaggi significativi consentendo ai robot di adattarsi autonomamente ad ambienti complessi, incerti e dinamici senza richiedere una modellazione esplicita di tutte le dinamiche o interazioni del sistema. In particolare, indaghiamo il coverage control decentralizzato utilizzando sia Reinforcement Learning sia reti neurali, permettendo a ciascun robot di apprendere una politica di controllo efficace basata esclusivamente sulle informazioni locali disponibili. Inoltre, impieghiamo Processi Gaussiani per ricostruire fenomeni spaziali in condizioni di utilizzo di sensori con portata limitata, consentendo ai robot di stimare e monitorare collettivamente processi ambientali con maggiore accuratezza. In conclusione, affrontiamo il problema dell'apprendimento online delle incertezze nel modello di movimento di un robot, con l'obiettivo di estendere tale capacità a scenari multi-robot come il controllo in formazione ravvicinata in presenza di disturbi (ad esempio il downwash), il trasporto cooperativo e altri compiti collaborativi.

Contents

List of Figures

List of Publications

Journal papers

- M. Catellani and L. Sabattini, "Distributed control of a limited angular field-of-view multi-robot system in communication-denied scenarios: A probabilistic approach," *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 739–746, 2023
- F. Bertoncelli, V. Radhakrishnan, M. Catellani, G. Loianno, and L. Sabattini, "Directed graph topology preservation in multi-robot systems with limited field of view using control barrier functions," *IEEE Access*, vol. 12, pp. 9682–9690, 2024
- L. Pan, M. Catellani, L. Sabattini, and N. Ayanian, "Robust trajectory generation and control for quadrotor motion planning with field-of-view control barrier certification," *IEEE Robotics and Automation Letters*, vol. 11, no. 2, pp. 1682–1689, 2025
- M. Mantovani, M. Catellani, and L. Sabattini, "Distributed multi-robot ergodic coverage control for estimating time-varying spatial processes," *IEEE Robotics and Automation Letters*, 2026
- L. Pan, M. Catellani, L. Sabattini, and N. Ayanian, "Online learning-enhanced high order adaptive control barrier functions using neural odes," *submitted to IEEE Robotics and Automation Letters*, 2026

Conference papers

- M. Catellani, F. Pratissoli, F. Bertoncelli, and L. Sabattini, "Coverage control for exploration of unknown non-convex environments with limited range multi-robot systems," in *International Symposium on Distributed Autonomous Robotic Systems*. Springer, 2022, pp. 550–562
- M. Catellani, E. Mazzocco, F. Bertoncelli, and L. Sabattini, "Distributed control for human-swarm interaction in non-convex environments using gaussian mixture models," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 3314–3319, 2023
- M. Catellani, F. Nironi, and L. Sabattini, "A mixed reality interface for human-swarm interaction," in *European Robotics Forum*. Springer, 2024, pp. 112–117

- M. Catellani, M. Belal, and L. Sabattini, "Dual-layer control architecture for cooperative environmental monitoring in multi-robot systems," in *International Symposium on Distributed Autonomous Robotic Systems*. Springer, 2024, pp. 15–27

- M. Catellani, A. Alboni, and L. Sabattini, "Distributed multi-robot control for streets surveillance from aerial images with neural networks," *IFAC-PapersOnLine*, vol. 59, no. 18, pp. 175–180, 2025

- M. Catellani and L. Sabattini, "Uncertainty-aware multi-robot flocking via learned state estimation and control barrier functions," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025

- M. Catellani, M. Gabbi, and L. Sabattini, "Hmpcc: Human-aware model predictive coverage control," in *IEEE International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, 2025

- M. Catellani, M. Mantovani, M. Montanari, and L. Sabattini, "Decentralized learning-based coverage control for multi-robot systems with obstacle awareness: A cnn-driven approach," *submitted to IFAC-PapersOnLine*, 2026

Chapter 1

Introduction

Precision agriculture is rapidly advancing through the deployment of drone swarms and autonomous ground robots, which monitor crop health, apply fertilizers, and harvest with minimal human involvement. These robotic systems enable large-scale operations that can cover hundreds of acres in just a few hours, leading to increased productivity and sustainability and providing real-time data to optimize decision-making [14, 15]. Early implementations of autonomous robots in agriculture have shown yield improvements up to 30%, while minimizing waste and reducing the need for human intervention, thus reducing labor costs [16]. These developments are crucial to face rising global food demands, a compelling challenge for our society as we must not only increase production to feed a growing population, but also ensure our solutions are efficient and sustainable, given the constraints of limited natural resources and the urgent need to reduce environmental pollution.

Logistics and transportation are also experiencing an increasing adoption of robotic teams. Many tech companies such as Google, Amazon, and Uber are starting to test automated last-mile delivery with drones and delivery robots combined with autonomous driving vehicles. Many startups also race for autonomous transportation, such as Starship which currently employs 2700 robots and recently reached 9M deliveries [17]. Results have shown that effective coordination among the team reduces congestion and improves delivery time, making these systems expected to handle a significant share of deliveries in the next few years. Drones have been adopted for missions where minimizing travel time is critical, such as medical supply deliveries in remote regions. Zipline specialized in autonomous delivery of vaccines, blood products, and medications, dramatically reducing delivery times from hours or days to minutes, enhancing healthcare accessibility and saving lives in challenging environments [18].

These are some examples of real-world applications where multi-robot systems have been successfully deployed to assist humans in various tasks, improving efficiency, safety, and productivity. A multi-robot system (MRS) is defined as a group of robots that work collaboratively to achieve a common goal or perform tasks that would be difficult or impossible for a single robot to accomplish alone [19]. Their potential is vast, and ongoing research continues to explore new possibilities, aiming to revolutionize a wide

range of industries and applications in the near future, as confirmed by recent reports [20, 21]. Coordination among the robots is essential to ensure effective collaboration and task execution, and it can be achieved through various strategies, mainly classified into centralized, decentralized, and hybrid approaches.

In the last years, significant improvements have been made by researchers in the field of multi-robot systems, following the idea that having a group of robots solving a complex task is more efficient and cheaper than using a single robot with high capabilities [22]. Among the main advantages of exploiting a multi-robot system, we can list:

- Robustness: with more robots working together, specifically with a decentralized approach, the system is redundant to single points of failure. If one robot malfunctions or is damaged, the others can continue the task, ensuring overall system reliability and robustness.
- Scalability: MRS can be easily scaled up or down by adding or removing robots from the team, allowing for flexible adaptation to different task requirements and environments.
- Flexibility: a team of robots can adapt to various tasks and environments more easily than a single robot. They can reconfigure themselves, change formations, and adjust their strategies based on the task at hand.
- Efficiency: multi-robot systems can usually complete tasks faster than a single robot, as they can divide the workload and work in parallel. This is particularly useful when dealing with large areas or complex tasks.
- Cost-effectiveness: using multiple simpler and cheaper robots can be more cost-effective than employing a single complex and expensive robot.

Among the main application domains of multi-robot systems, we can find search and rescue [23, 24], environmental monitoring [25, 26] and exploration [27], agriculture [15, 28], logistics and transportation [29, 30]. As previously mentioned, these applications are becoming more and more relevant in real world scenarios, with several companies starting to deploy multi-robot systems to improve their operations. However, despite the significant advancements in recent years, there are still several challenges and open problems that need to be addressed to fully realize the potential of multi-robot systems in practical applications.

Together with the advantages, adopting a multi-robot system also introduces several challenges and complexities that need to be carefully addressed. First of all, coordination among the robots is essential to ensure effective collaboration and task execution. A centralized approach is usually more efficient in terms of performace, but does not allow scalability and presents single points of failure, undermining the robustness of the system. For this reason, decentralized approaches appear as a promising solution to overcome these issues, but they require more complex control strategies to ensure that the robots can coordinate their actions without a central controller. Communication among the robots is another critical point, as it is essential for sharing information and coordinating actions [31]. However, communication can be limited by range and bandwidth constraints, or even completely denied in some scenarios [32]. For this reason, control strategies that rely only on local sensing capabilities and do not require communication are gaining increasing interest among researchers. Moreover, multi-robot systems often

operate in dynamic and uncertain environments, which can introduce additional challenges such as changing conditions and unpredictable events. Several control strategies have been proposed to address these challenges, but many of them still rely on ideal assumptions that make the deployment in real world scenarios difficult. Hence, a significant research effort is still required to develop robust and adaptive control strategies that can effectively handle the complexities of real-world environments.

In the following Section, we introduce the studies described in this thesis together with the aspects and issues of the investigated methodologies.

1.1 Contribution and Thesis Outline

Multi-robot systems have been widely studied in recent years, with several control solutions proposed to improve coordination strategies and to face various scenarios. However, the main focus of researchers is recently shifting towards the practical deployment of such systems in real world applications. Hence, a lot of effort is being put into developing control strategies that can effectively deal with the challenges and non-idealities introduced by real world implementations. Among these challenges, we can find:

- Limited sensing and communication capabilities: real sensors have a limited range and accuracy, and communication can be unrealiable or even completely unavailable in some scenarios. Moreover, sensors may have anisotropic characteristics, e.g., cameras have a limited cone-shaped field of view. When dealing with these limitations, control strategies must consider that some information may not be available to the robots.
- Uncertainty and disturbances: besides being limited, real sensors are also affected by noise and disturbances that can make the measurements not accurate. Moreover, robots may be affected by disturbances that can influence their motion, such as wind for aerial platforms or uneven terrain for ground robots. Unexpected motion dynamics may also arise from an inaccurate modeling of the robot, which is often simplified to make the control design easier, but may not capture all the real dynamics.
- Dynamic and uncertain environments: real environments are often dynamic and uncertain, with changing conditions and unpredictable events. When dealing with limited sensing capabilities, information acquired from the environment may quickly become outdated, requiring solutions capable of updating previously acquired information and adapting to new conditions.
- Hardware validation: uncertainties and non-idealities can be modeled in simulation, but real world experiments are essential to validate the proposed methodologies. Moreover, real implementations often introduce additional challenges that are not present in simulations, such as hardware limitations and environmental factors. Hence, conducting real world experiments is crucial to ensure the feasibility and effectiveness of the proposed control strategies.

In this thesis, we focus on developing control strategies for multi-robot systems that can effectively deal with the challenges and non-idealities introduced by real world imple-

mentations. In particular, we focus on decentralized control strategies that rely on local sensing as the primary source of information, and we further investigate solutions applicable to communication-denied scenarios as an extension to limited-range conditions. In addition, we investigate anisotropic sensing conditions, where each robot's sensors have a limited angular field of view. Throughout the thesis, we combine traditional control methods with machine learning techniques to enhance the adaptability and performance of the proposed strategies. Finally, we analyze uncertainties and disturbances that may affect the motion of a robot, such as wind or unmodeled dynamics. Most of the proposed methodologies are validated through realistic simulations, and some of them are also tested in real world experiments with Unmanned Aerial Vehicles (UAVs) and ground robots.

The dissertation is organized as follows.

In Chapter 2, we introduce the mathematical notation and the background concepts that will be used throughout the thesis. In particular, we introduce Voronoi partitioning and Voronoi-based coverage control, which play a key role as core components in many of the algorithms and frameworks developed in the following chapters. Subsequently, we present a limited-range extension to the classical coverage control approach, which allows to effectively deal with limited sensing and communication capabilities. Finally, we introduce Control Barrier Functions (CBFs) and Control Lyapunov Functions (CLFs), two powerful tools that ensure safety and stability in control systems, respectively.

Chapter 3 explores the problem of deploying a team of robots with limited sensing and communication capabilities to monitor and explore an unknown environment. Human intervention is also investigated, both as an external supervisor that can provide high-level guidance to the robot team, and as a dynamic obstacle that the robots must avoid during their operations. In particular, Section 3.1 focuses on the exploration of unknown non-convex environments. Building upon limited-range coverage control, we extend the control strategy by modeling a shared density of the environment that repels the robots from obstacles and areas that have already been explored. Following pheromone-based strategies observed in nature, we allow robots to deploy virtual markers in the environment that modify the underlying density, driving the team to unexplored areas. Additionally, the effect of the markers decays over time, enabling continual re-exploration of the environment. This study has produced the following publication:

- **Coverage Control for Exploration of Unknown Non-Convex Environments with Limited Range Multi-Robot Systems**
  Mattia Catellani, Federico Pratissoli, Filippo Bertoncelli, and Lorenzo Sabattini
  The 16th International Symposium on Distributed Autonomous Robotic Systems (DARS), 2022

In Section 3.2 and Section 3.3 we study how human intervention can enhance the performance of a multi-robot system in monitoring the environment. Firstly, we consider the human operator as an external supervisor, possibly with global knowledge of the environment, who can indicate high-interest areas to be covered by the robotic team. We develop a graphical interface that allows the operator to interactively define regions of interest on a map, which are then translated into a density function guiding the robots'

coverage behavior. The density is modeled as a Gaussian Mixture Model (GMM) to effectively represent the non-uniform importance of different areas, according to the operator's input. Secondly, following a similar approach, we replace the graphical interface with a Mixed Reality headset that allows the operator to directly define areas of interest while being physically present in the environment. The headset overlays virtual markers onto the real world, enabling intuitive interaction. The study of this topic has produced the following publications:

- **Distributed Control for Human-Swarm Interaction in Non-Convex Environments Esing Gaussian Mixture Models**
  Mattia Catellani, Eloisa Mazzocco, Filippo Bertoncelli, Lorenzo Sabattini
  IFAC-PapersOnLine, 2023.
- **A Mixed Reality Interface for Human-Swarm Interaction**
  Mattia Catellani, Flavia Nironi, Lorenzo Sabattini
  European Robotics Forum (ERF), 2024.

Finally, in Section 3.4 we further investigate the coverage problem with limited-range multi-robot systems, considering humans as dynamic obstacles that the robots must avoid during their operations. Since humans' motion is often unpredictable, our approach encompasses a prediction module based on past observations, which estimates the future trajectory of humans and the related uncertainty, growing along the prediction horizon. Then, we formulate a Model Predictive Control (MPC) problem that seeks to optimize the coverage performance of the team while ensuring safety with respect to predicted human positions. Probabilistic contstraints are employed to account for the uncertainty in human motion, ensuring that the robots maintain a safe distance with a specified confidence level. This study has produced the following publication:

- **HMPCC: Human-Aware Model Predictive Coverage Control**
  Mattia Catellani, Marta Gabbi, Lorenzo Sabattini
  IEEE International Symposium on Multi-Robot and Multi-Agent Systems (MRS), 2025.

In Chapter 4, we address the challenges arising from anisotropic sensing conditions, in which each robot's perception is constrained to a limited visual cone. This scenario is particularly relevant for robots equipped with cameras or directional sensors. Starting from Section 4.1, we develop a coordination strategy that forces robots to maintain visibility of each other while navigatin the environment. We design a Control Barrier Functions-based controller that constrains each robot's motion to keep neighboring robots within its visual field. This approach ensures continuous visibility, which is crucial when visual information is the only source of data for coordination. The research on this topic has produced the following publication:

- **Directed Graph Topology Preservation in Multi-Robot Systems with Limited Field of View Using Control Barrier Functions**
  Filippo Bertoncelli, Vivek Radhakrishnan, Mattia Catellani, Giuseppe Loianno, Lorenzo Sabattini
  IEEE Access, 2024.

Building upon the concept of maintaining visibility among robots with limited angular

fields of view, we consider scenarios in which communication is completely denied, eliminating any form of explicit coordination, and where the size of the robotic team is such that maintaining mutual visibility among all robots is no longer feasible. In Section 4.2 we study how a team of robots can still fulfill the desired task under these conditions, relying solely on local visual information. Our approach makes use of a state estimator, namely a particle filter, to infer the positions of neighboring robots based on intermittent visual observations. The estimated positions, together with their associated uncertainty, are then incorporated into a Control Barrier Functions-based controller to achieve safe task execution. Additionally, the specific design of the CBFs, which also incorporate CLF properties, enables each robot to bring other robots back into its field of view when the uncertainty in their positions becomes too high, potentially leading to unsafe situations. Hence, uncertainty evaluation leads to an exploration-exploitation trade-off, where robots must balance between gaining new information about their neighbors and exploiting the current knowledge to perform the task safely. This study has produced the following publication:

- **Distributed Control of a Limited Angular Field-of-View Multi-Robot System in Communication-Denied Scenarios: A Probabilistic Approach**
  Mattia Catellani, Lorenzo Sabattini
  IEEE Robotics and Automation Letters, 2023.

Control Barrier Functions have proven to be a powerful tool for ensuring safety in control systems. However, they are usually employed to design reactive controllers, leading to short-sighted behaviors that may not be optimal in the long term. To address this limitation, in Section 4.3 we propose to integrate CBFs within a Model Predictive Control (MPC) framework. Building upon our previous work on neighbors' state and uncertainty estimation, we formulate an MPC problem that optimizes each robot's trajectory over a finite horizon, optimizing the exploration-exploitation trade-off. This research has produced the following publication:

- **Robust Trajectory Generation and Control for Quadrotor Motion Planning with Field-of-View Control Barrier Certification**
  Lishuo Pan, Mattia Catellani, Lorenzo Sabattini, Nora Ayanian
  IEEE Robotics and Automation Letters, 2025.

Further investigation on limited field of view led us to explore the challenge of cooperative target tracking in Section 4.4. We consider a team of robots with limited angular field of view that must monitor a set of targets moving in the environment. The robots are divided into smaller groups, and each group is responsible for tracking a specific target. To ensure effective tracking, we design a dual-layer control architecture: at the lower layer, each robot employs a CBF-based controller to maintain visibility of its assigned target while avoiding collisions with other robots; at the higher layer, a coordination strategy optimizes the positions of the groups to maximize overall tracking performance. This hierarchical approach allows the system to scale to larger teams while ensuring effective target tracking. This study has produced the following publication:

- **Dual-Layer Control Architecture for Cooperative Environmental Monitoring in Multi-Robot Systems**

Mattia Catellani, Mehdi Belal, Lorenzo Sabattini
International Symposium on Distributed Autonomous Robotic Systems (DARS), 2024.

In Chapter 5, we explore how machine learning techniques can enhance the performance and adaptability of robots. Section 5.1 focuses on surveillance tasks, where a team of UAVs must monitor roads. The developed approach is similar to those already mentioned, encoding the problem into a coverage control framework, where the density distribution highlights areas of interest to be monitored, in this case roads. However, instead of manually defining the density, we employ a Convolutional Neural Network (CNN) to automatically extract roads from aerial images and generate the corresponding density map by fitting a Gaussian Mixture Model to the detected road pixels. This study has produced the following publication:

- **Distributed Multi-Robot Control for Streets Surveillance from Aerial Images With Neural Networks**
  Mattia Catellani, Andrea Alboni, Lorenzo Sabattini
  IFAC-PapersOnLine, 2025.

CNNs have also been employed in Section 5.2 to enhance the coverage performance of a multi-robot system in a complex scenario. We develop a learning-based approach that leverages a CNN to map the current observation of each robot to a desired control command that optimizes the coverage performance. The CNN is trained using data generated from a traditional coverage control algorithm, allowing the robots to learn effective coverage strategies from expert demonstrations. The observation is encoded into three channels: the local density map, an occupancy map of the surrounding environment, and a map of detected neighboring robots. This rich representation enables the CNN to capture the essential features of the environment and the team configuration, leading to effective coverage performance even in non-convex environments, where traditional methods may struggle. This study has produced the following publication:

- **Decentralized Learning-Based Coverage Control for Multi-Robot Systems with Obstacle Awareness: A CNN-Driven Approach**
  Mattia Catellani, Mattia Mantovani, Marco Montanari, Lorenzo Sabattini
  submitted to IFAC-PapersOnLine, 2026.

Then, recalling the challenges introduced by intermittent detections and measurements of neighboring robots, in Section 5.3 we propose a learning-based approach to enhance the state estimation process. We develop a Neural Network that mimics the behavior of a particle filter, learning to estimate the positions and covariance matrices of neighboring robots based on intermittent observations. The Neural Network is trained from data generated by a particle filter, allowing it to learn from expert demonstrations, but with significantly reduced computational complexity, further enhanced by parallel processing on GPUs. The estimated states are then employed within a CBF-based controller to achieve safe flocking behavior. This study has produced the following publication:

- **Uncertainty-Aware Multi-Robot Flocking via Learned State Estimation and Control Barrier Functions**
  Mattia Catellani, Lorenzo Sabattini

IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2025.

A different type of Machine Learning technique is explored in Section 5.4, where we investigate how Gaussian Processes (GPs) can be employed for tasks requiring monitoring of unknown spatial phenomena. We develop an ergodic control strategy that allows a team of robots to effectively monitor an environment while simultaneously learning a GP model of an underlying spatial phenomenon. The ergodic controller balances exploration and exploitation, ensuring that the robots visit areas of high uncertainty to improve the GP model, while also focusing on regions of interest defined by the current GP mean. This approach enables the robots to adaptively refine their monitoring strategy based on the learned model. We validate the proposed methodology through simulations and real-world experiments with UAVs, demonstrating its effectiveness and real-time applicability. This study has produced the following article:

- **Distributed Multi-Robot Ergodic Coverage Control for Estimating Time-Varying Spatial Processes**
  Mattia Mantovani, Mattia Catellani, Lorenzo Sabattini
  IEEE Robotics and Automation Letters, 2025.

Instead, Section 5.5 focuses on single-robot scenarios as the groundwork for subsequent extensions to multi-robot systems, proposing a learning-based framework for integrating CBFs into control systems affected by modeling errors or external disturbances. Our proposed solution makes use of Neural Ordinary Differential Equations (NODEs) to approximate the unmodeled residuals representing the mismatch between the nominal and true robot dynamics. NODEs allow to encode the learned model into CBF constraints, thereby enhancing safety compared to traditional adaptive control methods, as demonstrated by simulated and real-world experiments on a quadrotor subject to wind disturbances. This study has produced the following publication:

- **Online Learning-Enhanced High Order Adaptive Safety Control**
  Lishuo Pan, Mattia Catellani, Lorenzo Sabattini, Nora Ayanian
  submitted to IEEE Robotics and Automation Letters, 2025.

Finally, Chapter 6 concludes the thesis by summarizing the main results and outlining current limitations, also discussing how future research could address these challenges.

Chapter 2

Preliminaries

In this Chapter, we introduce the necessary background concepts and notation that will be used throughout this thesis. We begin by presenting a description of the multi-robot coverage control problem and its extension to limited-range settings in Section 2.1, which form the basis for our contributions. Then, Section 2.2 presents Control Barrier Functions and Control Lyapunov Functions, which we will leverage as key tools for ensuring safety and stability in our proposed control strategies.

2.1 Multi-Robot Coverage Control

Coverage control is a fundamental problem in multi-robot systems, where a team of robots is tasked with optimally covering the environment to monitor or service it effectively. The goal is to position the robots to maximize area coverage while minimizing overlap and ensuring efficient use of resources. Furthermore, given a probability density function that represents the importance of different regions in the environment, coverage control aims to prioritize areas based on their significance. The seminal work by Cortés et al. [33] introduced a distributed algorithm for multi-robot coverage control using Voronoi partitions, which has since become a foundational approach in this field. The main idea is to partition the environment into regions, each assigned to a robot, and then iteratively adjust the robots' position and update the partitioning until convergence to an optimal configuration is achieved.

Let $\mathbb{F}(\mathbb{R}^2)$ be the collection of finite point sets in $\mathbb{R}^2$. We denote an element of $\mathbb{F}(\mathbb{R}^2)$ as $\mathcal{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_N\} \subset \mathbb{R}^2$, where $\{\mathbf{p}_1, \ldots, \mathbf{p}_N\}$ are points in $\mathbb{R}^2$. In the rest of the thesis, the notation $\{\mathbf{p}_i\}_{i=1}^N$ will also be used to indicate sets. The environment where robots are required to operate will be indicated as $Q \subset \mathbb{R}^2$, and it is assumed to be convex and bounded. An arbitrary point in the environment will be denoted by $\mathbf{q} \in Q$. Given a generic vector $\mathbf{x} \in \mathbb{R}^n$, let $\|\mathbf{x}\|$ be its Euclidean norm. Following [33], we define the *Voronoi partition* of the environment Q generated by the set of points $\mathcal{P}$ as the collection of regions $\mathcal{V}(\mathcal{P}) = \{V_i\}_{i=1}^N$ where

$$V_i = \{\mathbf{q} \in Q \mid \|\mathbf{q} - \mathbf{p}_i\| \leq \|\mathbf{q} - \mathbf{p}_j\|, \forall j \neq i\}, \quad i = 1, \ldots, N. \tag{2.1}$$

In order to achieve optimal coverage of the environment, we can formulate an optimization problem aiming to maximize a performance function that quantifies how reliable is the measurement of a point $\mathbf{q} \in Q$ by robot i located at position $\mathbf{p}_i$. A common choice for such performance metric is a non-increasing differentiable function of the Euclidean distance $\|\mathbf{q}-\mathbf{p}_i\|$, namely $f(\|\mathbf{q}-\mathbf{p}_i\|) : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$. In addition, we consider a probability density function $\phi : Q \to \mathbb{R}_{\geq 0}$ that characterizes the importance of different regions in the environment Q. This density function may be known a priori to all the robots, or may be estimated online through sensor measurements, as in [34].

Given the Voronoi partition $\mathcal{V}(\mathcal{P})$ of the environment Q, the multi-robot coverage control problem can be formulated to maximize the optimization function $\mathcal{H} : Q \to \mathbb{R}$ defined as:

$$\mathcal{H}(\mathcal{P}, \mathcal{V}) = \sum_{i=1}^{N} \int_{V_i} f(\|\mathbf{q} - \mathbf{p}_i\|)\phi(\mathbf{q})d\mathbf{q}, \tag{2.2}$$

where each robot i is responsible for covering its Voronoi region V_i, and higher values of the function correspond to better coverage of the environment. In the literature, the performance function f is often chosen as $f(\|\mathbf{q} - \mathbf{p}_i\|) = -\|\mathbf{q} - \mathbf{p}_i\|^2$, leading to:

$$\mathcal{H}(\mathcal{P}, \mathcal{V}) = -\sum_{i=1}^{N} \int_{V_i} \|\mathbf{q} - \mathbf{p}_i\|^2\phi(\mathbf{q})d\mathbf{q}. \tag{2.3}$$

In order to solve the optimization problem in (2.3), the gradient of the optimization function with respect to the robot positions $\{\mathbf{p}_i\}_{i=1}^N$ can be computed as:

$$\frac{\partial \mathcal{H}(\mathcal{P}, \mathcal{V})}{\partial \mathbf{p}_i} = 2M_{V_i}(\mathbf{C}_{V_i} - \mathbf{p}_i), \quad i = 1, \ldots, N, \tag{2.4}$$

where M_{V_i} and $\mathbf{C}_{V_i}$ denote the mass and the centroid of the Voronoi region V_i with respect to the density function ϕ, respectively, and can be computed as:

$$M_{V_i} = \int_{V_i} \phi(\mathbf{q})d\mathbf{q}, \quad \mathbf{C}_{V_i} = \frac{1}{M_{V_i}} \int_{V_i} \mathbf{q}\phi(\mathbf{q})d\mathbf{q}. \tag{2.5}$$

According to (2.4), the configuration of points $\mathcal{P}$ that maximizes the optimization function $\mathcal{H}$ corresponds to the one where each point $\mathbf{p}_i$ coincides with the centroid $\mathbf{C}_{V_i}$ of its Voronoi region V_i. Such configurations are referred to as *centroidal Voronoi configurations* [35]. $\mathcal{H}(\mathcal{V}, \mathcal{P})$ can be used as a Lyapunov function to design a distributed control law that steers the robots towards a centroidal Voronoi configuration, maximizing the coverage of the environment. According to the Lloyd algorithm [36], considering single-integrator dynamics $\dot{\mathbf{p}}_i = \mathbf{u}_i$ for each robot i, where $\mathbf{p}_i$ and $\mathbf{u}_i$ denote the position and the control input of robot i, respectively, a distributed control law can be designed as:

$$\mathbf{u}_i = k(\mathbf{C}_{V_i} - \mathbf{p}_i), \quad i = 1, \ldots, N, \tag{2.6}$$

where $k \in \mathbb{R}_{>0}$ is a proportional gain. As proven in [33], updating the Voronoi partition $\mathcal{V}(\mathcal{P})$ at each time instant based on the current configuration $\mathcal{P}$ of the team, and applying the control law in (2.6), guarantees that the robots asymptotically converge to a centroidal Voronoi configuration, maximizing the coverage of the environment. An illus-

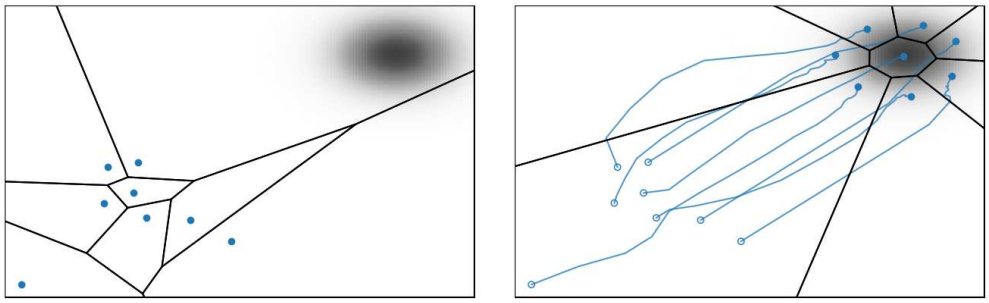


Figure 2.1: Example of multi-robot coverage control using Voronoi partitions. Left: initial positions of the robots and corresponding Voronoi partition. Right: final positions of the robots after applying the coverage control law, achieving a centroidal Voronoi configuration and optimally covering the region of interest.

trative example of a coverage control task using Voronoi partitions is shown in Figure 2.1.

The work by Cortés et al. [33] stands as a milestone in the field of coverage control, providing a robust framework for multi-robot systems to achieve optimal area coverage through Voronoi partitions. However, the original formulation assumes that each robot has global knowledge of the environment and can compute its Voronoi region based on the positions of all other robots. This assumption may not hold in practical scenarios where robots have limited sensing and communication capabilities, restricting their ability to gather information.

2.1.1 Limited-Range Coverage Control

To address the limitations of global knowledge in coverage control, recent research has focused on developing strategies that enable robots to operate effectively under limited-range sensing and communication constraints. In such scenarios, each robot can only detect and communicate with other robots within a certain radius, which significantly impacts the computation of Voronoi partitions and the overall coverage strategy. One approach to tackle this challenge is to modify the Voronoi partitioning process to account for the limited-range capabilities of the robots. This can be achieved by defining local Voronoi regions based on the positions of neighboring robots within the sensing radius, rather than considering the entire team. This localized approach allows robots to make decisions based on the information available to them, leading to more practical and scalable coverage strategies. Building upon this idea, Pratissoli et al. [37] proposed a distributed algorithm for limited-range multi-robot coverage control. Let us denote by $R \in \mathbb{R}_{>0}$ the sensing radius of each robot. We define as $\overline{B}(\mathbf{p}_i, r)$ the ball with radius equal to half the sensing range $r = R/2$ centered at the position of robot i. The *limited-range Voronoi partition* of the environment Q generated by the set of points $\mathcal{P}$ is defined as the collection of regions $\mathcal{V}^r(\mathcal{P}) = \{V_i^r\}_{i=1}^N$ where

$$V_i^r = \{\mathbf{q} \in \overline{B}(\mathbf{p}_i, r) \cap Q \mid \|\mathbf{q} - \mathbf{p}_i\| \leq \|\mathbf{q} - \mathbf{p}_j\|, \forall j \neq i\}, \quad i = 1, \ldots, N. \tag{2.7}$$

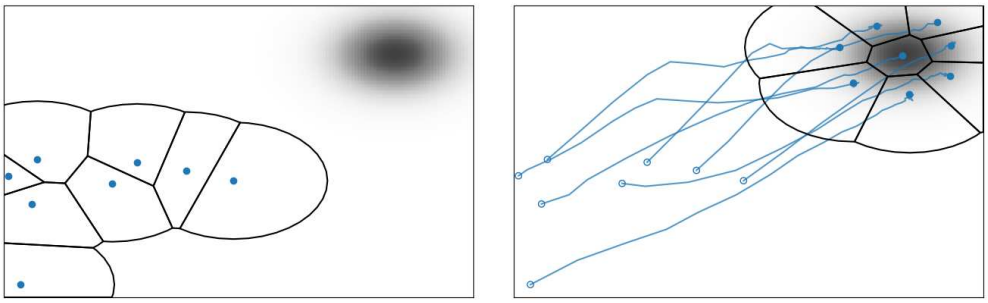


Figure 2.2: Example of limited-range multi-robot coverage control. Left: initial positions of the robots and corresponding limited-range Voronoi partition. Right: final positions of the robots after applying the distributed limited-range coverage control law, achieving a limited-range centroidal Voronoi configuration and optimally covering the region of interest.

It is important to note that considering the half-range r is crucial in order to generate a valid partitioning. Indeed, if the full sensing range R were used instead, it could lead to overlapping regions between neighboring robots, thus violating the partition property. By restricting the Voronoi regions to half the sensing range, we ensure that each robot's coverage area is distinct and non-overlapping (see [37, Remark 2] for more details).

Similar to the standard coverage control problem, we can define a performance function $f^r : \mathbb{R}_{>0} \to \mathbb{R}_{>0}$ indicating the quality of coverage provided by the robots with limited-range sensing as:

$$f^r(x) = \begin{cases} -x^2 & \text{if } x \leq r \\ -r^2 & \text{otherwise.} \end{cases} \tag{2.8}$$

Note that $f^r(x)$ is non-increasing and differentiable even outside the sensing range, as required. The limited-range optimization function $\mathcal{H}^r : Q \to \mathbb{R}$ can then be defined by modifying (2.2) as:

$$\mathcal{H}^r(\mathcal{P}, \mathcal{V}^r) = -\sum_{i=1}^{N} \int_{V_i^r} \|\mathbf{q} - \mathbf{p}_i\|^2 1_{[0,r]} \phi(\mathbf{q}) d\mathbf{q} - r^2 \sum_{i=1}^{N} \int_{V_i^r} 1_{(r,\infty)} \phi(\mathbf{q}) d\mathbf{q}. \tag{2.9}$$

Solving the above optimization problem leads to the definition of a distributed control law that steers the robots towards the centroid of their limited-range Voronoi regions, namely:

$$\mathbf{u}_i = k(\mathbf{C}_{V_i^r} - \mathbf{p}_i), \quad i = 1, \ldots, N, \tag{2.10}$$

where $\mathbf{C}_{V_i^r}$ denotes the centroid of the limited-range Voronoi region V_i^r. As demonstrated in [37, Theorem 1], applying the control law in (2.10) while updating the limited-range Voronoi partition $\mathcal{V}^r(\mathcal{P})$ at each time instant based on the current configuration $\mathcal{P}$ of the team, guarantees that the robots asymptotically converge to a *limited-range centroidal Voronoi configuration*, where each robot i is located at the centroid $\mathbf{C}_{V_i^r}$ of its limited-range Voronoi region V_i^r, thus maximizing the coverage of the environment even under limited-range sensing constraints. An example of a coverage control task using limited-range Voronoi partitions is illustrated in Figure 2.2.

2.2 Control Barrier Functions and Control Lyapunov Functions

Control Barrier Functions (CBFs) and Control Lyapunov Functions (CLFs) are powerful tools in control theory that enable the design of controllers to ensure safety and stability properties of dynamical systems. CBFs are primarily used to enforce safety constraints by ensuring that the system's state remains within a safe set, while CLFs are employed to guarantee the stability of the system by driving the state towards a desired equilibrium point. By combining CBFs and CLFs, it is possible to design controllers that simultaneously achieve safety and stability objectives, which is particularly useful in multi-robot systems where both aspects are critical.

Consider a system in the form

$$\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u}, \tag{2.11}$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are Lispschitz continuous functions, and $\mathbf{u} \in U \subset \mathbb{R}^m$ is the control input, where U is the set of admissible control inputs for $\mathbf{u}$. Let $\mathcal{C} := \{\mathbf{x} \in \mathbb{R}^n \mid b(\mathbf{x}) \geq 0\}$ be the set of configurations satisfying the safety requirements, i.e., the safe set.

Definition 2.2.1 (Class $\mathcal{K}$ and extended class $\mathcal{K}$ functions)**.** A continuous function $\alpha : [0, a) \to [0, \infty)$ with $a > 0$ is a class $\mathcal{K}$ function if it is strictly increasing and $\alpha(0) = 0$. If $\alpha : \mathbb{R} \to \mathbb{R}$, then α is said to belong to extended class $\mathcal{K}$.

Definition 2.2.2 (CBF [38, 39])**.** Given a set $\mathcal{C}$, the function $b : \mathbb{R}^n \to \mathbb{R}$ is a candidate CBF for system (2.11) if there exists a class $\mathcal{K}$ function α such that

$$\sup_{u\in U}[L_f b(\mathbf{x}) + L_g b(\mathbf{x})\mathbf{u} + \alpha(b(\mathbf{x}))] \geq 0, \tag{2.12}$$

for all $\mathbf{x} \in \mathcal{C}$, where L_f and L_g are the Lie derivatives along f and g, respectively.[1] According to [39], given a CBF b and a safe set $\mathcal{C}$, any Lipschitz continuous controller $\mathbf{u}(t)$ that satisfies (2.12) makes the set $\mathcal{C}$ *forward invariant* for system (2.11), i.e., if $\mathbf{x}(t_0) \in \mathcal{C}$, then $\mathbf{x}(t) \in \mathcal{C}$, $\forall t \geq t_0$.

With this condition, it is possible to synthesize optimization-based controllers that modify the desired control input $\mathbf{u}_{\text{des}}$ in a minimally invasive way, solving the following optimization problem:

$$\begin{aligned} \mathbf{u}(\mathbf{x}) = &\underset{\mathbf{u}\in U}{\operatorname{argmin}} \quad \|\mathbf{u} - \mathbf{u}_{\text{des}}\|^2 \\ &\text{s.t.} \quad L_f b(\mathbf{x}) + L_g b(\mathbf{x})\mathbf{u} + \alpha(b(\mathbf{x})) \geq 0. \end{aligned} \tag{2.13}$$

Therefore, this controller aims at providing the control input $\mathbf{u}$ closest to the desired input $\mathbf{u}_{\text{des}}$ while ensuring the safety of the system by enforcing the forward invariance of the safe set $\mathcal{C}$.

[1] The Lie derivative evaluates the change of a function along a vector field (see [40]).

It is important to note that the controller $\mathbf{u}(t)$ does not guarantee convergence to the set $\mathcal{C}$ if the system starts outside of it. For this reason, we introduce Control Lyapunov Functions (CLFs).

Definition 2.2.3 (CLF [41])**.** A continuously differentiable function $V : \mathbb{R}^n \to \mathbb{R}$ is a globally and exponentially stabilizing CLF for (2.11) if there exists a class $\mathcal{K}$ function ζ such that

$$\inf_{\mathbf{u}\in U}[L_f V(\boldsymbol{x}) + L_g V(\mathbf{x})\mathbf{u} + \zeta(V(\mathbf{x}))] \leq 0. \tag{2.14}$$

CLFs properties ensure that a controller $\mathbf{u}(t)$ that satisfies (2.14) stabilizes the system to a point $\mathbf{x}^*$ or a set [41].

CLFs can be easily integrated with CBFs in the same optimization-based controller, solving the following optimization problem:

$$\begin{aligned} \mathbf{u}(\mathbf{x}) = \underset{\mathbf{u}\in U,\ \delta\in\mathbb{R}_{\geq 0}}{\operatorname{argmin}} \quad & \|\mathbf{u} - \mathbf{u}_{\text{des}}\|^2 + \delta^2 \\ \text{s.t.} \quad & L_f b(\mathbf{x}) + L_g b(\mathbf{x})\mathbf{u} + \alpha(b(\mathbf{x})) \geq 0, \\ & L_f V(\mathbf{x}) + L_g V(\mathbf{x})\mathbf{u} + \zeta(V(\mathbf{x})) \leq \delta. \end{aligned} \tag{2.15}$$

Here, a slack variable $\delta \in \mathbb{R}_{\geq 0}$ is introduced to relax the CLF constraint, ensuring that the optimization problem is always feasible. The controller synthesized by solving (2.15) guarantees safety by enforcing the forward invariance of the safe set $\mathcal{C}$ through the CBF constraint, while also promoting stability towards a desired equilibrium point or set via the CLF constraint.

2.2.1 High-Order Control Barrier Functions

In some scenarios, the safety constraint defined by the function $b(\mathbf{x})$ may not be directly enforceable using standard CBFs, particularly when the relative degree of $b(\mathbf{x})$ with respect to the system dynamics is greater than one.

Definition 2.2.4 (Relative degree)**.** The relative degree $r \in \mathbb{N}$ of a sufficiently differentiable function $b : \mathbb{R}^n \to \mathbb{R}$ with respect to a system is the number of times we need to differentiate along the system dynamics until the control input explicitly appears.

In such cases, High-Order Control Barrier Functions (HOCBFs) can be employed to ensure safety by considering higher-order derivatives of the safety function [42]. HOCBFs extend the concept of CBFs to accommodate functions with higher relative degrees, allowing for the enforcement of safety constraints that depend on the system's state and its derivatives.

Let us consider a set of functions $\{\psi_i\}_{i=1}^r$ with $\psi_i : \mathbb{R}^n \to \mathbb{R}$, defined recursively as:

$$\psi_i(\mathbf{x}) = \dot{\psi}_{i-1}(\mathbf{x}) + \alpha_i(\psi_{i-1}(\mathbf{x})) \tag{2.16}$$

with $\psi_0(\mathbf{x}) = b(\mathbf{x})$. Furthermore, we define a sequence of sets $\{\mathcal{C}_i\}_{i=1}^r$ with $\mathcal{C}_i \subset \mathbb{R}^n$ as:

$$\mathcal{C}_i = \{\mathbf{x} \in \mathbb{R}^n \mid \psi_{i-1}(\mathbf{x}) \geq 0\} \tag{2.17}$$

Definition 2.2.5 (HOCBF [42])**.** Let us consider a sequence of sets $\{\mathcal{C}_i\}_{i=1}^r$, and a sequence of equations $\{\psi_i\}_{i=1}^r$ as previously defined. A function $b : \mathbb{R}^n \to \mathbb{R}$ is a candidate HOCBF of relative degree r for system (2.11) if there exist $(r-i)$-th order differentiable class $\mathcal{K}$ functions $\{\alpha_i\}_{i=1}^r$ such that:

$$\sup_{\mathbf{u} \in U} [L_f^r b(\mathbf{x}) + L_g L_f^{r-1} b(\mathbf{x}) \mathbf{u} + O(b(\mathbf{x})) + \alpha_r(\psi_{r-1}(\mathbf{x}))] \geq 0, \tag{2.18}$$

for all $\mathbf{x} \in \mathcal{C}_1 \cap, \ldots, \cap \mathcal{C}_r$, where $O(b(\mathbf{x}))$ captures the higher-order derivatives of $b(\mathbf{x})$ and is defined as:

$$O(b(\mathbf{x})) = \sum_{i=1}^{r-1} L_f^i(\alpha_{r-i} \circ \psi_{r-i-1})(\mathbf{x}) + \frac{\partial^i(\alpha_{r-i} \circ \psi_{r-i-1})(\mathbf{x})}{\partial t^i} \tag{2.19}$$

Any Lipschitz continuous controller $\mathbf{u}(t) \in U$ that satisfies (2.18) renders the set $\mathcal{C} = \cap_{i=1}^r \mathcal{C}_i$ forward invariant (see [42, Theorem 4]).

Chapter 3

Decentralized Control under Limited-Range Sensing

In this Chapter, we focus on strategies for sending a group of robots with limited sensing and communication abilities into an unknown environment for monitoring and exploration. We also consider human involvement, looking at scenarios where a person acts either as a high-level supervisor directing the robot team or as a moving obstacle that the robots need to safely navigate around.

3.1 Coverage Control for Exploration of Unknown Non-Convex Environments with Limited Range Multi-Robot Systems

This work introduces a novel framework for the exploration and monitoring of unknown environments using a team of coordinated mobile robots. Our approach extends Voronoi-based coverage control by integrating time-varying probability density functions (PDFs). Specifically, we propose a mechanism where robots utilize locally available information to dynamically update the PDF, effectively biasing their trajectories toward previously unvisited regions. The proposed control strategy is validated through extensive simulations and hardware experiments with mobile robots, demonstrating its efficacy in achieving high coverage rates across large environments.

3.1.1 Introduction

The problem of coordinating multiple robots with the task to explore (and map) an unknown environment has been widely studied in the last years. One of the most widely adopted strategies, firstly described in [43], exploits potential fields to generate collision-free trajectories. Every robot is treated as a positive ion, while obstacles are assumed as positively charged and the goal as negatively charged, thus resulting in every robot being driven to the goal while rejecting both other agents and obstacles. Other control strategies can be derived from the well known random walk approach, as the ones proposed in [44], where robots move randomly within the environment.

An alternative approach is the one proposed by [45], where the authors study the problem of multi-robot exploration and coverage with limited communication. They propose a decentralized algorithm based on leaving marks on the unknown terrain, which can be sensed by the robots and allow them to cover the terrain. The limited communication issue has been considered also by [46], where the control of the multi-robot system relies on the communication with a base station. In order to assist the robot coverage of the environment, [47] proposes an algorithm that deploys radio beacons and, similarly, [48] proposes robots that, in the process of moving around, deploy network nodes into the environment.

One of the most relevant and studied exploration methodologies is the frontier based coordination approach [49]. The environment is discretized in standard occupancy grids and a frontier cell consists in each already explored cell that is an immediate neighbor of an unknown, unexplored cell. A mapping of the environment is required, usually generated exploiting grid maps or graphs, to keep updated the portion of the environment already explored and to localize the frontier nodes positions [50].

Coverage based control strategies [33] coordinate a group of autonomous robots with the aim of maximizing the coverage of the environment and generally exploit a Voronoi partitioning of the environment. The tools behind these methodologies can be exploited to deal with the exploration of unknown environments, like in [51], where the Voronoi

diagram is used to generate an obstacle-free convex neighbourhood among strongly convex obstacles. A significant work is presented by [52] where a frontier-based exploration methodology is integrated with a Voronoi partitioning of the environment to avoid duplicated exploration areas. In both of these works the boundary of environment and the obstacles needs to be fully or partially known. A recent study presented in [53] deals with the problem caused by updating a single global grid map managing only local information. In [27] an exploration and navigation framework has been proposed to explore GPS-denied environments using the cooperation between UAVs and UGVs to increase the accuracy of the SLAM algorithm.

The mentioned studies and the majority of the approaches in multi-robot exploration are based on a discretization of the environment, typically exploiting occupancy grids. This approach is not suitable for large environments, where the number of cells grows significantly, thus leading to severe computational and memory consumption. Moreover, these approaches require a strong communication network in order to keep updated the shared map among the robots [54]. A methodology that deals with communication issues has been presented in [55], where the total area is partitioned based on the topology exploiting the Generalized Voronoi Graphs for exploration purposes. However, the robots are constrained to move on the Voronoi diagrams edges. Moreover, this strategy is mostly effective in environments structured in a way that naturally drives robots during exploration, e.g. with corridors, losing effectiveness in more generic spaces. An alternative and relevant approach to obtain a mapping and exploration of large environments is presented by [56]. A Gaussian mixture model for global mapping is used to model a complex environment. This approach, developed and tested for single-robot monitoring, has the benefits to require a small memory footprint and a low volume of communication. Following this strategy, in this work, we propose a methodology that exploits a mixture of Gaussian distributions to coordinate a group of robots and explore an unknown environment. Given the Gaussian function defined by only a few parameters and the lack of a grid discretization of the environment, the strategy aims to an efficient memory usage and lightweight communication among robots.

Contribution Our methodology uses the coverage control theory described in [37], which introduces a limited Voronoi diagram able to deal with unknown and non-convex environments, differently from the approach in [33]. Exploiting this strategy, which allows to obtain an optimal configuration of the agents in the environment in order to maximize the covered area, we provide a solution to dynamically explore an unknown environment while continuously passing through it, in order to keep it constantly monitored. We combine the coverage based control methodology with a mixture of Gaussian distributions to model the areas of the environments already explored. The proposed approach performs the exploration and monitoring without a discretization of the environment (grid map or graph) and without any knowledge of the boundaries of the environment. Moreover, it is worth noting that the few parameters that define a Gaussian function can be easily shared among the robots keeping low the communication and the memory usage, as stated in [56]. Finally, most of the works found in literature provide simple simulations to show the proposed strategy and only a few works can be found that test the control on real platforms. We extensively tested the proposed control in both simulated and real

environment experiments. We compare the proposed approach with other typical ones that do not require a discretization or a partial knowledge of the environment, such as random walk and potential field approaches.

3.1.2 Problem Statement

Consider a multi-robot system constituted by n holonomic robots able to move in two dimensions, controlled with the objective of monitoring and exploring an unknown environment. We assume each robot to be modeled as a single integrator system,[1] whose position $\mathbf{p}_i \in \mathbb{R}^2$ evolves according to $\dot{\mathbf{p}}_i = \mathbf{u}_i$, where $\mathbf{u}_i \in \mathbb{R}^2$ is the control input, $\forall i = 1, \ldots, n$.

We consider the following assumptions:

1. *Localization with respect to a global reference frame*: each robot is able to localize itself with respect to a global reference frame, which is shared among the robots in the team;
2. *Limited sensing capabilities*: each robot is able to measure the position of neighboring robots and objects (including boundaries of the environment), within its limited sensing range (defined by a circle with radius $r_{sens} \in \mathbb{R}_{>0}$).
3. *Shared memory*: robots can access a central server that stores information about the environment's probability density, and can also transmit new data to update this density.

Based on these assumptions, the problem addressed in this Section is then formalized as follows:

Problem. *Define a distributed control strategy that allows a multi-robot system with limited sensing capabilities to explore an arbitrary unknown, possibly non-convex, large environment. The methodology needs to consider the possibility to not be able to discretize the environment due to memory and communication limitations.*

3.1.3 Time-varying probability density function for exploration

As introduced in Section 3.1.2, the aim of this work is to monitor and explore an unknown and possibly non-convex environment with a group of robots with limited sensing capabilities. In order to fulfill the exploration of the whole environment, the robots need to keep updated the areas already explored by the multi-robot system. A probability density function can be exploited for this purpose focusing the attention on the areas to be explored and avoiding areas recently visited. The proposed distributed control algorithm uses this approach to keep updated and shared information about the explored environment.

[1] We would like to remark that, even though the single integrator is a very simplified model, it can still effectively be exploited to control real mobile robots: using a sufficiently good trajectory tracking controller, the single integrator model can be used to generate velocity references for widely used mobile robotic platforms, such as wheeled mobile robots, and unmanned aerial vehicles.

The probability density function is defined as a combination of multiple Gaussian density functions. Each Gaussian function $\varphi(\mathbf{q})$ is determined by a mean value $\overline{\mathbf{q}} \in \mathbb{R}^2$ and a standard deviation $\sigma \in \mathbb{R}_{>0}$, and is defined as follows:

$$\varphi(\mathbf{q}) = \exp\left(-\frac{\|\mathbf{q} - \overline{\mathbf{q}}\|^2}{2\sigma^2}\right). \tag{3.1}$$

Every agent in the environment must be able to access this data when $\overline{\mathbf{q}}$ lies inside its sensing range, thus the exchange of information can be achieved exploiting a central storage unit or physically placing a marker or a beacon that each robot can easily localize. During the motion and the exploration, every robot places several Gaussian functions in the environment. The combination of all the shared Gaussian functions determines the motion of the multi-robot system. Every time a new area is explored by a robot, a new Gaussian function $\varphi(\mathbf{q})$, defined as in (3.1), is added by the robot to the shared density distribution following the algorithm described later (Algorithm 1), affecting the motion of the overall multi-robot system. During the exploration of an unknown environment, we consider two possible scenarios the robot can face: (1) the robot explores areas that need to be crossed only once during the navigation, and (2) areas that need to be crossed more than once to complete the exploration without deadlocks. Thus, we define two types of Gaussian functions to be added by the robot following the definition in (3.1): a time-constant Gaussian function and a time-varying Gaussian function. The first one intuitively remains constant over time during the whole exploration:

$$f_{F_i}(\mathbf{q}) = \exp\left(-\frac{\|\mathbf{q} - \overline{\mathbf{q}}_i\|^2}{2\sigma_i^2}\right). \tag{3.2}$$

Conversely, the time-varying Gaussian function decreases over time: for this purpose, we make use of a *forgetting factor* $k_j(t) \in [0, 1]$, which gradually decreases the intensity of the function over time and is defined as

$$k_j(t) = \begin{cases} 1 - \frac{t - T_{0_j}}{T_{forget}} & \text{if } t \in [T_{0_j}, T_{0_j} + T_{forget}] \\ 0 & \text{otherwise,} \end{cases} \tag{3.3}$$

where T_{0_j} is the time instant at which the Gaussian component is added and T_{forget} represents how long the component should remain active. Thus, the time-varying Gaussian function can be defined as

$$f_{V_j}(\mathbf{q}, t) = \min\left(\exp\left(-\frac{\|\mathbf{q} - \overline{\mathbf{q}}_j\|^2}{2\sigma_j^2}\right), k_j(t)\right) \tag{3.4}$$

where the *forgetting factor* $k_j(t)$, decreasing over time, lowers the peak of the probability function until completely nullifying it at $t = T_{0_j} + T_{forget}$.
The definition of two different functions allows to specify whether an explored area needs to be crossed again multiple times or instead is needed to be visited only once, depending on the specific application. This differentiation is carried out marking explored areas that don't need to be visited again with the function $f_{F_i}(\mathbf{q})$, while areas required to be crossed multiple times are marked with the function $f_{V_j}(\mathbf{q}, t)$: time variance decreases

the intensity of the Gaussian function that models the area over time, allowing a specific region to be visited again in the future by the system. In this way, the robots will continue monitoring the environment even once the exploration is completed, keeping the gathered information up to date. Moreover, time variance is useful to avoid local minima which cause the robots to get stuck, making it difficult to achieve an entire exploration of the environment. Every robot stores two lists of Gaussian functions, time-varying and time-constant functions, whose combination generates the density distribution perceived by the robot and exploited to compute the control input. During the exploration process, the lists of Gaussian functions are shared among the robots' network. Each robot, during the motion, computes the Gaussian components and communicates their parameters to the network. Every robot's control input is continuously updated following the changes on the shared density distribution. The combination of the various different Gaussian components is computed as follows:

$$\phi(\mathbf{q}) = c + m + 1 - \sum_{i=1}^{c} f_{Fi}(\mathbf{q}) - \sum_{j=1}^{m} f_{Vj}(\mathbf{q}), \tag{3.5}$$

where c is the number of constant density functions f_{Fi} and m is the number of time-varying density functions f_{Vj}. The presented formulation generates a density distribution ϕ that has smaller values in areas already visited. The main objective for each robot is to add Gaussian components while exploring the environment decreasing the intensity of the density distribution around the recently visited areas in order to drive the robots' attention towards more interesting areas. The density distribution ϕ is then used, following the limited-range coverage control theory presented in Section 2.1.1 to determine the control input for each robot.

Every robot determines the addition of a new Gaussian component according to Algorithm 1. The main idea of the algorithm is to add new Gaussian functions in order to keep the robot in motion. A new component is added whenever a robot's position is sufficiently far from the mean point of existing components: the threshold is set equal to the standard deviation σ. The standard deviation σ can be seen geometrically as the width of the Gaussian function and is set to be half of the robot's sensing range r: $\sigma = r/2$. In a normal distribution, 95.45% of the samples are inside a 2σ radius from the center point, therefore the value $\sigma = r/2$ reflects the information that is collected by the robot in its sensing range.

It is worth noting that new components are not added in the actual position of the robot: in fact, this might lead to the definition of symmetric probability distributions, that would lead to have the centroid of the region in the same position as the robot, resulting in a null control input calculated in (2.10). To avoid this, the new components are centered in a previously occupied position: in particular, this delay Δ is calculated as the time needed for a robot to travel a distance of σ with maximum velocity v_{MAX}:

$$\Delta = \frac{\sigma}{v_{MAX}} \tag{3.6}$$

Moreover, if a robot gets stuck, a new component is added to introduce a perturbation in the probability distribution of the surrounding environment, moving the centroid in a

Algorithm 1: Exploration Algorithm

input : Robot's position $\mathbf{p}$
Time delay Δ
List of probability functions $f_V(\mathbf{q})$ and $f_F(\mathbf{q})$ with mean points $\{\overline{\mathbf{q}}_1, \ldots, \overline{\mathbf{q}}_{m+n}\}$ and variance σ^2
output: Updated probability distribution

```
1 begin
2     if (‖p − q̄_i‖) > σ, ∀i ∈ [0, (m + n)] then
3         if robot is moving then
4             Add a new time-varying Gaussian component with mean point
              corresponding to the position of p Δ seconds earlier
5         if robot has stopped then
6             Add a new time-varying Gaussian component with q̄ in the most
              recent robot position that is at a distance ≥ 0.75σ from p
7     if robot lies in an area that needs to be visited only once then
8         Add a time-constant Gaussian component with q̄ = p and σ = r/2.
```

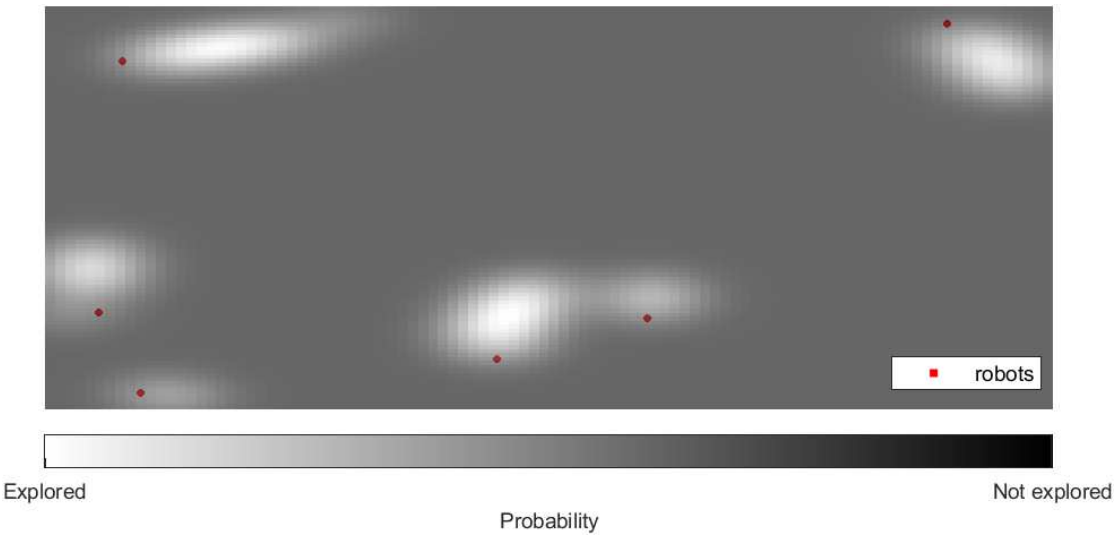


Figure 3.1: Probability distribution of the environment

different position.

The result of using Gaussian functions described above is the definition of a probability distribution indicating whether an area has already been explored or not: an example of such a function is depicted in Fig. 3.1.

3.1.4 Experimental Evaluation

Exploration Strategy

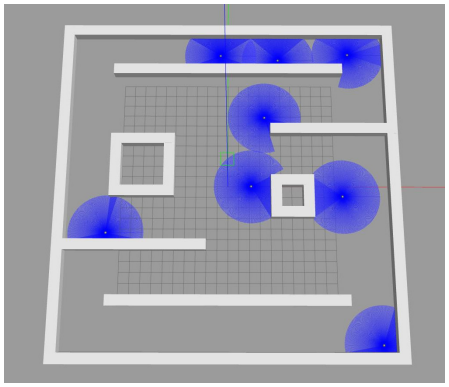
As previously introduced, the core of our exploration strategy lies in the design of the time-varying density function $\phi(\mathbf{q})$. This function is specifically formulated to represent the novelty or "unexploredness" of the environment over time. By incorporating this density into the limited-range centroid calculation (2.5) from Section 2.1.1, we effectively

guide the multi-robot system. The limited-range coverage control framework ensures that each agent is drawn towards the most novel areas within its immediate sensing capabilities. This synthesis of a dynamic density function with localized Voronoi partitioning compels the agents to continually seek out unvisited regions, thereby producing an emergent and distributed exploration behavior across the entire swarm.

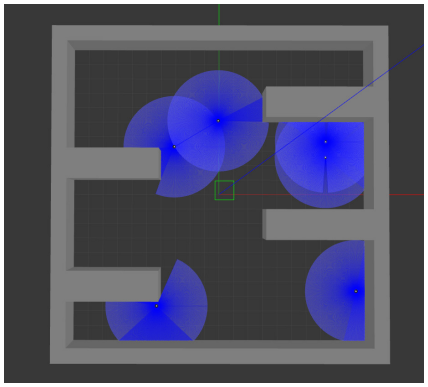
Implementation

The proposed algorithm has been implemented using ROS 2 [57], and has been validated by means of extensive simulations and real-world experiments. In particular, several simulation environments have been created using the Gazebo simulator, representing wide areas populated by a large number of robots. Two series of rigorous simulations have been conducted, the first one aiming for a statistical analysis of the results and the performance of the proposed control strategy, the other one to make a comparison with other strategies found in literature. As for the first series of simulations, the initial number of robots was randomly chosen, as well as their starting location in the environment. The simulations conducted include configurations with 2, 4, 6 and 8 robots with a sensing range $r_{sens} = 4m$ and a maximum linear velocity $v_{max} = 0.5\ m/s$. Moreover, various environment dimensions have been tested along with different environment configurations and different obstacles locations. In particular, we considered non-convex polygonal environments with an area of $200m^2$, $400m^2$, $900m^2$ and $2500m^2$. Two of the scenarios tested in the simulations are shown in Fig. 3.2: they differ in number of robots, environment dimension and complexity. For each experimental scenario, we ran 20 tests of 5 minutes of simulation: overall 320 tests were conducted. For each experiment we saved the robots locations over time, useful to compute the portion of the covered surface.
The same statistical approach was followed to make a comparison with other strategies. In this case, our solution was compared with the classic potential field navigation algorithm [43] and a random walk solution presented in [58], where robots drive with a constant straight motion, only changing their direction when an obstacle is detected. As before, the same robots were used in teams of 6, 8, 10 and 12 agents placed in random starting positions within the same $2500m^2$ environment already implemented. 20 simulations were run for every team configuration, for a total number of 80 simulations for each method.

We also conducted experiments on real robotic platforms exploiting the ROS 2 framework. The tests took place in a limited size environment with polygonal obstacles, using three Turtlebot3 Burger differential drive robots (see Fig. 3.3). The robots localization with respect to a global reference system was obtained through the Optitrack motion capture system. Every robot, following the control algorithm, shares the locally added Gaussian function among the robots. We conducted several tests randomly choosing the initial locations and orientations of the robots.
Both in simulations and real world experiments, information defining the Gaussian functions parameters were shared among the agents using a central storage unit, where each robot is able to both save and retrieve data. In addition, we considered non circulating



(a) 8 robots monitoring a $900m^2$ environment.



(b) 6 robots monitoring a $400m^2$ environment.

Figure 3.2: Two of the scenarios tested in simulation exploiting Gazebo environment. The two experiments differ in number of robots, environment dimension and complexity. The blue circles represent the sensing areas of the robots.

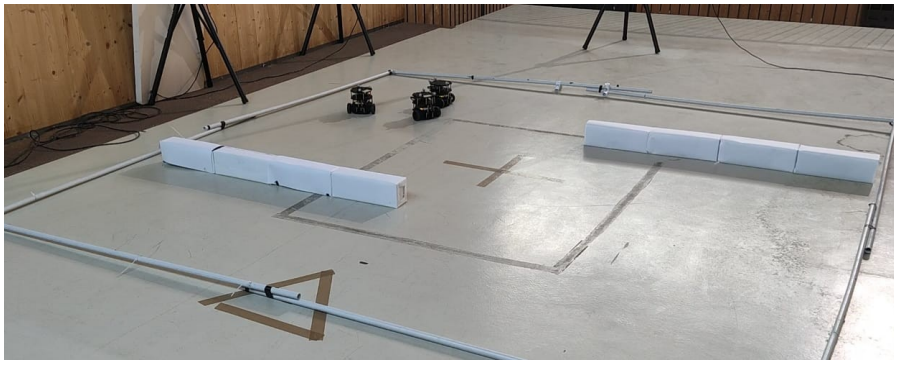


Figure 3.3: The experimental set-up of one of the experiments conducted with real robotic platforms. Three Turtlebot3 burger robots were tasked to explore an unknown non-convex area.

areas such as corners or dead end streets as places that need to be visited only once, marking them with time-constant Gaussian functions $f_{F_i}(\mathbf{q})$ defined in (3.2). Assuming that every robot is capable of detecting corners in the environment adopting a state-of-the-art strategy such as the one proposed in [59], it's possible for the agents to understand when a non circulating area is visited and consequently define a new time-constant Gaussian function $f_{F_i}(\mathbf{q})$.

Results and discussion

As previously mentioned, a large number of simulations were executed in order to have a rigorous statistical analysis of the results. We show how the performance of the control strategy depends on the starting number of robots and is not affected by the initial environment configuration or initial robots locations. In all the experiments and simulations, the multi-robot system was able to entirely explore the unknown non-convex environment. In particular, the percentage of the covered area during the exploration increases with the number of robots used.

Figure 3.4 focuses on analyzing the results from the simulations conducted on the environment shown in Fig. 3.2b, which has an area of $400m^2$. From the results in Fig. 3.4, we can conclude that 4 robots are sufficient to successfully explore the desired environment,

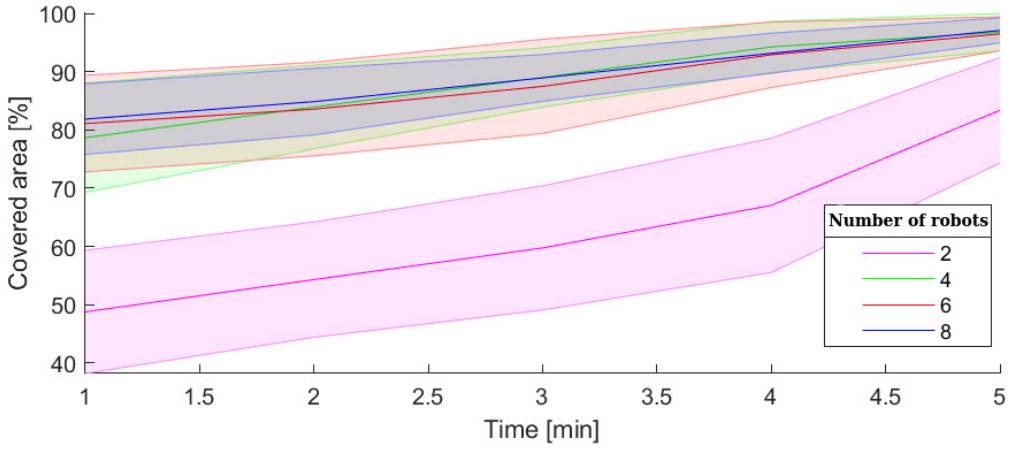


Figure 3.4: Explored area in $400m^2$ environment

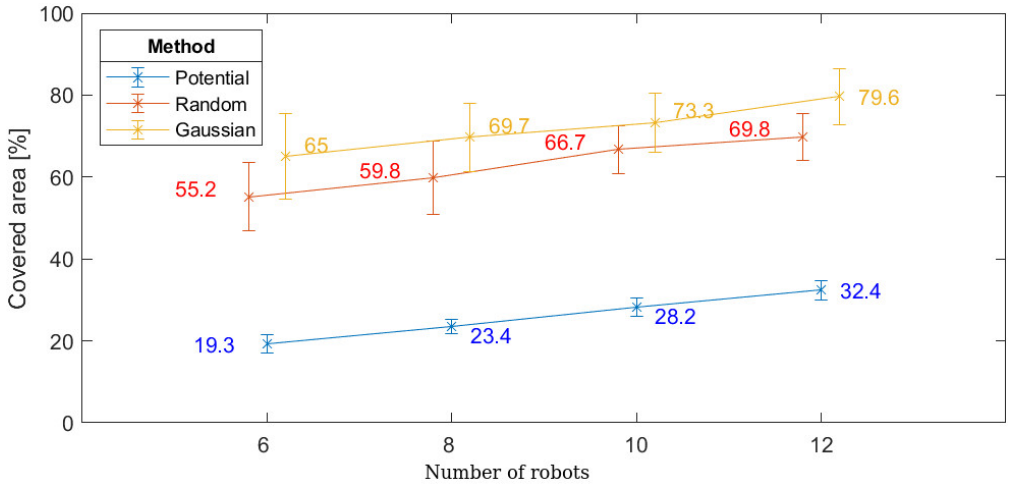


Figure 3.5: Comparison with other solutions

while slightly better performances are obtained with 6 and 8 robots.

Figure 3.5 instead shows the comparison between the results obtained with our solution and other strategies. The advantages with respect to the Potential Field navigation are clear: this solution had a poor performance because of the large environment used, which results in a great number of local minima where robots got stuck. Even the comparison with the Random Walk approach proves our solution to achieve a better performance: the multi-robot system was able to explore a larger region of the environment thanks to the ability to coordinate the agents in an optimal way.

Finally, real-world experiments have also proven very good exploration of the environment: in the example in Fig. 3.6, where 3 robots were employed, the covered area reached 90% of totality. This value is perfectly in line with expectations from simulated trials, because the total area is much smaller than the one in simulated environments, but less robots were employed.

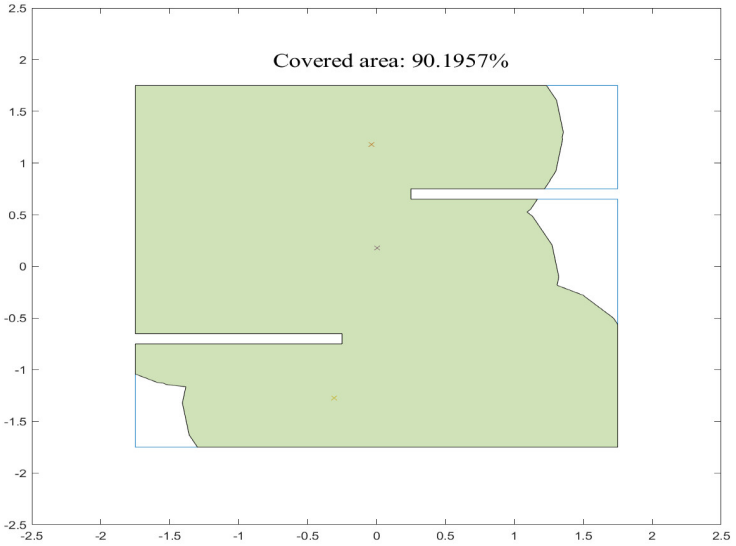


Figure 3.6: Explored area in field trials

3.1.5 Conclusions

In this Section we presented a distributed coverage based control to coordinate a group of autonomous robots with limited sensing capabilities with the aim to explore an unknown environment. The proposed control strategy exploits a density distribution to drive the multi-robot system towards areas of the environment to be explored. The density distribution is build from the combination of multiple Gaussian functions placed by the robots in the environment during the motion. We show how the full exploration of the unknown environment can be achieved without a discretization of the environment through graphs or grid cells differently from the majority of the studies found in literature. Moreover, the robots need to store and share among the network only the Gaussian parameters, which require a low communication volume and a low memory consumption. Several experiments and simulations were performed on a group of mobile robots to validate the performance of the proposed control method and make a comparison with other solutions found in literature. As a future work we aim to further investigate the proposed approach in comparison with frontier-based exploration methodologies.

3.2 Distributed Control for Human-Swarm Interaction In Non-Convex Environments using Gaussian Mixture Models

This Section presents a novel implementation of a human-swarm interface that allows humans to define an area with desired shape to be reached by a multi-robot system. Human-swarm interaction can be useful in order to exploit human intelligence and knowledge for the operation of swarm robots. The proposed work deals with limitations usually met when dealing with real-world implementation, e.g. limited sensing capabilities of the agents and hard conditions where communication is difficult or even completely denied. Gaussian Mixture Models are exploited in order to define an appropriate probability density function of the environment based on the area selected by a human operator. Then, velocity input for each robot is calculated in a distributed manner using Voronoi tessellation and Lloyd's algorithm. Finally, results of both virtual and real-world tests are presented, showing the final configuration reached by the multi-robot system in comparison with the desired region defined on the graphical interface.

3.2.1 Introduction

An interesting topic in the research field of multi-robot systems is the interaction between them and humans. [60] show that combining the abilities of humans and robots can lead to higher success rates for trivial operations. This benefit can be gained assigning the role of supervisor to the human, in order to exploit their superior intelligence and external point of view, while robots can concentrate on retrieving data from the environment and accomplish the mission. An interesting approach is proposed by [61], where a moving region of specific shape is defined for all the robots to stay inside. Potential energy functions are exploited to calculate the control input for each robot, and the results show great performances in shape control while maintaining a minimum distance between the agents. However, this solution requires a central computer with global knowledge, and only simple regular-shaped regions can be defined with a mathematical equation. Another solution is presented by [62], where the human is equipped with a haptic device acting as a controller robot, while swarm robots are placed into the environment performing a coverage task. A non-uniform density function represents areas with higher sensing interest, and a goal is defined by the operator manipulating the device. This approach exploits Voronoi tessellation and Lloyd's algorithm (see [33]) to obtain the control input and deals with the limited capabilities of the agents, but only allows to define a goal instead of a region with a desired shape, and the implementation is strictly related to the presence of a haptic device as controller robot. Swarm shaping is studied in one of the two approaches proposed by [63], where Gaussian Mixture Models are exploited to define the region of interest and robots are controlled in a distributed manner following the already mentioned Lloyd's algorithm. However, the agents are supposed to operate in a convex and obstacles-free environment, making this solution unlikely to be applied in real-world operations.

Contribution

Our solution evolves from the work of [63] with the aim of overcoming the limitation of assuming the environment to be convex and without obstacles inside. We make use of Gaussian Mixture Models to define a non-uniform density, in order to describe the desired region of interest to be reached by the agents. Then, a limited Voronoi diagram is calculated following the methodology presented by [37], which makes each robot capable of dealing with unknown and non-convex environments. In addition, it is worth noting that communication among robots is not necessary, while communication with a central unit is only needed at the beginning of the mission, where the parameters defining the Gaussian Mixture Model have to be transmitted to the agents. In this way, the human operator can be seen as an external supervisor, possibly with a global view of the environment, while on-field operations are demanded to swarm robots, which can operate even in hard or dangerous conditions. We extensively tested our control strategy in simulations and also with few real-world experiments, where the final configuration of the multi-robot system is compared with the desired shape of the defined region of interest.

3.2.2 Problem Description

Consider a multi-robot system composed by n robots moving in two dimensions, controlled in order to reach a specific area of the environment. We assume each robot to be modeled as a single integrator system, whose position $\mathbf{p}_i \in \mathbb{R}^2$ evolves according to $\dot{\mathbf{p}}_i = \mathbf{u}_i$, where $\mathbf{u}_i \in \mathbb{R}^2$ is the control input, $\forall i = 1, \dots, n$.
We consider the following assumptions:

- *Localization*: each robot is able to localize itself with respect to a global reference frame, which is common for every robot within the team.
- *Limited sensing capabilities*: each robot is able to detect and measure the position of every object (including other robots and environmental boundaries) inside its limited sensing range, defined as a circle with radius $r_{sens} \in \mathbb{R}_{\geq 0}$.

Based on these assumptions, we can formalize the problem addressed in this section as follows:

Problem. *Implement a human-swarm interface that allows a user to define a specific area of the environment with a desired shape to be reached by a multi- robot system.*

Proposed Architecture of the Human-Swarm Interface

To solve the mentioned problem, we propose a human-friendly methodology, whose aim is to autonomously interact with swarm robots, while the human operator is only required to draw the desired region to be reached. This solution can be briefly described as a 5-step procedure:

1. The human operator draws the region of interest on a graphical interface.
2. A suitable Gaussian Mixture Model is calculated to fit the desired shape.

3. Parameters defining the Gaussian Mixture Model are communicated to the agents.

4. The probability density of the environment is calculated by each robot.

5. The control action is calculated in a distributed manner.

3.2.3 GMM-based Probability Density Function

Here we analyze how the geometrical shape drawn by the operator is converted into a probability density function of the environment, and how this function is exploited to highlight the desired region of interest. Before going into details, it is necessary to introduce Gaussian Mixture Models, as they will play a key role in our implementation. A Gaussian Mixture Model (GMM) is a multivariate distribution that consists of multivariate Gaussian distribution components, each one defined by a mean point $\boldsymbol{\mu}_i \in \mathbb{R}^2$ and a covariance matrix $\boldsymbol{\Sigma}_i$ (see [64]). Given k components, with $k \in \mathbb{R}_{>0}$, the overall model is obtained as the result of their combination following a specific mixture proportion, defined by a vector of weighting factors $\boldsymbol{\omega} = [\omega_1, \ldots, \omega_k]^T \in \mathbb{R}^k_{>0}$ related to each component, with $\sum_{i=1}^{k} \omega_i = 1$.

Given a polygon $S \subset Q$ drawn by the user on a graphical interface, representing the area of interest in the environment, a GMM fitting the desired shape is estimated with a *Maximum Likelihood* method as described by [65]. This method uses an *Expectation-Maximization* algorithm to iteratively find the optimal set of parameters $(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\omega})$.
Once the Mixture Model has been defined, we can define the probability density function as the sum of the contributions brought by the single components. According to the definition provided by [64], the contribution of a single d-dimensional component (in our case, $d = 2$), is calculated as

$$\phi_i(\mathbf{q}, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \frac{1}{\sqrt{|\boldsymbol{\Sigma}_i|(2\pi)^d}} \exp\left(-\frac{1}{2}(\mathbf{q} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1}(\mathbf{q} - \boldsymbol{\mu}_i)\right). \tag{3.7}$$

From the above equation, the contribution of a single Gaussian component to the global probability density is obtained from the covariance matrix $\boldsymbol{\Sigma}$, defining the spatial distribution around the mean point $\boldsymbol{\mu}$, specifically calculated to fit the drawn polygon. Finally, the overall probability function is obtained as the sum of each component weighted according to the mixture proportion:

$$\Phi(\mathbf{q}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{i=1}^{k} \omega_i \, \phi_i \left(\mathbf{q}, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right). \tag{3.8}$$

This probability function defines a non-uniform density of the environment, assigning to each point $\mathbf{q}$ high probability values if placed inside the region of interest drawn on the graphical interface. An example can be seen in Figure 3.7, where the region of interest is assigned much higher probability values than the rest of the environment. In the following, we describe how to take advantage of this probability density function to calculate a proper control action to fulfill the goal of the mission.

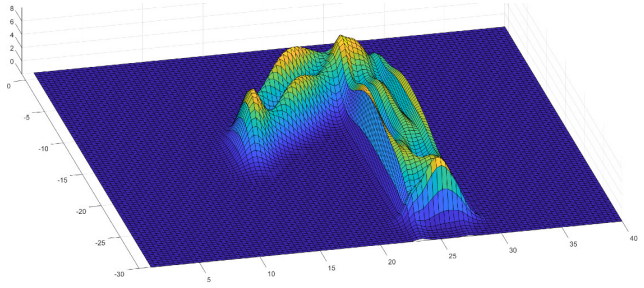


Figure 3.7: Probability density defined from the GMM

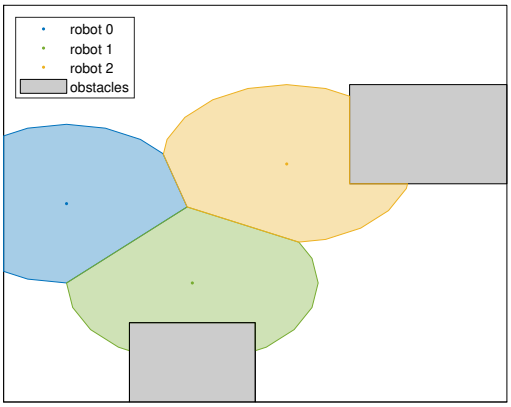


Figure 3.8: Limited Voronoi partitioning in the presence of obstacles.

3.2.4 Distributed Control Algorithm

As previously stated in Section 3.2.1, the aim of the proposed work is to tackle limitations usually met in real-world applications. More in details, the proposed solution is expected to work as a distributed methodology to operate in unknown non-convex environments where communication among the agents is denied. Based on the assumptions in Section 3.2.2, every robot is capable of detecting and localizing obstacles, environmental borders and other robots falling inside its sensing range r_{sens}. For the sake of simplicity, we can denote the surface occupied by obstacles as

$$\mathcal{O} = \bigcup_{i=1}^{m} O_i \tag{3.9}$$

where $m \in \mathbb{R}_{\geq 0}$ is the number of obstacles and $\{O_1, \ldots, O_m\} \subset \mathbb{R}^2$ is the set of convex regions defining the surface occupied by each of them. Thus, we can define a new region $\widetilde{\overline{B}}(\mathbf{p}_i, r_{sens})$ as the difference between the area $\overline{B}(\mathbf{p}_i, r_{sens})$ covered by the agent's sensing range and the surface $\mathcal{O}$ occupied by obstacles:

$$\widetilde{\overline{B}}(\mathbf{p}_i, r_{sens}) = \overline{B}(\mathbf{p}_i, r_{sens}) - \mathcal{O}. \tag{3.10}$$

From the robot's perspective, this region can be reconstructed using onboard sensors, as it corresponds to the obstacle-free portion of the environment within its sensing range. Coordination among the agents is performed with a limited-range Voronoi partitioning as

Algorithm 2: Best Fitting GMM Calculation Algorithm

```
input  : S, k, δ
output: (μ, Σ, ω)
begin
    Discretize area of S.
    i = 1.
    while i ≤ k do
        Get GMM with i components using Expectation-Maximization algorithm.
        Calculate BIC(i).
        if BIC(i) − BIC(i − 1) ≥ −δ then
            break
        else
            i++
```

described in Section 2.1.1. An example of the obtained partition is depicted in Fig. 3.8.

Subsequently, the centroid of each Voronoi cell $\mathbf{C}_{V_i}$ is calculated similarly to (2.5), taking into account the probability density function $\Phi(\mathbf{q}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ defined in (3.8):

$$\mathbf{C}_{V_i} = \frac{\int_{V_i} \mathbf{q}\, \Phi(\mathbf{q}, \boldsymbol{\mu}, \boldsymbol{\Sigma})\, d\mathbf{q}}{\int_{V_i} \Phi(\mathbf{q}, \boldsymbol{\mu}, \boldsymbol{\Sigma})\, d\mathbf{q}}. \tag{3.11}$$

Finally, the desired control input for the i-th robot is calculated proportionally to the distance from the centroid of its cell as in (2.10). Such a control law, together with the probability density function defined in (3.8), leads agents towards the region of interest, which is assigned with a higher probability density. It is worth noting that, according to the definition of limited-range Voronoi partition in (2.7) and centroid (3.11), no global knowledge is required by the agents to calculate their control action, since both the limited Voronoi partitioning and the centroid can be computed in a distributed manner. Moreover, communication among swarm robots is also not required. As a matter of fact, the only data needed by each robot to calculate its control input are its own position, the neighbors' location and the probability density of the environment, as was extensively tested by [66].

However, it must be taken into account that, in certain circumstances, the limited Voronoi region could result in a non-convex region, thus its centroid could be placed outside. This means that the robot could be driven towards an obstacle and crash into it. Therefore, the proposed control strategy does *not* guarantee obstacle avoidance, so a further implementation could be needed to safely operate in real-world applications.

In the following, an experimental evaluation of the developed solution is presented, describing how this has been implemented and showing the final results obtained in different scenarios.

3.2.5 Experimental Evaluation

In this section we will describe how the proposed solution was implemented to be tested on both virtual and real mobile platforms. First of all, the implementation of the graphical

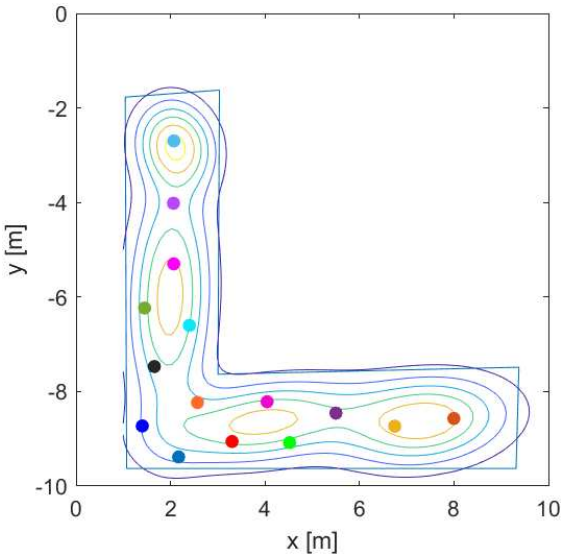


Figure 3.9: Simulation with 15 networking nodes acting as robots

interface is presented, in order to explain the transition from a drawn polygon to a specific GMM. The definition of the region of interest through the graphical interface is equally applied to both simulations and real-world tests. Subsequently, experimental evaluation is presented both in simulations and real-world environments, showing the behavior of the controlled system to reach the region of interest defined through the graphical interface.

Implementation of the Graphical Interface

The implementation is carried out using MATLAB, and a blank canva is presented to the operator, who is expected to draw a polygon $S \subset \mathbb{R}^2$ moving the mouse cursor on it and to decide the maximum number of components $k \in \mathbb{R}_{>0}$ to be generated. Subsequently, the best fitting Gaussian Mixture Model is calculated following Algorithm 2: a discretization of the area is performed and a suitable Mixture Model is iteratively calculated with an increasing number of components according to the *Expectation-Maximization* algorithm proposed by [65]. Then, the Bayesian Information Criterion (BIC) is calculated according to [67]: the lower is the BIC, the better the model is fitting the region. The process continues until convergence is achieved, or the maximum number of components k is reached. It is interesting to note that the proposed algorithm iteratively searches for the optimal number of components fitting the desired shape, in order to have a low number of parameters to be stored and computationally efficient calculations to be carried out by the swarm robots during the mission. The optimal set of parameters $(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\omega})$ defining the GMM calculated with Algorithm 2 is stored to be communicated to the robotic agents.

Virtual Tests

MATLAB has also been used for a first implementation of the algorithm, where swarm robots were approximated to networking nodes moving in the environment as single in-

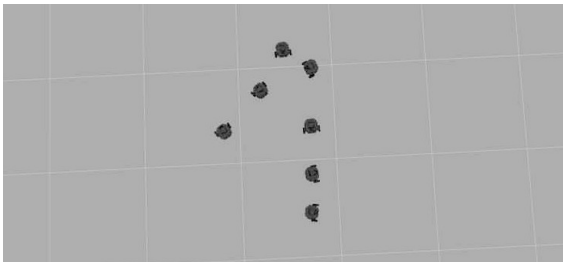


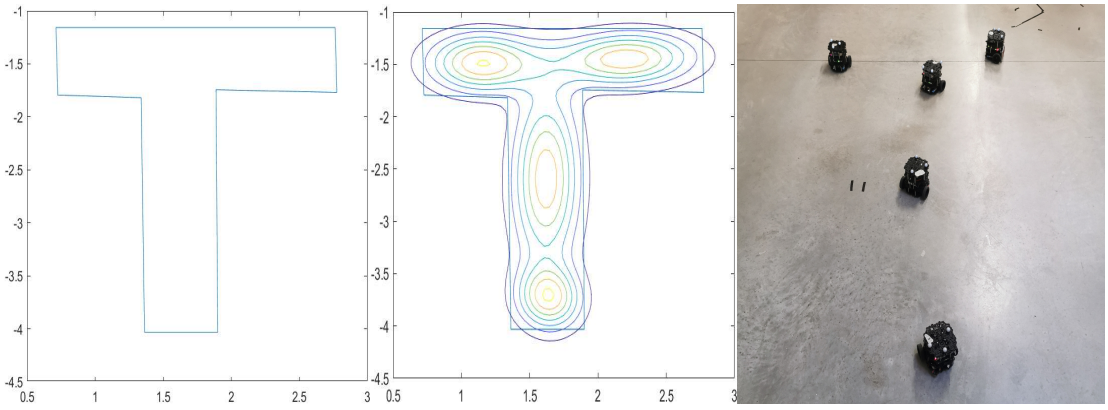
Figure 3.10: Simulation within a virtual environment

Figure 3.11: Real-world implementation

tegrator systems. This first set of tests has shown that the multi-robot system behaved as expected, as we can see from Fig. 3.9 where the final configuration of the network perfectly fitted the desired shape of the region of interest. Moreover, the low computational effort required for those simulations allowed to employ a large number of robots, thanks to the absence of a physical engine modeling interactions of robots with each other and with the environment.
Subsequently, trials were performed employing mobile platforms and using ROS 2 as a control architecture. The behavior of the controlled multi-robot system was extensively tested with simulations carried out within the Gazebo physical engine employing virtual models of the TurtleBot3 Burger robot. Robots were controlled in a distributed manner, and the set of parameters defining the Gaussian Mixture Model was communicated to each one at the beginning of the mission using a custom ROS message, directly sent over the ROS network from the MATLAB implementation of the graphical user interface. An example of final configuration reached by the multi-robot system can be seen in Fig. 3.10, where the probability density shown in Fig. 3.7 was exploited to define the region of interest.

3.2.6 Real-world Experiments

A further step in testing the developed solution has been made with real-world trials, carried out employing the real counterpart of the TurtleBot3 Burger platforms used for simulations. As for virtual trials, the set of parameters defining the GMM was communicated to the agents only at the beginning of the mission over a ROS network. This kind



(a) **(b)** **(c)**

Figure 3.12: Complete workflow of the proposed methodology: (a) the region of interest is drawn on a graphical interface, (b) a GMM is calculated fitting the desired shape, (c) the multi-robot system is actuated and reaches the desired configuration

of tests were performed with randomly chosen starting positions of the agents within a $4.5 \times 3.5\ m^2$ environment, and the Optitrack motion capture system was exploited to obtain localization of the robots with respect to a global reference frame. It is important to note that communication with a central unit has been exploited by each robot only to gain information on its global position and the relative position of its neighbors, in order to emulate localization capabilities as described in the assumptions in Section 3.2.2. Initially, results were investigated with the same probability density that was shown in Fig. 3.7, in order to have a comparison with the behavior obtained in simulations where the final configuration in Fig. 3.10 was reached. Swarm robots ended up reaching the desired region of interest as shown in Fig. 3.11, demonstrating that the presented approach ensures good performance in swarm shaping. Because of the limited size of the area at our disposal for the execution of on-field tests, only simple scenarios were set up and no obstacles were placed in the environment. Several heterogeneous regions of interest were tried out, and the multi-robot system always displayed a performing behavior in reaching a final configuration fitting the desired shape. A further example is shown in Fig. 3.12, where the entire workflow of the presented methodology is displayed, from the definition of the region of interest, to the generation of a Gaussian Mixture Model fitting the desired shape, and finally to the actuation of robotic agents. As one would expect, the higher is the number of robots employed in the mission, the better they will fit the region of interest.

Finally, we conducted a further set of virtual trials, focusing on the behavior of the controlled multi-robot system within a non-convex environment. As we mentioned in the above sections, the ability to deal with obstacles and cluttered environments is a fundamental feature for robot swarms to be employed in real-world operations. Several virtual environments were prepared with randomly positioned obstacles of different regular shapes, in order to lower the chance of generating non-convex limited Voronoi regions that could make robots crash. Results have shown that the multi-robot system behaves as expected, avoiding obstacles while navigating and reaching the final desired shape only covering areas within the region of interest that do not contain obstacles. Examples are shown in Fig. 3.13, where swarm robots sense the presence of obstacles preventing them

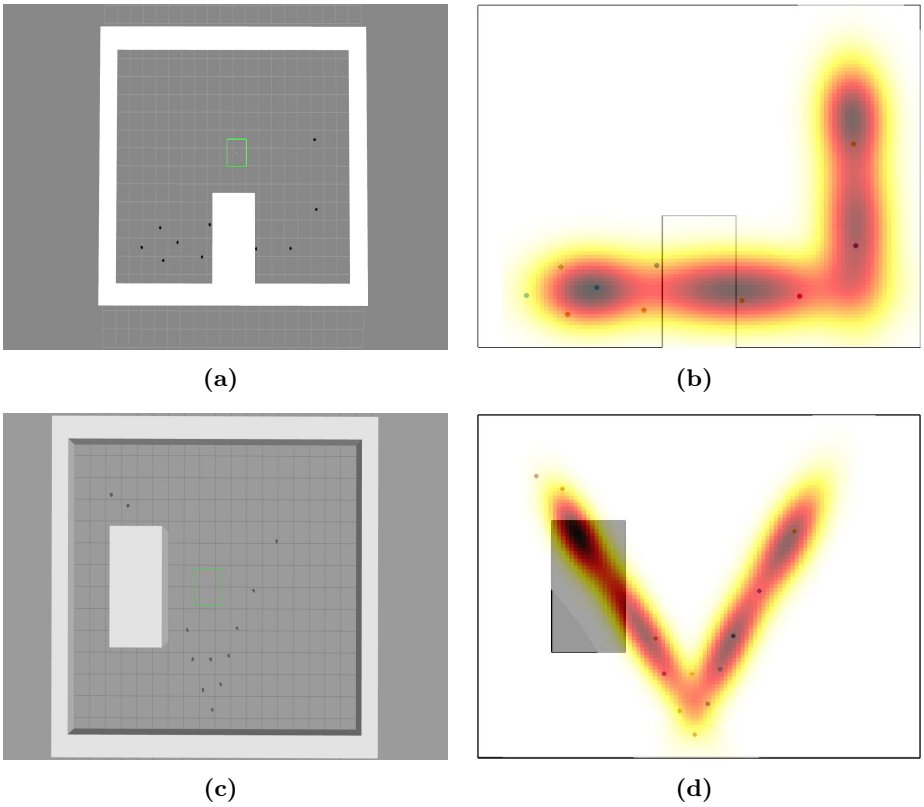


Figure 3.13: Multi-robot system dealing with obstacles in the environment. (a), (c): The final configuration is reached and the obstacle is avoided. (b), (d): Comparison between the final configuration and the probability density representing the region of interest.

to reach certain areas, so they rearrange themselves to cover obstacles-free portions of the region of interest. The area covered by the multi-robot system has been evaluated and compared with the overall area enclosed by the region of interest as shown in Fig. 3.14, resulting in a complete sensing of the desired region.
In conclusion, we can say that those trials validated the proposed methodology, so it can be stated that the presented control architecture ensures great performances in covering a specific area with desired shape using a multi-robot system, even when dealing with non-convex environments or obstacles.

3.2.7 Conclusion

In this section, a human-swarm interaction methodology was presented, whose aim is to allow a human operator to define a region of interest with a desired geometrical shape to be explored by the agents in the environment. With this solution, the operator plays the role of a supervisor with a global knowledge of the environment, while on-field execution of the mission is demanded to robotic agents. Hence, safety conditions are guaranteed to the human being, who is not required for an in-person visit in a possibly dangerous environment. This strategy exploits the definition of Gaussian Mixture Models from a polygon drawn on a graphical interface to define a non-uniform density function, where higher importance is given to the region of interest, together with a limited Voronoi partitioning of the environment. This approach sets up a distributed strategy in order to deal with limitations usually met in real-world implementations, e.g. limited sensing

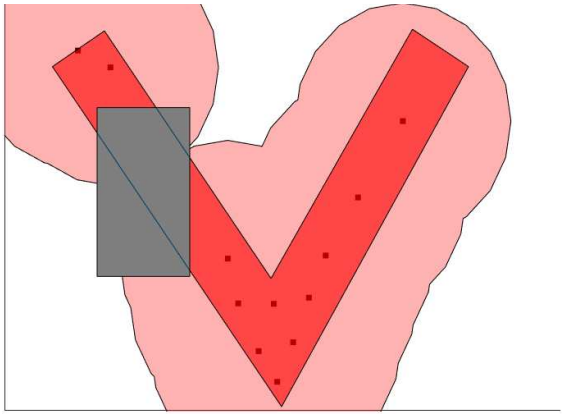


Figure 3.14: Comparison between total area of the desired region and area covered by swarm robots with 3.5 m sensing range

capabilities of the agents and communication impossibility. It is important to note that the definition in (3.10) only considers the difference between free and occupied surfaces, but it does not take into account blind spots generated using a sensor on a real robot. In particular, the real field of view of a ground vehicle will be slightly different from the one calculated with this assumption, while no differences are met when employing aerial vehicles observing from above. However, the mentioned assumption does not prevent from using this methodology in real-world applications, since the absence of blind spots in the sensing range can only positively affect the overall behavior of the controlled system. Several tests have been made both in simulation and on real mobile platforms, and the results show that the multi-robot system behaves as expected and reaches a final configuration fitting the desired shape of the region of interest. At the moment, only regular environments have been tested, with simple-shaped obstacles placed inside of them only in simulations, therefore robots have been able to avoid collisions.
Future work will extend on-field trials to more complex scenarios, where agents are required to operate in larger areas with obstacles placed into the environment. Furthermore, a necessary step could be the integration of an inner control layer to always guarantee obstacle avoidance, in order to allow for a real-world application of the proposed control strategy. An interesting approach could exploit Control Barrier Functions to define a minimum distance to be maintained between robots and obstacles (see [68]), calculating the optimal control action compliant with this constraint from the desired one obtained with the methodology presented in this work. Finally, another interesting enhancement of the proposed architecture could exploit a previously taken picture of the operation area, allowing its integration into the graphical interface. In this way, the operator will be able to interactively draw the region of interest in order to precisely fit a specific area of the environment. The image could also be elaborated with the aim of finding specific features, from which the region of interest can be automatically generated. An example of a scenario where this solution could be exploited is a search and rescue mission taken in a urban environment with a group of UAVs, where buildings must be treated as obstacles and a specific area must be reached by the agents, where a certain target feature is located.

3.3 A Mixed Reality Interface for Human-Swarm Interaction

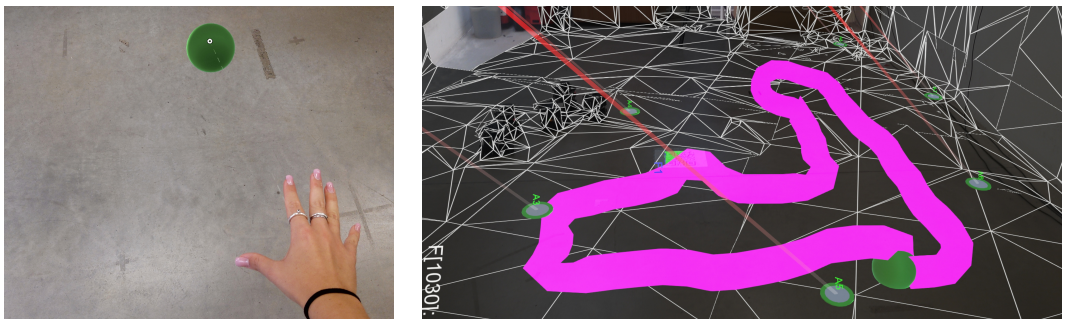
The increasing deployment of multi-robot systems underscores their potential across diverse research and applied domains. While the previous Section introduced a graphical-interface-based human-swarm interface for specifying regions of interest, full autonomy remains challenging in many real-world scenarios, and human input continues to play a crucial role. In this section, human-robot interaction is further advanced by enabling the operator to supervise the multi-robot system directly in Mixed Reality, combining human spatial reasoning with robotic sensing and actuation. By assigning the human operator the role of supervisor, robots focus on environmental data retrieval, enhancing safety and task execution efficiency. This work presents a Mixed Reality solution utilizing the Microsoft HoloLens 2 headset, allowing a human operator to designate specific areas for a multi-robot team to reach and cover within the physical environment.

3.3.1 Introduction

Even though research has been moving towards increasing robots autonomy, there are still many situations where robots are unable to operate fully autonomously. An interesting approach to overcome existing limitations could be to intelligently exploit human-robot interaction (HRI) in order to combine their capabilities, instead of aiming for a complete automation of the task. As pointed out in [60], a human operator can be assigned with the role of supervisor, exploiting their intelligence and their possibly external point of view, while robots can concentrate on retrieving information from the environment and fulfill the operation. In addition, this strategy enhances safety for the human being, demanding the possibly dangerous on-field execution of the task to the robots. Recently developed Augmented Reality (AR) or Mixed Reality (MR) interfaces offer a promising approach for HRI, thanks to the possibility to define a shared environment where humans and robots can interact and collaborate. MR devices such as Microsoft HoloLens, in particular, offer an immersive experience to the user, allowing different types of interaction, not only limited to a touch input on a screen.
In this section, we develop a MR solution to interact with a MRS. In our implementation, a human operator is equipped with a Microsoft HoloLens 2 headset, enabling them to designate a specific area with a desired shape to be reached and covered by the team.

Assumptions

Consider a MRS composed by n robots moving in two dimensions, controlled in order to reach a specific area of the environment. We assume each robot to be modeled as a single integrator system, whose position $\mathbf{p}_i \in \mathbb{R}^2$ evolves according to $\dot{\mathbf{p}}_i = \mathbf{u}_i$, where $\mathbf{u}_i \in \mathbb{R}^2$ is the control input, $\forall i = 1, \dots, n$.
We assume robots are able to localize themselves and other robots detected inside their circular sensing region, defined by a radius $R_s \in \mathbb{R}^2$. Communication among them is not



(a) (b)

Figure 3.15: (a) Hand gestures for cursor manipulation. (b) Region of interest drawn in the MR environment.

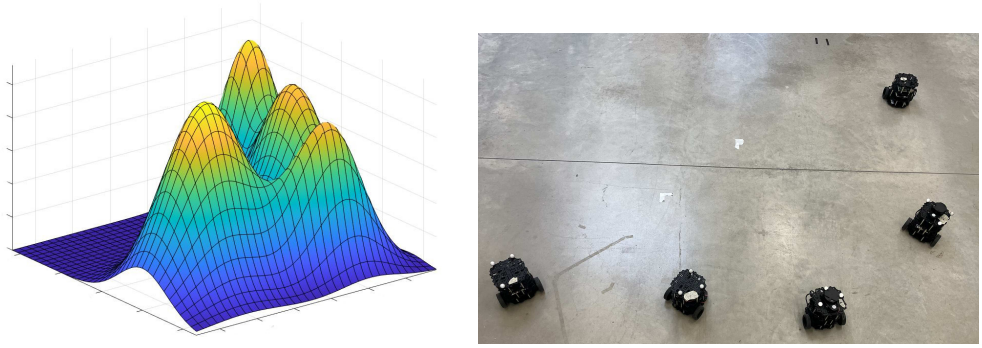
allowed.

3.3.2 Control Architecture

Here, we present the tools employed to build up the MR interface for HRI. As previously stated, the goal of our implementation is to create a shared environment for a MRS and a human operator, in order to combine their capabilities and fulfill the task. In particular, the goal of the presented solution is to allow a human to define a desired region of interest inside the environment, and move the robots into this area in order to retrieve information, exploiting human intelligence for the high-level planning of the task while demanding the on-field execution to the robotic team.
In our work, we make use of Microsoft HoloLens 2 headset featuring a see-through display for Mixed Reality. A virtual scene has been created with Unity, featuring a virtual sphere to be used as a cursor by the operator, who can manipulate the cursor with specific hand gestures, as shown in Fig. 3.15a. QR codes have been placed in known locations in the real environment, to be detected by the HoloLens cameras in order to align the reference frame of the virtual scene with the real-world reference frame. While moving the sphere around, a trail is drawn to show the area that is being selected (see Fig. 3.15b), and the coordinates of the cursor are continuously sent to an external computer to keep track of the movement. Finally, once the elaboration system has fitted a GMM to the region that the human operator has defined, it communicates the GMM parameters $(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\omega})$ to the robots, making them move in accordance with the coverage control algorithm outlined in Section 2.1. Robots and the external computer exploit ROS 2 to run the control software and to establish communication with each other.

3.3.3 Experimental Evaluation

Finally, we demonstrate the execution of an experiment where a human operator must identify a region in a real-world environment that has to be monitored by robots, with the aim of evaluating the effectiveness of the proposed architecture. In order to monitor the robots using a motion capture system, the user must wear the HoloLens 2 headset and scan a QR code to align the inertial reference frame with the global one, which



(a) **(b)**

Figure 3.16: (a) Probability density associated to the drawn region of interest. (b) Final configuration of the MRS.

is specified in the real world. Then, they can grab the virtual sphere and encircle the desired area to be reached, resulting in the final shape, as shown in Fig. 3.15b. Then, the external computer fits a GMM to the selected region, resulting in the probability distribution shown in Fig. 3.16a, where the higher attractive effect of the area encircled by the human operator is clearly visible. Finally, robots are controlled as described in Section 2.1. A sensing range $R_s = 2$ m is considered for each robot, and a random starting position is chosen for each of them. As we can see from Fig. 3.16b, the final configuration assumed by the robotic team fits the desired area defined by the operator, and robots optimally spread in order to maximize the covered surface of the region of interest.

3.3.4 Conclusions

In this work we proposed a MR solution to interact with a MRS. Our solution makes use of Microsoft HoloLens 2 headset to encircle a region of the environment to be reached by the robots. We performed a qualitative analysis of results employing real mobile robots, showing that the MRS successfully reaches the desired area. In future works, we intend to gather quantitative data to examine the effectiveness of our solution. Additionally, we plan to carry out user studies to evaluate the architecture's usability and assess the degree to which people feel comfortable engaging with robots in a shared environment.

3.4 HMPCC: Human-Aware Model Predictive Coverage Control

We address the problem of coordinating a team of robots to cover an unknown environment while ensuring safe operation and avoiding collisions with non-cooperative agents, such as humans present in the workspace. Traditional coverage strategies often rely on simplified assumptions, such as known or convex environments and static density functions, and struggle to adapt to real-world scenarios, particularly when transitioning from supervised human-robot interaction (as in the previous Mixed Reality framework) to treating humans as dynamic obstacles.

In this work, we propose a human-aware coverage framework based on Model Predictive Control (MPC), namely HMPCC, where human motion predictions are integrated into the planning process. By anticipating human trajectories within the MPC horizon, robots can proactively coordinate their actions and adapt to dynamic conditions, extending the GMM-based environment modeling from prior supervisory approaches to now account for real-time human presence. Team members operate in a fully decentralized manner, without relying on explicit communication, an essential feature in hostile or communication-limited scenarios.

Our results show that human trajectory forecasting enables more efficient and adaptive coverage, improving coordination between human and robotic agents while preserving the Voronoi-based coverage objectives from previous sections.

3.4.1 Introduction

Tasks such as search and rescue, environmental monitoring, exploration, surveillance, and inspection often occur in hostile and critical scenarios, where teams of robots are deployed in spaces potentially shared with humans. In such scenarios, humans behavior can be unpredictable in both their decision-making and movements, increasing the uncertainty and complexity of the environment. Therefore, it is essential to coordinate the motion of robots to account for the presence and behavior of nearby humans while exploring an unknown environment.

A common solution to this coordination problem is area coverage. Traditional methods like Voronoi tessellation [33] and Lloyd's algorithm [36] partition the space and position robots at the centroids of their cells as shown in Section 2.1, often guided by a density function representing areas of interest. However, these approaches typically assume convex environments, known density distributions, and simple robot dynamics.

One of our key contributions is to overcome these limitations and extend coverage techniques to more realistic and complex scenarios. The challenge intensifies when robots must operate alongside humans. While prior work often treats human presence reactively or with static models, we integrate human trajectory prediction (HTP) to proactively and safely coordinate robot motion, ensuring adaptive and reliable area coverage. HTP has been studied for decades and is increasingly relevant in domains such as autonomous

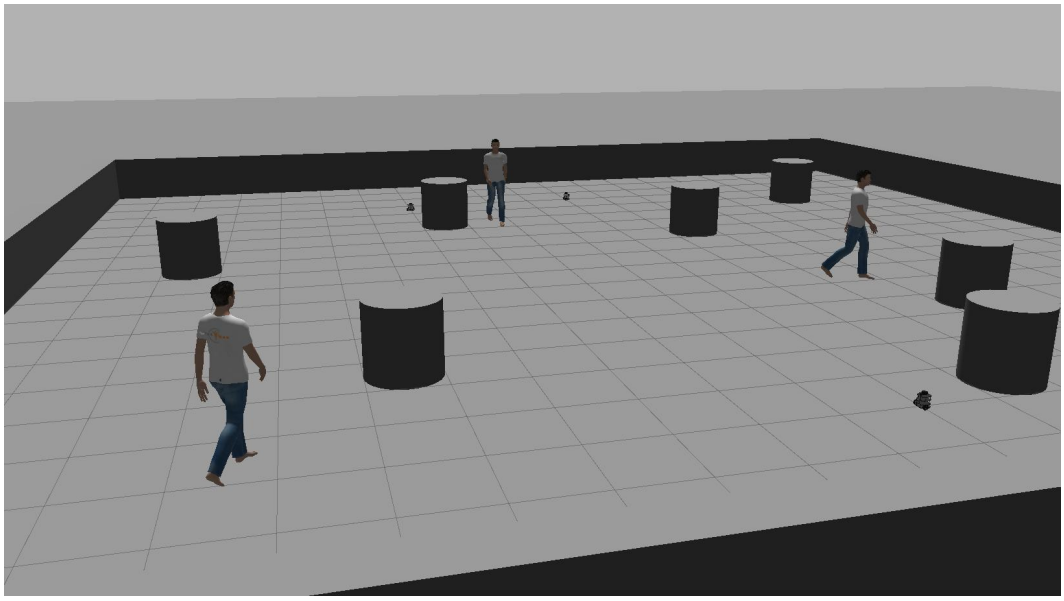


Figure 3.17: Gazebo simulation with humans and TurtleBot3 robots. HMPCC integrates human motion prediction into trajectory planning, successfully guiding robots with arbitrary dynamics in obstacle-rich environments.

driving, social robotics, and surveillance. According to [69], HTP methods are broadly classified into modeling-based and context-based approaches. Modeling-based methods use physical principles and domain knowledge, including crowd models like the Social Force Model (SFM [70]) and velocity-based techniques such as Reciprocal Velocity Obstacles (RVO [71]) and Optimal Reciprocal Collision Avoidance (ORCA [72]). Context-based methods rely on internal and external stimuli, often using data-driven deep learning models like Long Short-Term Memory networks (e.g., SocialLSTM [73]), Generative Adversarial Networks (e.g., SocialGAN [74]), and Conditional Variational Autoencoders (e.g., Trajectron++ [75]). FlowChain [76], a flow-based model using continuously-indexed flows, is another notable example. In this work, we address optimal coverage of an unknown environment by a team of robots that share the environment with humans. Exploration is guided by a Gaussian-mixture density map, while all agents operate in a decentralized manner and without communication, reflecting communication-denied or hostile settings. Robot motion is planned with an MPC framework that embeds human trajectory prediction, namely HMPCC. Simulations as in Fig. 3.17 prove that this method ensures safe and effective coverage even when human intentions are unknown.

Contributions

The main contributions of this work are twofold and can be summarized as follows:

- an MPC-based framework for coverage control that accounts for generic robot's dynamics and non-convexity of the environment; and
- the integration of human trajectory prediction into the MPC formulation to account for human motion when controlling the robots.

3.4.2 Related Work

Several works have extended coverage control to more realistic scenarios, such as non-convex environments and general robot dynamics. In [77], a decentralized Voronoi coverage approach is proposed for polygonal environments using geodesic-based tessellation and the TangentBug [78] algorithm for local planning. Battacharya et al. [79] introduce an entropy-weighted Voronoi partition, guiding robots toward high-entropy regions. Potential-based repulsive forces have also been used for collision avoidance [80]. Instead, dynamics non-linearity has been recently addressed in [81] for unicycle robots. MPC has also emerged as an important technique to handle constraints and non-linear dynamics. In [82], an MPC-based coverage algorithm achieves convergence to a centroidal Voronoi configuration, assuming known density functions. Rickenbach et al. [83] further extend MPC to unknown environments by incorporating active density learning.

Despite these improvements over traditional methods, explicitly accounting for humans in the coverage problem remains a key challenge that is still largely unexplored. Unlike robots, whose behavior can be governed by control policies, humans are autonomous agents who make independent decisions. This makes their behavior significantly more difficult to model, highlighting the need to consider human presence as part of the coverage problem.

In the literature, many works have investigated how to account for human presence in coverage and navigation tasks. In [84], the authors studied different collision scenarios, including a heterogeneous crowd of both robots and humans, who are modeled with a holonomic kinematic model and distributed according to the Voronoi coverage controller. Claes et al. [85] addressed the problem of collision avoidance in a shared space by employing various cost maps and Monte Carlo sampling, incorporating different cost factors to account for both humans and robots. The task of multi-robot coordination in environments shared with humans is tackled by Talebpour et al. [86]. Humans are modeled as dynamic obstacles with orientation-based Gaussian cost maps representing their personal space. These social costs are used in bid evaluations, and robots actively request team collaboration when humans block their paths.

3.4.3 Problem Formulation

We assume robots obey the following motion law:

$$\dot{\mathbf{x}}_i = f(\mathbf{x}_i) + g(\mathbf{x}_i)\mathbf{u}_i \tag{3.12}$$

where $\mathbf{x}_i \in \mathbb{R}^n$ is the state of the i-th robot, which may include position, velocity, orientation, or a combination of them, $\mathbf{u}_i \in U \subset \mathbb{R}^m$ is its control input, U indicates the set of admissible inputs, and $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are Lipschitz continuous functions.

Robots operate in a 2D environment populated with $N_o \in \mathbb{N}$ obstacles and $N_h \in \mathbb{N}$ humans. The likelihood of events of interest occurring in the environment is modeled

by the function ϕ as a Gaussian Mixture Model. We assume that a prediction model is available to estimate the future trajectory of each human over a discrete time horizon $T \in \mathbb{N}$, along with the associated uncertainty. Specifically, at the current time $t = t_0$, the predicted position $\boldsymbol{\xi}_i$ of the i-th human at future time step $t_0 + k$ is modeled as a Gaussian distribution:

$$\boldsymbol{\xi}_i(t_0 + k \mid t_0) = \mathcal{N}(\boldsymbol{\mu}_i^k, \boldsymbol{\Sigma}_i^k), \tag{3.13}$$

where $k = 0, \ldots, T-1$ denotes the time index along the prediction horizon. The mean $\boldsymbol{\mu}_i^k \in \mathbb{R}^2$ and the covariance matrix $\boldsymbol{\Sigma}_i^k \in \mathbb{R}^{2\times 2}$ represent the predicted position and the associated uncertainty of the i-th human k steps into the future, respectively.

3.4.4 MPC For Coverage Control

Here, we present and discuss the proposed optimization-based framework for decentralized coverage control. Specifically, our goal is to use MPC to minimize a specifically designed cost function while satisfying safety and dynamics constraints.

Cost Function

We define the cost function in our optimization problem as the sum of two contributions, $\mathcal{J}_i = \mathcal{J}_i^{\text{cov}} + \mathcal{J}_i^{\text{u}}$. The first replicates the idea of coverage control, and the second encourages smooth control inputs.

1. **Coverage Cost** First, we define a suitable coverage cost function that can be minimized in a decentralized manner by each agent. Similarly to the definition of the limited-range coverage optimization function $\mathcal{H}^r$ in (2.9), we define the cost function $\mathcal{J}_i^{\text{cov}} : Q \to \mathbb{R}$ for the i-th robot as:

$$\mathcal{J}_i^{\text{cov}} = \sum_{k=0}^{T-1} \mathcal{J}_i^{\text{cov}}(k) = \sum_{k=0}^{T-1} \int_{V_i^r(t_0)} \|\mathbf{q} - \mathbf{p}_i^k\|^2 \phi(\mathbf{q})\, d\mathbf{q} \tag{3.14}$$

   where $\mathbf{p}_i^k := \mathbf{p}_i(t_0 + k)$. It is important to note that the integral in the cost function is consistently evaluated over the limited Voronoi cell computed at the initial time step, $t = t_0$, denoted as $V_i^r(t_0)$, throughout the entire prediction horizon. This approximation is necessary to maintain a fixed integration domain. Without this approximation, the integration domain would change along the horizon, resulting in a nonlinear optimization problem that would be more challenging to solve in real time. It is easy to show that minimizing $\mathcal{J}_i^{\text{cov}}$ returns a set of inputs $\mathcal{U}_i = [\mathbf{u}_i^0, \ldots, \mathbf{u}_i^{T-1}]$ where $\mathbf{u}_i^0$ steers the i-th robot toward the centroid of V_i^r as in (2.10) for single integrator dynamics. For the reader's convenience, we recall the optimal control input for limited-range coverage introduced in (2.10):

$$\mathbf{u}_i = k(\mathbf{C}_{V_i^r} - \mathbf{p}_i) \tag{(2.10)}$$

   Proposition 1. *Consider the i-th robot moving in an obstacle-free, convex environment obeying a single integrator dynamics law, $\dot{\mathbf{p}}_i = \mathbf{u}_i$. The first input $\mathbf{u}_i^0$ of*

the sequence $\mathcal{U}_i$ *minimizing* (3.14) *corresponds to* (2.10).

Proof. Following the proof in [37, Theorem 1], we show that (2.10) implements a gradient descent of the cost function (3.14). The partial derivatives of the cost function can be computed as follows:

$$\begin{aligned}\frac{\partial \mathcal{J}_i^{\text{cov}}}{\partial \mathbf{p}_i} &= -2\int_{V_i^r} (\mathbf{q}-\mathbf{p}_i)\phi(\mathbf{q})\, d\mathbf{q} \\ &= -2M_{V_i^r}(\mathbf{C}_{V_i^r}-\mathbf{p}_i)\end{aligned} \tag{3.15}$$

where $M_{V_i^r} = \int_{V_i^r} \phi(\mathbf{q})\, d\mathbf{q}$ is the mass of the limited Voronoi cell V_i^r. Therefore, applying (2.10) to the time derivative of the cost function we get:

$$\begin{aligned}\dot{\mathcal{J}}_i^{\text{cov}} &= \frac{\partial \mathcal{J}_i^{\text{cov}}}{\partial \mathbf{p}_i}\dot{\mathbf{p}}_i \\ &= -[2M_{V_i^r}(\mathbf{C}_{V_i^r}-\mathbf{p}_i)]^\top k(\mathbf{C}_{V_i^r}-\mathbf{p}_i) \leq 0.\end{aligned} \tag{3.16}$$

Given the quadratic form and the negative sign in front of the expression, the time derivative is non-positive, proving the statement. □

Based on the above considerations, the traditional approach can be regarded as a special case of the proposed method. In particular, when the environment is convex, the MPC formulation becomes unconstrained, and—assuming single-integrator dynamics—the resulting solution coincides with that of the traditional approach.

2. **Control Input Cost** Given a positive definite input cost matrix $\mathbf{R} \in \mathbb{R}^{m\times m}$, we define the second term of the cost function, namely $\mathcal{J}_i^{\text{u}} : \mathbb{R}^m \to \mathbb{R}$, as:

$$\mathcal{J}_i^{\text{u}} = \mathbf{u}_i^\top \mathbf{R}\mathbf{u}_i. \tag{3.17}$$

This additional term penalizes aggressive maneuvers and energy consumption, making smooth trajectories preferable.

The idea behind the design of $\mathcal{J}_i$ is to maintain the convergence properties of $\mathcal{H}^r$, extensively discussed in the literature, while extending its applicability to more generic motion models, instead of the simplistic single integrator usually adopted in traditional approaches [33, 37].

Constraints

The main advantage of using MPC over a traditional closed-form solution lies in the possibility to encode constraints into the optimization problem.

1. **Motion Dynamics**: We want the solution of the optimization problem to follow the dynamic equation (3.12), which can be written in a discretized form as:

$$\mathbf{x}_i^{k+1} = f\big(\mathbf{x}_i^k\big) + g(\mathbf{x}_i^k)\mathbf{u}_i^k \Delta t, \quad \forall k \in \{0,\dots,T-1\} \tag{3.18}$$

2. **Obstacle avoidance**: Obstacle avoidance is not directly addressed in traditional approaches. Usually, a downstream solution is adopted that modifies the control input from coverage control to enhance safety. Instead, we encode safety constraints into the optimization problem to obtain an optimal solution directly applicable to the robot. Given a safety distance $D_s \in \mathbb{R}_{>0}$, we define the constraint for each obstacle $\boldsymbol{\varrho}_j$ as:

$$\begin{aligned} \|\mathbf{p}_i^k - \boldsymbol{\varrho}_j\|^2 - D_s^2 \geq 0, \quad & \forall j \in \{1, \dots, N_o\} \\ & \forall k \in \{0, \dots, T-1\} \end{aligned} \tag{3.19}$$

3. **Collision avoidance**: Similarly to obstacle avoidance, we define a set of constraints to avoid collisions with other robots as:

$$\begin{aligned} \|\mathbf{p}_i^k - \mathbf{p}_j\|^2 - D_s^2 \geq 0, \quad & \forall j \neq i \\ & \forall k \in \{0, \dots, T-1\} \end{aligned} \tag{3.20}$$

MPC Optimization Problem

Combining all components, we formulate a quadratic optimization problem that yields a sequence of control inputs $\mathcal{U}_i = [\mathbf{u}_i^0, \dots, \mathbf{u}_i^{T-1}]$ for the i-th robot. In accordance with the standard MPC framework, only the first input of the optimized sequence $\mathbf{u}_i^0$ is applied at each time step, and the optimization is repeated at the next iteration. For the sake of notation simplicity, we omit the explicit dependence on the prediction step k, although all constraints are enforced at each step of the prediction horizon:

$$\min_{\mathcal{U}_i} \ \mathcal{J}_i^{\text{cov}} + \mathcal{J}_i^{\text{u}} \tag{3.21a}$$

$$\text{s.t. } \mathbf{x}_i^{k+1} = f\big(\mathbf{x}_i^k\big) + g(\mathbf{x}_i^k)\mathbf{u}_i^k \Delta t \tag{3.21b}$$

$$\|\mathbf{p}_i^k - \boldsymbol{\varrho}_j\|^2 - D_s^2 \geq 0, \quad j \in \{1, \dots, N_o\} \tag{3.21c}$$

$$\|\mathbf{p}_i^k - \mathbf{p}_j\|^2 - D_s^2 \geq 0, \quad \forall j \neq i \tag{3.21d}$$

$$\mathbf{x}_i^k \in \mathcal{X} \tag{3.21e}$$

$$\mathbf{u}_{\min} \preceq \mathbf{u}_i^k \preceq \mathbf{u}_{\max} \tag{3.21f}$$

In the Quadratic Programming (QP) problem above, the state $\mathbf{x}_i$ is constrained to lie within the admissible set $\mathcal{X}$ (3.21e), and the control input is subject to the bounds defined in (3.21f), where the operator $\preceq$ denotes element-wise inequality.

3.4.5 Human-Aware Coverage Control

Introducing humans into the coordination problem significantly increases complexity. Unlike robots, humans do not share their motion plans, and their behavior is often unpredictable and partially observable. This uncertainty makes it difficult to guarantee safety and maintain performance. To address this, the control framework must incorporate models that predict or bound human motion and adjust robot behavior accordingly.

This section describes how we extend the baseline controller to account for the presence of humans in the environment.

Specifically, our goal is to coordinate robots to perform coverage control and avoid humans during their motion. Assuming access to a probabilistic predictor for human trajectories (e.g., [74, 76, 87]), each robot can construct a probabilistic map over the future positions of nearby humans, represented as Gaussian distributions characterized by a mean $\boldsymbol{\mu}_i^k$ and covariance matrix $\boldsymbol{\Sigma}_i^k$ for the k-th prediction step. The predictor takes as input the past $M \in \mathbb{N}$ steps of the human trajectory and outputs a probability density function over future positions for T steps ahead, where T matches the prediction horizon of the MPC. Given the probabilistic nature of the problem, we design a probabilistic collision avoidance mechanism with chance constraints. Briefly, our goal is to design a constraint of the form

$$P\big(\|\boldsymbol{\xi}_j - \mathbf{p}_i\| < D_s\big) \leq \alpha, \tag{3.22}$$

meaning that the probability of the distance between the j-th human and the i-th robot falling below the safety threshold D_s must remain below the risk level $\alpha \in [0, 1)$. Starting from the notion of squared Mahalanobis distance d_M, defined as:

$$d_M^2 = (\mathbf{p}_i - \boldsymbol{\mu}_j)^\top \boldsymbol{\Sigma}_j^{-1} (\mathbf{p}_i - \boldsymbol{\mu}_j) \tag{3.23}$$

evaluating the distance from a point to a probabilistic distribution characterized by mean $\boldsymbol{\mu}_j$ and covariance matrix $\boldsymbol{\Sigma}_j$, we can define our chance constraint for human avoidance as follows (see [88]):

$$d_M^2 \geq -2 \ln \Big(\sqrt{\det(2\pi \boldsymbol{\Sigma}_j)} \frac{\alpha}{\pi D_s^2} \Big). \tag{3.24}$$

Furthermore, to account for the growing uncertainty in human motion over time, we introduce a slack variable $\delta^k \in \mathbb{R}_{\geq 0}$ that relaxes the constraint at each prediction step k. This allows the constraint to become progressively less conservative further along the horizon. The modified constraint at step k is given by

$$(d_M^k)^2 \geq -2 \ln \Big(\sqrt{\det(2\pi \boldsymbol{\Sigma}_j^k)} \frac{\alpha}{\pi D_s^2} \Big) - \delta^k. \tag{3.25}$$

Progressive relaxation is achieved by defining an additional cost term $\mathcal{J}_\delta : \mathbb{R} \to \mathbb{R}$ as:

$$\mathcal{J}_\delta = \sum_{k=0}^{T-1} w^k \delta^k \tag{3.26}$$

where $w^k \in \mathbb{R}_{>0}$ is a linearly decaying weighting factor defined as $w^k = \Omega\big(1 - \frac{k}{T}\big)$, where $\Omega \in \mathbb{R}$ is the initial value.

Finally, the overall optimization problem for human-aware coverage control can be writ-

ten, adapting (3.21), as:

$$\min_{\mathcal{U}_i} \mathcal{J}_i^{\text{cov}} + \mathcal{J}_i^{\text{u}} + \mathcal{J}_i^{\delta} \tag{3.27a}$$

$$\text{s.t. } \mathbf{x}_i^{k+1} = f\big(\mathbf{x}_i^k\big) + g(\mathbf{x}_i^k)\mathbf{u}_i^k \Delta t \tag{3.27b}$$

$$\|\mathbf{p}_i^k - \boldsymbol{\varrho}_j\|^2 - D_s^2 \geq 0 \tag{3.27c}$$

$$\mathbf{x}_i^k \in \mathcal{X} \tag{3.27d}$$

$$\mathbf{u}_{\min} \preceq \mathbf{u}_i^k \preceq \mathbf{u}_{\max} \tag{3.27e}$$

$$(d_M^k)^2 \geq -2 \ln \left(\sqrt{\det(2\pi \boldsymbol{\Sigma}_j^k)} \frac{\alpha}{\pi D_s^2} \right) - \delta^k. \tag{3.27f}$$

3.4.6 Experimental Evaluation

This section evaluates our solution's performance against standard approaches, highlighting the benefits of incorporating trajectory predictions. We will run simulations in three different settings, specifically: *(A)* a convex environment without humans, *(B)* a non-convex environment without humans, and *(C)* a non-convex environment with humans. Robots operate in a fully decentralized manner without inter-agent communication. The likelihood function ϕ is defined as a Gaussian Mixture Model with random components. The prediction horizon is set to $T = 10$. We employ different dynamic models across experiments to evaluate the adaptability of our solution. Results are evaluated over the following performance metrics:

- *Coverage Optimization Function* $\mathcal{H} : Q \to \mathbb{R}$, corresponding to the range-unlimited coverage function (2.3), indicating how close the team is to the *centroidal Voronoi tessellation*:

$$\mathcal{H}(\mathcal{P}) = -\sum_{i=1}^{N_r} \int_{V_i} \|\mathbf{q} - \mathbf{p}_i\|^2 \phi(\mathbf{q}) d\mathbf{q}. \tag{3.28}$$

- *Coverage Effectiveness* $\mathcal{E} : Q \to \mathbb{R}$, assessing how well the likelihood density $\phi(\mathbf{q})$ is covered by the limited-range capabilities of the robots:

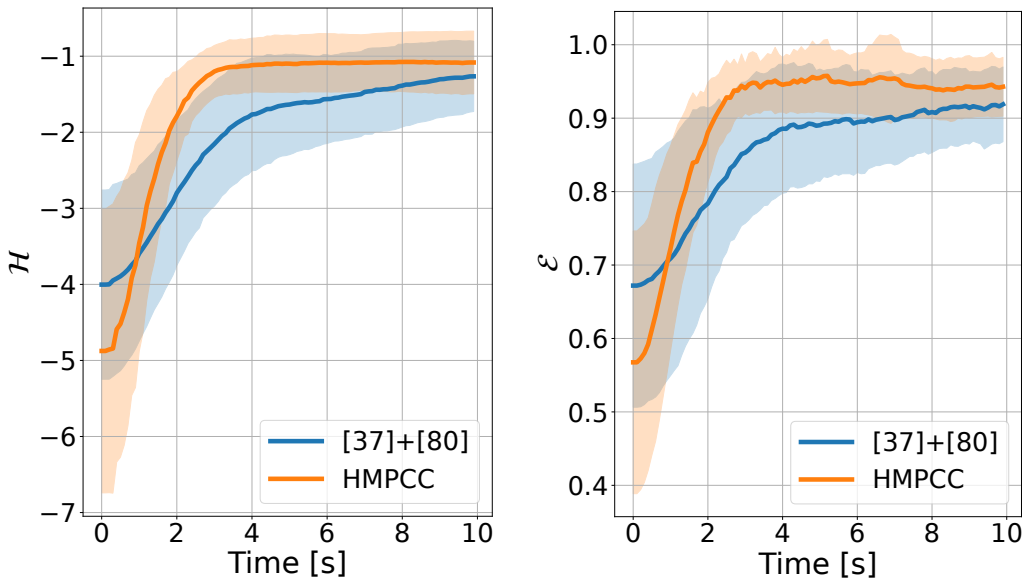
$$\mathcal{E} = \frac{\sum_{i=1}^{N} \int_{V_i} \phi(\mathbf{q}) d\mathbf{q}}{\int_Q \phi(\mathbf{q}) d\mathbf{q}}. \tag{3.29}$$

We compare our method against a hybrid approach that combines limited-range coverage control [37] with potential-based repulsion for collision avoidance [80].

Convex Environment without Humans

The first set of simulations was conducted in simplified scenarios commonly assumed by traditional methods, featuring an obstacle-free environment. The experiments are conducted with $N = 6$ robots, randomly initialized in a 10×10 m^2 environment, obeying a double integrator dynamic law:

$$\mathbf{x}_i(t+1) = A\mathbf{x}_i(t) + B\mathbf{u}_i(t)\Delta t \tag{3.30}$$



(a) Coverage Function

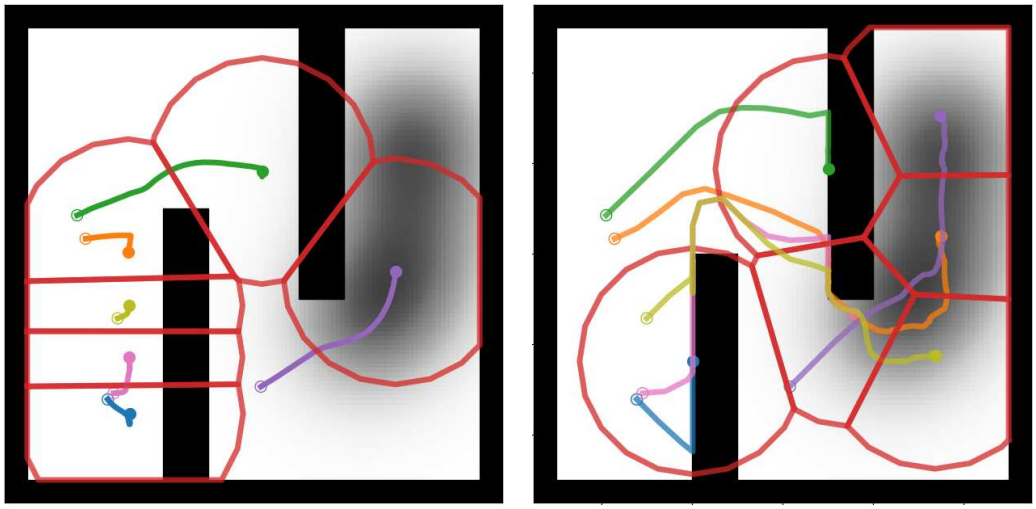
(b) Coverage Effectiveness

Figure 3.18: Convergence rate: comparison between limited-range coverage control [37] with repulsive collision avoidance [80] and our HMPCC. Our approach achieves faster convergence for both metrics.

where the state of the i-th robot $\mathbf{x}_i = \begin{bmatrix} \mathbf{p}_i \\ \dot{\mathbf{p}}_i \end{bmatrix}$ contains its position and velocity, and $A = \begin{bmatrix} 0, I; 0, 0 \end{bmatrix} \in \mathbb{R}^{4\times4}$, $B = \begin{bmatrix} 0; I \end{bmatrix} \in \mathbb{R}^{4\times2}$. Here $0 \in \mathbb{R}^{2\times2}$, $I \in \mathbb{R}^{2\times2}$ are the zero and identity matrix, respectively. The values of the coverage objective function $\mathcal{H}$ and the effectiveness $\mathcal{E}$ were recorded at each time-step over 10 simulation runs. The duration of each run is set to 10 s with time-step $\Delta t = 0.1$ s. Results are presented in Fig. 3.18 as mean values with standard deviations. As shown in the plots, HMPCC achieves faster convergence for both $\mathcal{H}$ and $\mathcal{E}$. This improved efficiency stems from the fact that the control input is not directly tied to the distance from the centroid. Instead, planning to minimize the cost function (3.14) enables robots to move more efficiently toward regions of higher density, resulting in accelerated convergence.

Non-Convex Environment without Humans

As previously noted, a major limitation of traditional coverage control approaches is their reliance on the assumption of convex environments. When obstacles are present, the addition of a separate obstacle-avoidance mechanism often causes robots to become trapped in local minima, preventing them from reaching the desired coverage areas. To test how HMPCC behaves in those scenarios, we ran another set of simulations, increasing the duration of each run to 15 s. We employ teams with $N = \{4, 6, 8, 10\}$ robots following single integrator dynamics $\mathbf{p}_i(t+1) = \mathbf{p}_i(t) + \mathbf{u}_i(t)\Delta t$.
Fig. 3.19 illustrates the difference between the baseline hybrid approach, combining [37] and [80], shown in Fig. 3.19a, and our proposed method in Fig. 3.19b. It is evident that the baseline controller gets stuck when direct paths are obstructed, whereas our constrained optimization-based approach enables the robots to navigate around obstacles and reach the target areas. As a result, HMPCC achieves superior final performance, as



(a) Baseline **(b)** HMPCC

Figure 3.19: Comparison in a significant simulation run between limited-range coverage control [37] with repulsive collision avoidance [80] and our MPC-based approach. Robots trajectories are depicted with different colors, their starting and final positions are indicated by empty and filled dots, respectively. (a) The baseline is prone to local minima, while (b) HMPCC allows robots to plan obstacle-avoiding trajectories to reach areas of interest.

illustrated in Fig. 3.20a for the coverage function $\mathcal{H}$, and Fig. 3.20b for the effectiveness $\mathcal{E}$.

Non-Convex Environment with Humans

Finally, we introduced humans into the environment to evaluate how the proposed method benefits from predicting their trajectories. Similarly to the previous approach, we ran sets of 10 simulations each to compare HMPCC with the baseline without prediction. We set the number of robots to $N = 6$ and humans to $N_h = \{3, 6, 9\}$, and the risk factor for the chance constraint (3.22) to $\alpha = 0.1$. Unicycle robots were employed in this evaluation phase, with a motion law as follows:

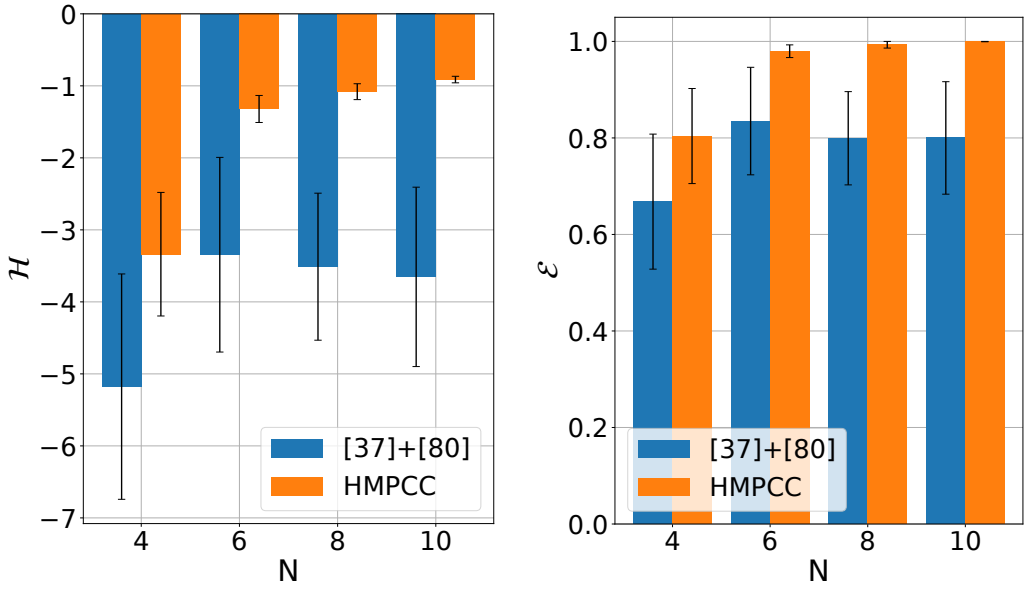
$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \begin{bmatrix} \cos\theta_i & 0 \\ \sin\theta_i & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_i \\ \omega_i \end{bmatrix} \Delta t, \tag{3.31}$$

where the state $\mathbf{x}_i = [\mathbf{p}_i; \theta_i] \in \mathbb{R}^3$ indicates position and orientation of the i-th robot, while $v_i, \omega_i \in \mathbb{R}$ are its linear and angular velocity, respectively. Instead, we model human motion as:

$$\boldsymbol{\xi}_j(t+1) = \boldsymbol{\xi}_j(t) + \begin{bmatrix} \cos\vartheta_j \\ \sin\vartheta_j \end{bmatrix} \nu_j \Delta t \tag{3.32}$$

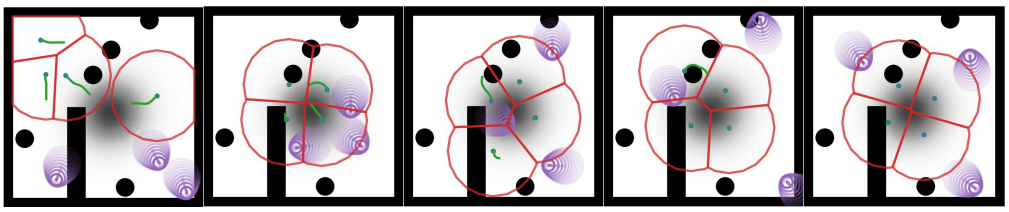
$$\vartheta_j(t+1) = \vartheta_j(t) + \psi_j \Delta t \tag{3.33}$$

with $\nu_j \in \mathbb{R}_{\geq 0}$ and $\psi_j \sim \mathcal{N}(0, \sigma)$ being a constant linear velocity and a random angular velocity, respectively. Human motion prediction was modeled using a constant-velocity assumption, resulting in the probabilistic trajectory defined in (3.13), with the following



(a) Coverage Function **(b)** Coverage Effectiveness

Figure 3.20: Evaluation in non-convex environments with increasing number of robots. Our solution outperforms the baseline both in terms of $\mathcal{H}$ (a) and $\mathcal{E}$ (b).



(a) $t = 0.0$ s **(b)** $t = 3.0$ s **(c)** $t = 6.5$ s **(d)** $t = 9.0$ s **(e)** $t = 11.0$ s

Figure 3.21: Snapshots of a representative task execution involving a team of 4 unicycle robots and 3 humans. Robots are shown as blue dots, their planned trajectories as green curves, and humans in purple. The uncertainty associated with human motion is visualized as expanding circles along the prediction horizon. (a) Robots start in random positions, (b) reach the area of interest, (c) temporary disperse to make room for approaching humans, (d) avoid obstacles in the environment, and (e) finally return to the area of interest.

mean and covariance update:

$$\boldsymbol{\mu}_j^k = \boldsymbol{\mu}_j^{k-1} + \begin{bmatrix} \cos\vartheta_j^{k-1} \\ \sin\vartheta_j^{k-1} \end{bmatrix} \hat{\nu}_j, \qquad \boldsymbol{\Sigma}_j^k = \boldsymbol{\Sigma}_j^{k-1} + \mathbf{Q}, \tag{3.34}$$

where $\mathbf{Q} \in \mathbb{R}^{2\times 2}$ is a noise matrix that accounts for growing uncertainty over the prediction horizon, and $\hat{\nu}_j \in \mathbb{R}_{\geq 0}$ is the j-th human velocity predicted from the last M steps:

$$\hat{\nu}_j^t = \frac{\|\boldsymbol{\xi}_j^t - \boldsymbol{\xi}_j^{t-M+1}\|}{(M-1)\Delta t} \tag{3.35}$$

In this setting, we also analyzed the success rate across multiple simulation runs to compare our method with the baseline. Execution was considered unsuccessful if at least one robot collided with an obstacle or a human, or exited the boundaries of the environment. A summary of the results is provided in Table 3.1. It is important to note that the values of $\mathcal{E}$ and $\mathcal{H}$ are captured at the end of each simulation, which is fixed to 15 s. For this reason, robots may have temporarily moved away from high-density areas to avoid

Table 3.1: Results in human-filled scenarios.

N_h	$\mathcal{E}$	$\mathcal{H}$	Success rate	
	HMPCC	**HMPCC**	**Baseline**	**HMPCC**
3	0.952	-1.110	60%	100%
6	0.942	-1.613	20%	100%
9	0.875	-1.993	40%	80%

collisions with humans. Moreover, we did not report the metrics for the baseline method due to an insufficient number of successful executions, rendering the results statistically insignificant. Additionally, Fig. 3.21 presents snapshots of a representative task execution, illustrating how the robots plan trajectories that avoid both obstacles and humans, and temporarily relocate from areas of interest to make room for approaching humans. Finally, we tested HMPCC in a physical simulation using Gazebo and TurtleBot3 unicycle robots to assess the computational feasibility of our approach. We prepared a simulation with 3 robots and 3 humans in a 20×20 m^2 environment with obstacles, as illustrated in Fig. 3.17. The simulation was run on a PC equipped with an *Intel Core i7* CPU, an *NVIDIA GeForce RTX 4060* GPU, and 16 GB of RAM, using ROS for the implementation. We set the control frequency of the robots to 10 Hz.

Summary of the Results

In summary, the presented results demonstrate several advantages of our method over traditional coverage control approaches:

- **faster convergence** — even in simple environments, an optimization-based approach appears to be more efficient in reaching target areas;
- **robustness to complexity** — by encoding dynamics and obstacle avoidance as constraints, our method handles non-convex environments and varied motion models, reducing susceptibility to local minima; and
- **human-aware deployment** — the ability to predict the area where humans are more likely to be in the future fits with the planning strategy to efficiently avoid them.

3.4.7 Conclusion

In this section, we introduced HMPCC, a human-aware optimization-based coverage control strategy. By formulating collision avoidance and robot dynamics as constraints, HMPCC overcomes key limitations of traditional methods. The integration of human trajectory prediction ensures safe operation even when human intentions are unknown. Experimental results show that HMPCC outperforms a traditional baseline in both safety and coverage efficiency. Future work will focus on integrating more advanced prediction models (e.g., FlowChain [76]), exploring human-robot collaborative scenarios, and validating the approach through real-world experiments to assess the computational feasibility of our solution.

Chapter 4

Handling Anisotropic Constraints: Control with Limited Angular Field of View

In the previous Chapter, we addressed the challenge of monitoring and exploring unknown environments under the constraint of limited sensing capabilities, modeled as an omnidirectional circular region around each agent. While this assumption is suitable for sensors such as LiDARs or standard range finders, many real-world robotic platforms rely on visual sensors—such as monocular or depth cameras—that are inherently directional. This introduces a significant additional layer of complexity: an agent can only sense neighbors and the environment within a specific cone-shaped Field of View (FoV), making the maintenance of connectivity and situational awareness far more challenging.

In this Chapter, we extend our analysis to multi-robot systems equipped with directional sensing capabilities. We first address the coordination problem by imposing strict constraints that force robots to maintain their neighbors within their limited FoV at all times, ensuring continuous connectivity. Subsequently, we relax this requirement to allow for intermittent disconnections, introducing a more flexible control strategy. To ensure safety in this relaxed setting, we develop a specialized controller that permits temporary loss of visual contact but triggers an active reorientation maneuver to regain contact with unobserved neighbors whenever a risk of collision is detected. Finally, we propose a hierarchical approach that aggregates subgroups of robots into "virtual agents", effectively emulating a fully actuated system with omnidirectional sensing capabilities to overcome individual limitations. These methodologies are validated through theoretical analysis and experimental results, demonstrating effective coordination even under severe sensing constraints.

4.1 Directed Graph Topology Preservation in Multi-Robot Systems with Limited Field of View Using Control Barrier Functions

This Section addresses the crucial challenge of maintaining the directed graph topology in multi-robot systems, particularly when operating under limited field-of-view constraints and with a lack of communication among robots. Traditional methods for multi-robot coordination rely heavily on inter-robot communication, which may not always be feasible, particularly in constrained or hostile environments. Our work presents a novel distributed control algorithm that leverages Control Barrier Functions (CBFs) to maintain the graph topology of a multi-robot system based solely on local, onboard sensor data. This approach is particularly beneficial in situations where external communication channels are disrupted or unavailable.

The key contributions of this research are threefold: First, we design a novel control algorithm that efficiently maintains the graph topology in multi-robot systems using CBFs, which operate on neighbor detection data. Second, we perform an experimental evaluation of the algorithm, demonstrating its efficacy in controlling the flight of a team of drones using only local robot data. Third, we apply our methodology to a distributed coverage control scenario, showing that our approach can effectively manage a multi-robot system using only local information.

4.1.1 Introduction

The large majority of the distributed algorithms employed to coordinate swarms of robots rely on the ability of each agent in the swarm to obtain information about its surroundings. The transfer of information through communication channels can be modeled using a graph whose nodes represent the robots and the edges are the direct communication links. The correct performance of many distributed algorithms relies on the properties of the graph describing the communication network. Preservation of topological properties similar to the one depicted in Fig. 4.1, such as connectivity, has been studied in the literature, typically considering undirected (i.e., bi-directional) communication, and explicit communication among the robots [89, 90, 91, 92, 93]. However, in a scenario where the robots cannot communicate with each other, the conventional network graph modeling and its properties do not appropriately describe the system, nor provide performance guarantees. Each robot then has to rely on its onboard sensors to decide its future action. However, in scenarios where robots are incapacitated from inter-communication, the conventional network graph model becomes inadequate, failing to guarantee system performance. In such contexts, robots must depend solely on onboard sensors for decision-making. The effectiveness of distributed algorithms, even under these constraints, has been demonstrated, particularly when only relative positional information of neighbors is available. For instance, in coverage control, a robot may only need data from its immediate neighbors to compute its target.

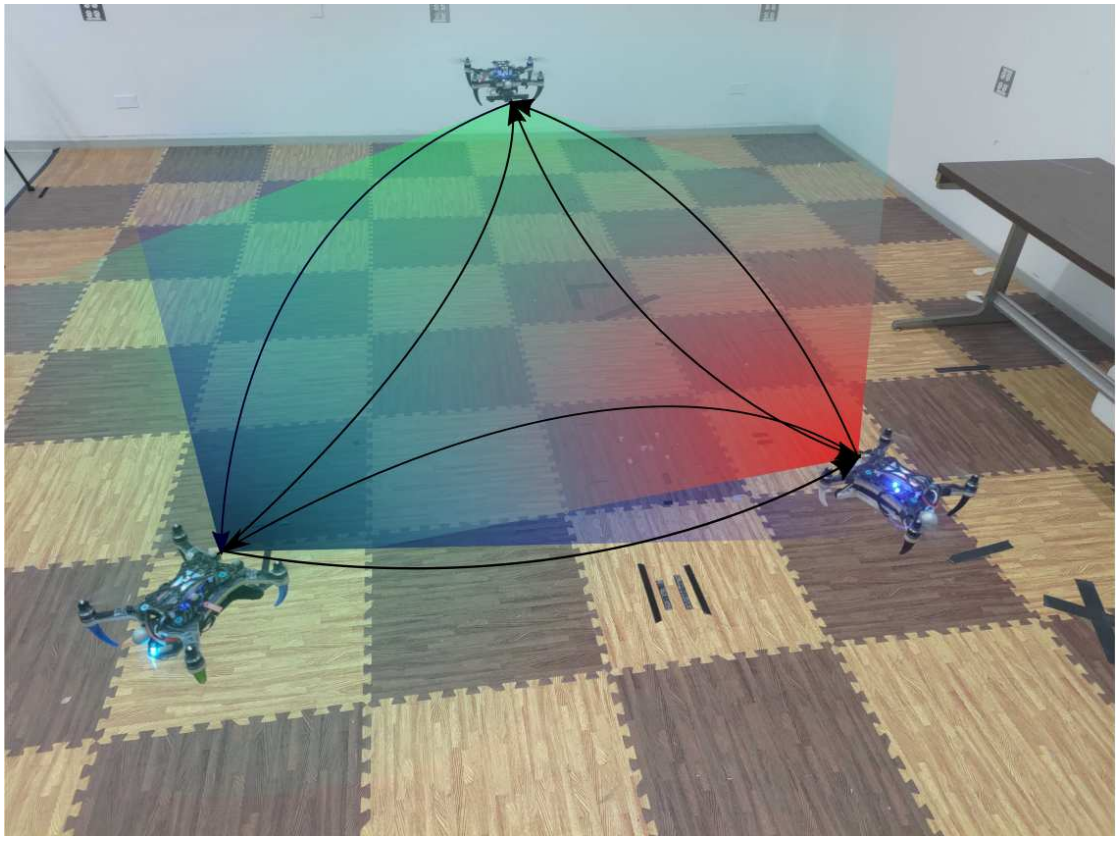


Figure 4.1: Graph topology preservation according to the depicted arrows during a coverage control problem task with CBF enforcing visual constraint using aerial robots.

Related Works and Contributions

Our work addresses the challenge of topology preservation in a fleet of drones governed by a distributed algorithm, specifically in scenarios where communication is suddenly interrupted. The drones must then independently maintain the requisite level of information for proper operation. While a variety of sensors, from IR to lidars and cameras, are available for environment scanning and neighbor detection, these often have anisotropic capabilities. The information level of a single drone can thus be represented by a directed graph. Maintaining this graph's topology (i.e., preserving all outgoing edges from a node) equates to sustaining the information level of each drone. Similar solutions for this problem have been presented in the literature, however, there are some key differences to be considered. In [94], the authors face the problem of maintaining the continuous tracking of a target using a fleet of drones. They propose a multi-objective distributed optimization problem that ensures connectivity maintenance of the fleet, obstacle avoidance, and occlusion avoidance. To achieve these safety constraints, the authors employ a controller based on Control Barrier Functions (CBFs), that modifies the motion of the drones. However, since in their example drones can communicate and exchange information, the authors did not include the maintenance of the target inside the field of view in the optimization problem, allowing partial loss of visibility. Without the ability to communicate, drones not facing each other would crash into each other since they do not have accurate information on their surroundings. In [95], the authors propose a Barrier Function-based Image Servoing control that maintains detected features in the field of

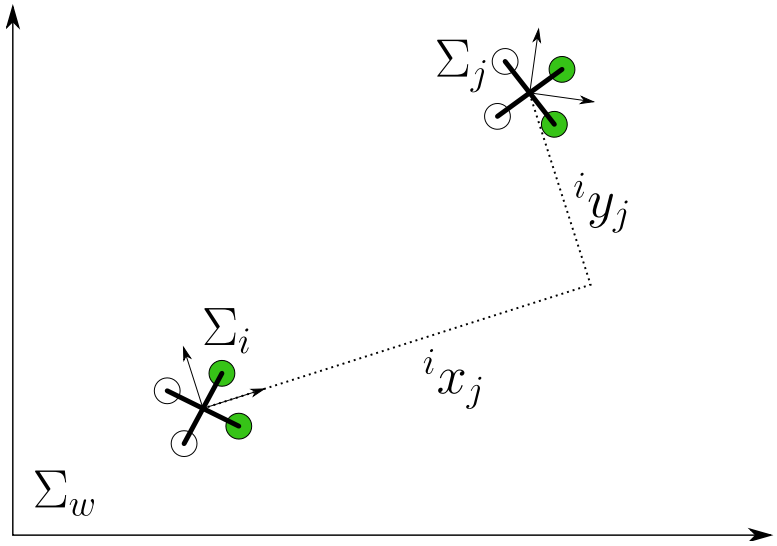


Figure 4.2: Visualization of the system and relative coordinates.

view. The controller enables the system to position itself with respect to a detected object while maintaining said object in the field of view to ensure proper feedback information is provided. This problem is very similar to the problem addressed in this section but with a substantial difference: the target for the system, by definition, satisfies the constraints. In this work, we treat the problem of achieving a general task that may have a setpoint that does not satisfy the constraints. The contribution of this work can then be summarized as follows:

- The design of a novel distributed control algorithm that ensures the graph topology maintenance for a multi-robot system, leveraging CBFs on neighbor detection to allow the continuation of the task at hand if possible.
- The experimental evaluation of the algorithm to control the flight of a team of drones, using only local robot data (i.e., no communication).
- The application of the proposed approach to coverage control using only local information.

Notation and Definitions

Let $\mathcal{G} = (\mathcal{U}, \mathcal{E})$ be a graph characterized by a set $\mathcal{U}$ of vertices and a set $\mathcal{E} \subseteq \mathcal{U} \times \mathcal{U}$ of directed edges. Given an edge $(i, j) \in \mathcal{E}$, then the vertex j is a neighbor of the vertex i. Let $\mathcal{N}_i \in \mathbb{N}$ be the set of neighbors of the vertex i in $\mathcal{G}$. A graph $\mathcal{G}$ is said to be *undirected* if $(i, j) \in \mathcal{E}$ implies $(j, i) \in \mathcal{E}$. Let us define the *proximity graph* as a graph $\mathcal{G}_\mathcal{P}$ in which the edge set $\mathcal{E}_{\mathcal{G}_\mathcal{P}}$ depends on the location of the vertices. In this section, we consider graphs defined for points $\{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ in $\mathbb{R}^2$. Hence, we can define a proximity graph function $G : \mathbb{F}(\mathbb{R}^2) \to (\mathcal{U}, \mathcal{E})$ that associates to $\mathcal{P} \in \mathbb{F}(\mathbb{R}^2)$ an undirected graph with the set $\mathcal{P}$ of vertices and the set $\mathcal{E}_\mathcal{G}(\mathcal{P})$ of edges, where $\mathcal{E}_\mathcal{G} : \mathbb{F}(\mathbb{R}^2) \to \mathbb{F}(\mathbb{R}^2 \times \mathbb{R}^2)$ has the property that $\mathcal{E}_\mathcal{G}(\mathcal{P}) \subseteq \mathcal{P} \times \mathcal{P} \setminus \text{diag}(\mathcal{P} \times \mathcal{P})$.
We consider robots moving in a closed environment free of obstacles. The inertial reference frame associated with the environment is denoted with Σ_w. The local reference frame attached to the body of each robot i in the environment is denoted with Σ_i.

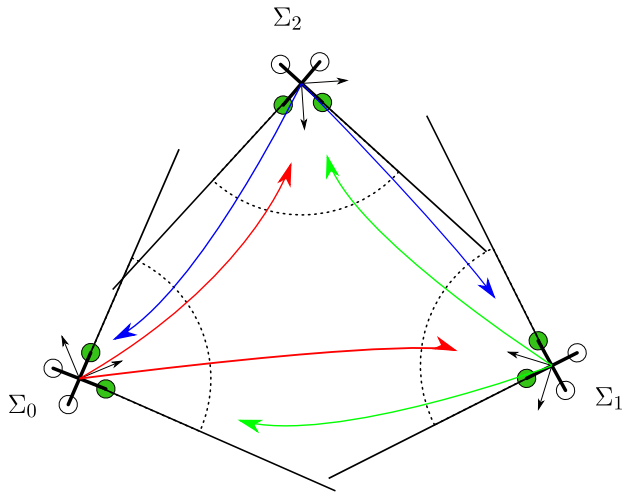


Figure 4.3: Visualization of the graph describing the system. Each drone has all the others inside the field of view, therefore each drone has information on the whole system.

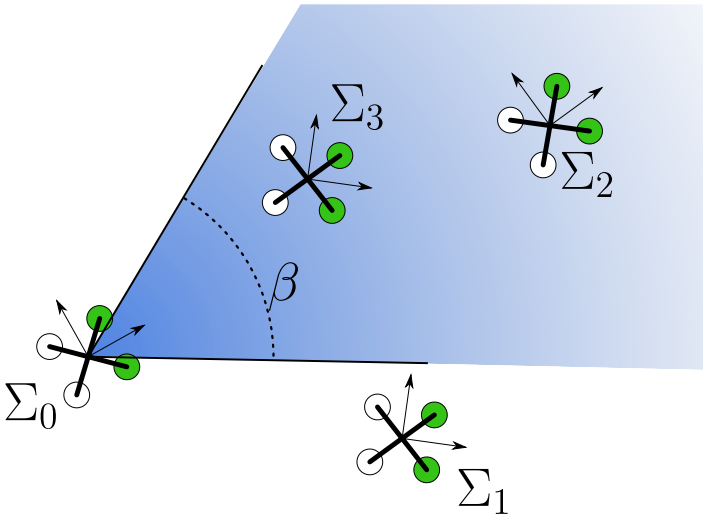


Figure 4.4: Visualization of the field of view of drone 0. In particular, only drone 3 and 2 are detected since 1 is on the right of the visible space.

Figure 4.2 provides a visualization of the considered reference frames. The notation ${}^{w}\Sigma_i$ denotes the relative transformation from the global reference frame Σ_w to the local frame Σ_i expressed in the global frame. Similarly, the notation ${}^{i}\mathbf{x}$ indicates a generic vector $\mathbf{x}$ expressed relatively to Σ_i. If not specified, a vector is expressed with respect to Σ_w. The notation ${}^{w}R_i \in SO2$ is used to denote the rotation matrix from Σ_w to Σ_i, such that ${}^{w}\mathbf{x} = {}^{w}R_i {}^{i}\mathbf{x}$.

4.1.2 Problem Description

Consider a team of n aerial robots flying in an obstacle-free bounded environment. The aerial robots fly using only information from onboard sensors. Each drone knows its global position at takeoff and estimates the pose of its local reference frame ${}^{w}\Sigma_i$ through odometry information. In the following, we assume that all the drones fly at the same

altitude and move with a bounded planar velocity. Furthermore, each drone can detect the neighbors that are in the field of view of its camera. The set of detected neighbors for each robot i is denoted as $\mathcal{N}_i \in \mathbb{N}$. The relative 2D position of the neighbour j with respect to Σ_i is expressed as ${}^i\mathbf{p}_j = [{}^ix_j, {}^iy_j]^\top$. Let $\mathcal{G}$ be the directed graph describing the connection between the aerial robots. Specifically, each aerial robot is represented in $\mathcal{G}$ as a vertex, while an edge (i, j) is in the graph if and only if j is detected by i, formally expressed as $(i, j) \in \mathcal{E}_\mathcal{G} \iff j \in \mathcal{N}_i$. A visualized example of the graph is provided in Fig. 4.3. The state of each aerial robot i is denoted with ${}^w\chi_i = \begin{bmatrix} {}^wx_i & {}^wy_i & {}^w\theta_i \end{bmatrix}^\top$, where ${}^wx_i, {}^wy_i \in \mathbb{R}$ represent the coordinates of the 2D position of Σ_i in Σ_w, while ${}^w\theta_i$ is the yaw angle. The altitude is excluded since we assume every robot flies at the same constant altitude. The aerial robot is controlled to obtain the following kinematic model

$$ {}^w\dot{\chi}_i = {}^wR_i\,{}^i\mathbf{u}_i = {}^wR_i \begin{bmatrix} {}^iv_x \\ {}^iv_y \\ {}^i\omega \end{bmatrix}, \tag{4.1} $$

where ${}^iv_x, {}^iv_y, \omega \in \mathbb{R}$ represent the linear and angular velocities of the drone in Σ_i. Furthermore, each aerial robot has a limited angular sector in which it can detect neighboring drones. This angular sector is centered at the local X axis with amplitude β, as visualized in Fig. 4.4.

The drones in this configuration are instructed to perform a decentralized task. Assume, without loss of generality, that the task requires each drone to know the relative position of its neighbors as well as its own position in the environment. In ideal conditions, this information is usually obtained through wireless communication channels, onboard sensors, or a combination of both. However, in the analyzed scenario, the communications are externally interrupted, therefore the drones are required to operate only with data from onboard sensors. As a conservative measure in this situation, the robots should behave to maintain the current level of operation, therefore we propose to solve the following problem:

Problem. *Maintain the topology of the directed graph $\mathcal{G}$ describing the system, to provide each robotic agent with the necessary information to allow the safe execution of the global task assigned to the team, despite the absence of communication between the agents.*

The problem presented can be divided into three core aspects:

- Maintaining the topology of the directed graph $\mathcal{G}$: each drone should have the necessary information to execute the assigned (external) task.
- Enforce safe execution: each drone should avoid crashing with other drones.
- Execute the global task: the controller should minimally modify the behavior of the drones such that the constraints are enforced while the objective of the task is being pursued.

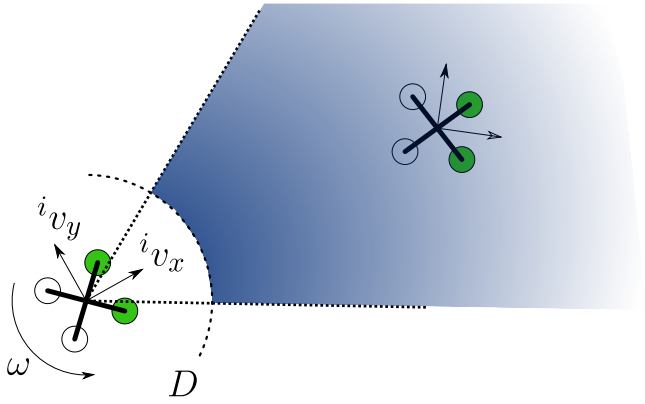


Figure 4.5: Depiction of the safe space where robot i aims to keep the detected drones.

Proposed Solution

Given the considered limitations on the communication between the agents, the problem of maintaining the topology is equivalent to maintaining the current connections between the robots. A connection between the robots depends only on the relative pose of the robots with respect to each other, therefore to maintain a connection each robot should move in order to maintain the currently detected neighbors inside its field of view. We, therefore, propose a CBF-based controller that minimally modifies a desired input velocity to maintain the currently detected robots in the field of view. The CBF is defined starting from the relative position of the detected robots with two separated components, one from the left side of the field of view and one from the right side. Furthermore, to increase the safety of the motion, a third component is added to guarantee the maintenance of a safe distance from the neighbors. The three components are treated as different constraints inserted into an optimization solver. Figure 4.5 displays the area where a robot should keep its neighbors in order to avoid losing the connection and avoid crashing. From the premises, the problem of maintaining the topology of the graph is distributed since each drone is responsible for maintaining its outward edges. If a single drone does not adhere to the behavior, even by simply rotating in place 180°, it can change the topology of the graph removing some edges. Therefore each drone is equally responsible for maintaining the topology of the system, implying that the proposed solution solves Problem 4.1.2 in a distributed fashion, moreover the CBF-constrained controller is formulated as a Quadratic Programming problem with linear constraints, whose solution can be obtained in milliseconds.

4.1.3 Field of View Maintenance

In this section, we detail the formulation of the motion constraint for each drone, in order to maintain its detected neighbors in the camera field of view. In order to maintain the drone j in the field of view of drone i, we propose the addition of a CBF to optimize the velocity input to ensure the feasibility of the constraint. For the problem at hand, we propose the use of a CBF to guarantee the constraint is always satisfied during the motion of the drones. For this purpose, we introduce, for each drone i, the following $h(\cdot)$

function, defined for each neighbor $j \in \mathcal{N}_i$

$$h({}^i\mathbf{p}_j) = \begin{bmatrix} h_1({}^i\mathbf{p}_j) \\ h_2({}^i\mathbf{p}_j) \\ h_3({}^i\mathbf{p}_j) \end{bmatrix} = \begin{bmatrix} \tan(\beta/2)^i x_j + {}^i y_j \\ \tan(\beta/2)^i x_j - {}^i y_j \\ ||{}^i\mathbf{p}_j||^2 - D^2 \end{bmatrix}. \tag{4.2}$$

The first two components $h_1(\cdot)$ and $h_2(\cdot)$ express the left and right border of the field of view as a linear function. The third component expresses the difference between the distance of drone j from drone i with the safe distance $D \in \mathbb{R}_{\geq 0}$. Imposing the condition $h(\cdot) \geq \mathbf{0}$ is equivalent to constraining drone j to be: 1) on the left of the right border, 2) on the right of the left border, 3) at a safe distance from i. This component is equivalent to the CBF proposed in [68], we refer interested readers to the original paper for further details.

The associated optimization problem is formulated as follows. Let ${}^i\mathbf{u}_i = [{}^iv_x, {}^iv_y, {}^i\omega]^\top \in \mathbb{R}^3$ be the vector of input velocities for the i-th drone expressed in its body frame. Let ${}^i\mathbf{u}_i^*$ be the desired velocity for the drone obtained from a top-level distributed algorithm, i.e., the task controller. The following optimization-based controller is employed to ensure the safety condition while modifying the desired velocity in a minimally invasive fashion as

$$\begin{aligned} &\underset{{}^i\mathbf{u}_i \in \mathbb{R}^m}{\operatorname{argmin}} ({}^i\mathbf{u}_i - {}^i\mathbf{u}_i^*)^\top H ({}^i\mathbf{u}_i - {}^i\mathbf{u}_i^*) \\ &\quad \text{s.t. } \dot{h}({}^i\mathbf{p}_j, {}^i\mathbf{u}_i) \geq -\alpha(h({}^i\mathbf{p}_j)) \quad \forall j \in \mathcal{N}_i. \end{aligned} \tag{4.3}$$

The cost function of the optimization is expressed as a quadratic function of the error between the input ${}^i\mathbf{u}_i$ and the desired control action ${}^i\mathbf{u}_i^*$. The positive definite matrix $H \in \mathbb{R}^{3\times 3}$ is inserted in order to assign different weights to specific components of the velocity error. We define the class $\mathcal{K}$ function $\alpha(\cdot)$ as the function

$$\alpha(h({}^i\mathbf{p}_j)) = \begin{bmatrix} \gamma_F h_1({}^i\mathbf{p}_j)^3 \\ \gamma_F h_2({}^i\mathbf{p}_j)^3 \\ \gamma_S h_3({}^i\mathbf{p}_j) \end{bmatrix}, \tag{4.4}$$

where $\gamma_F, \gamma_S \in \mathbb{R}_{\geq 0}$ are positive constants. The use of such $\alpha(\cdot)$ function is motivated by the necessity of slowing the relative motion when the robot approaches the border of the field of view from the inside. The cubic function possesses the same requirements as the linear function but its values are smaller when $h(\cdot)$ is close to zero, with respect to the linear function used for example in h_3. This difference in shape imposes that $\dot{h}$ has to be "less decreasing" when approaching the border with respect to the linear counterpart. This effect dampens the behavior of h allowing for smoother behavior.

The time derivative $\dot{h}(\cdot)$ is obtained as follows. The function $h({}^i\mathbf{p}_j)$ introduced in (4.2)

can be rewritten as

$$h({}^i\mathbf{p}_j) = \begin{bmatrix} \tan(\beta/2) & 1 \\ \tan(\beta/2) & -1 \\ {}^ix_j & {}^iy_j \end{bmatrix} {}^i\mathbf{p}_j - \begin{bmatrix} 0 \\ 0 \\ D^2 \end{bmatrix}. \tag{4.5}$$

The time derivative of this formulation is then expressed as

$$\dot{h}({}^i\mathbf{p}_j, {}^i\mathbf{u}_i) = \frac{\partial h({}^i\mathbf{p}_j)}{\partial {}^i\mathbf{p}_j} {}^i\dot{\mathbf{p}}_j = \begin{bmatrix} \tan(\beta/2) & 1 \\ \tan(\beta/2) & -1 \\ -2{}^ix_j & -2{}^iy_j \end{bmatrix} {}^i\dot{\mathbf{p}}_j. \tag{4.6}$$

The velocity of j with respect to i is then expressed through kinematic computations

$${}^i\dot{\mathbf{p}}_j = {}^iR_w {}^w\dot{\mathbf{p}}_j - \begin{bmatrix} {}^iv_x \\ {}^iv_y \end{bmatrix} + \omega \begin{bmatrix} {}^iy_j \\ -{}^ix_j \end{bmatrix}. \tag{4.7}$$

Since we assume not to have information on the 2D velocity of the neighbor drones, as a conservative approach we assume each neighbor is moving towards the closest border at the maximum allowed velocity. It is then possible to write (4.6) as

$$\dot{h}({}^i\mathbf{p}_j, {}^i\mathbf{u}_i) = \frac{\partial h({}^i\mathbf{p}_j)}{\partial {}^i\mathbf{p}_j} {}^iR_w {}^w\dot{\mathbf{p}}_j + A_c {}^i\mathbf{u}_i \tag{4.8}$$

where

$$A_c = \begin{bmatrix} -\tan(\beta/2) & -1 & \tan(\beta/2){}^iy_j - {}^ix_j \\ -\tan(\beta/2) & 1 & \tan(\beta/2){}^iy_j + {}^ix_j \\ -2{}^ix_j & -2{}^iy_j & 0 \end{bmatrix}. \tag{4.9}$$

The optimization problem in (4.3) therefore becomes

$$\begin{aligned} &\underset{{}^i\mathbf{u}_i \in \mathbb{R}^m}{\operatorname{argmin}} ({}^i\mathbf{u}_i - {}^i\mathbf{u}_i^*)^\top H ({}^i\mathbf{u}_i - {}^i\mathbf{u}_i^*) \\ &\text{s.t. } A_c {}^i\mathbf{u}_i \geq -\alpha(h({}^i\mathbf{p}_j)) - \frac{\partial h({}^i\mathbf{p}_j)}{\partial {}^i\mathbf{p}_j} {}^iR_w {}^w\dot{\mathbf{p}}_j \quad \forall j \in \mathcal{N}_i \end{aligned} \tag{4.10}$$

In this formulation of the optimization problem, the constraint is separated into terms that depend on the decision variable ${}^iu_i$ and terms that depend only on the current configuration. Such formulation facilitates the implementation of the Quadratic Programming problem into a solver to easily and quickly obtain the optimal solution. Enforcing a neighbor drone j to remain in the field of view of i ensures that the existing outgoing edges of the associated graph $\mathcal{G}$ are preserved.

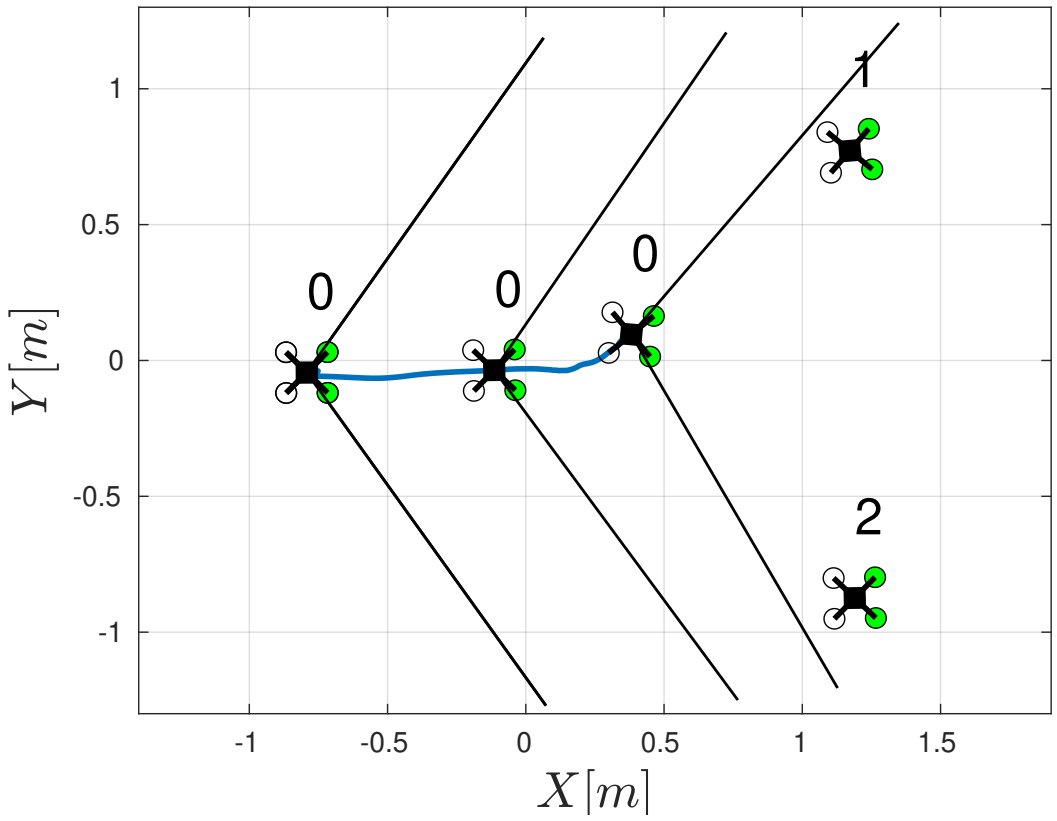


Figure 4.6: 2D plot of the position of the aerial robots while one is tasked to move in between the other two.

4.1.4 Experimental Evaluation

Experimental Setup

We performed experiments with 3 quadrotors deployed in an indoor flying space of $10\times6\times4$ m^3 at the Agile Robotics and Perception Lab (ARPL) lab at New York University. The used aerial platforms, depicted in Fig. 4.1 are custom small-scale aerial robots equipped with a Qualcomm® Snapdragon$^{\text{TM}}$Flight$^{\text{TM}}$ Pro board and on-board VIO, planning, and control based on [96]. The framework has been developed in ROS. Communication among drones is implemented using a synchronized multi-master network module. The optimization problem realizing the CBF is implemented using OSQP [97] and given as input to the low-level control on each robot. The value of the field of view β is set to 100°. The minimum distance D among drones is set to $0.5m$. The maximum velocity magnitude used to compute ${}^i\dot{\mathbf{p}}_j$ is set to $0.1m/s$.

The cost matrix H is a diagonal matrix set to $H = diag(1.0,\ 1.0,\ 0.05)$ in order to inform the optimization solver that the components ${}^iv_x, {}^iv_y$ of ${}^i\mathbf{u}_i$ are more important then ω. The solution to the CBF optimization problem considers only a maximum of 5 neighbor drones. Since the constraint is active only if a neighbor drone reaches the edge of the field of view, potentially occluding other drones, this limitation of the number of considered drones on the 5 with the lowest $h(\cdot)$ allows the solution on board of the problem with a computation time of 0.4 ms on average. In all the following tests, the

parameters γ_F and γ_S are set to 1.0 and 2.0, respectively. The value for γ_F has been empirically tuned to allow the usage of most of the available field of view. The value for γ_S has been selected according to what is presented in [68] to maintain the drones distant enough to not generate any downdraft. The safe distance D is set to $0.5m$. For testing purposes, the high-level coverage control algorithm and the CBF optimization are computed off-board in the test. The detection of the neighbors for each agent is computed off-board using measurements from a motion-capture system that are filtered given the orientation of the considered drone.

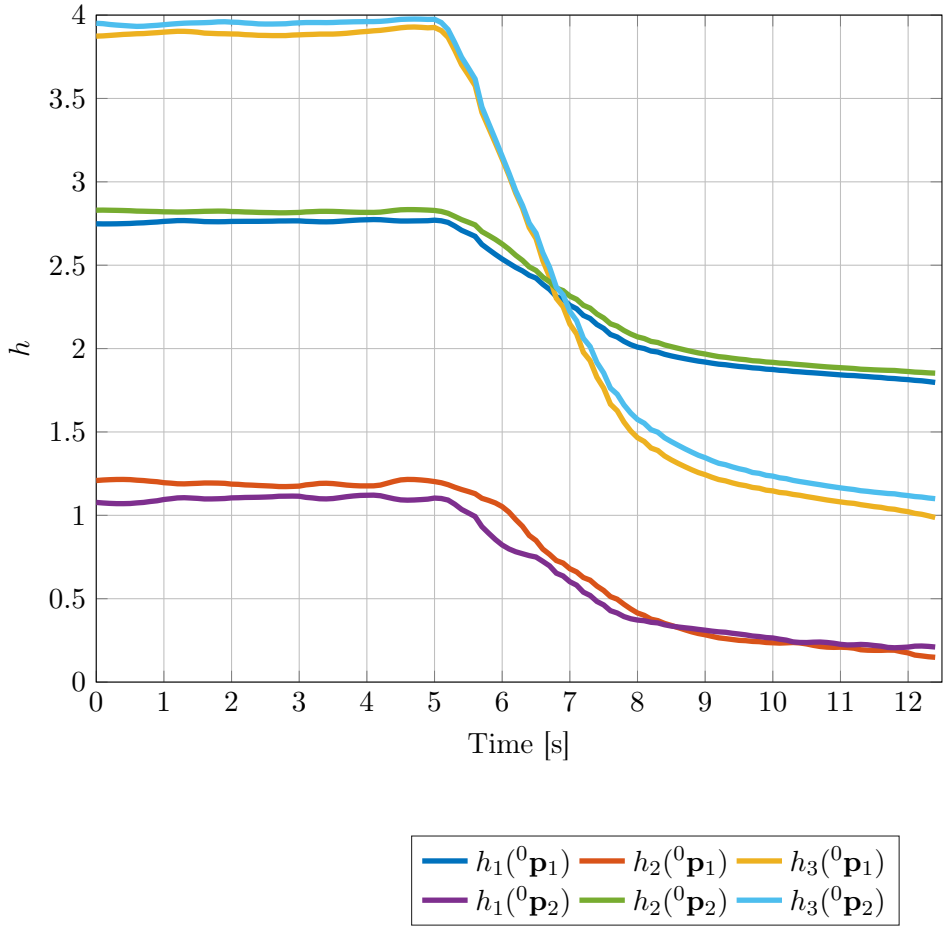


Figure 4.7: Visualization of the values of h during the test. The value is generally maintained greater than 0.

CBF Validation

To validate the proposed formulation we propose the following test. A single aerial platform is tasked to move in between two other drones. Without any restrictions, the drone would simply pass through, ignoring the positions of the other two drones. The introduction of the proposed CBF stops the drone motion maintaining the other two drones in the field of view. Figure 4.6 displays the 2D poses of the 3 drones involved, the drone on the left is tasked to pass in the middle of the drones on the right. The motion is stopped before the neighbors exit the field of view. Figure 4.7 displays the behavior of the components of h of drone 0 with respect to both neighbor drones. It is clearly shown that the neighbors are kept in a safe set and any violation is negligible. This confirms

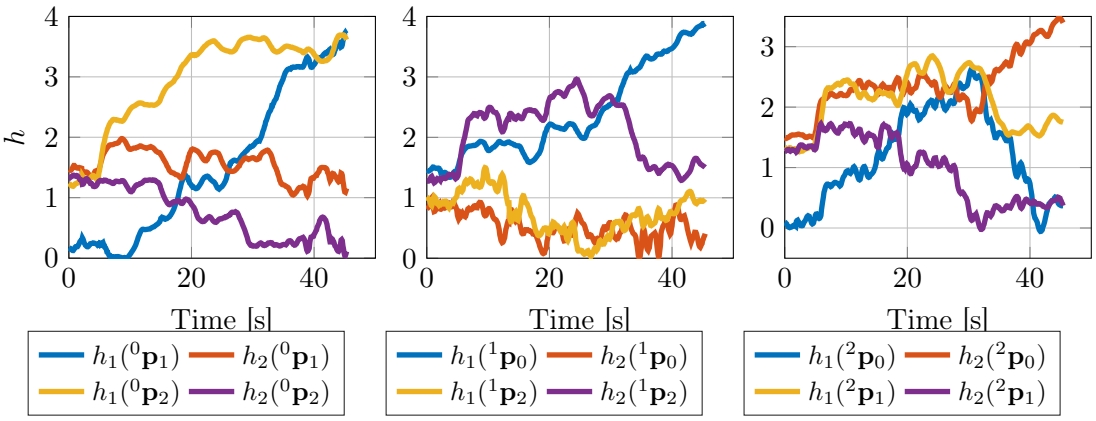


Figure 4.8: Visualization of the values of h for the three robots (drones 0, 1, and 2 are displayed from left to right). Each robot detects two other drones. The left component $h_1(\cdot)$ is visualized in blue and yellow, while the right component $h_2(\cdot)$ is visualized in red and purple.

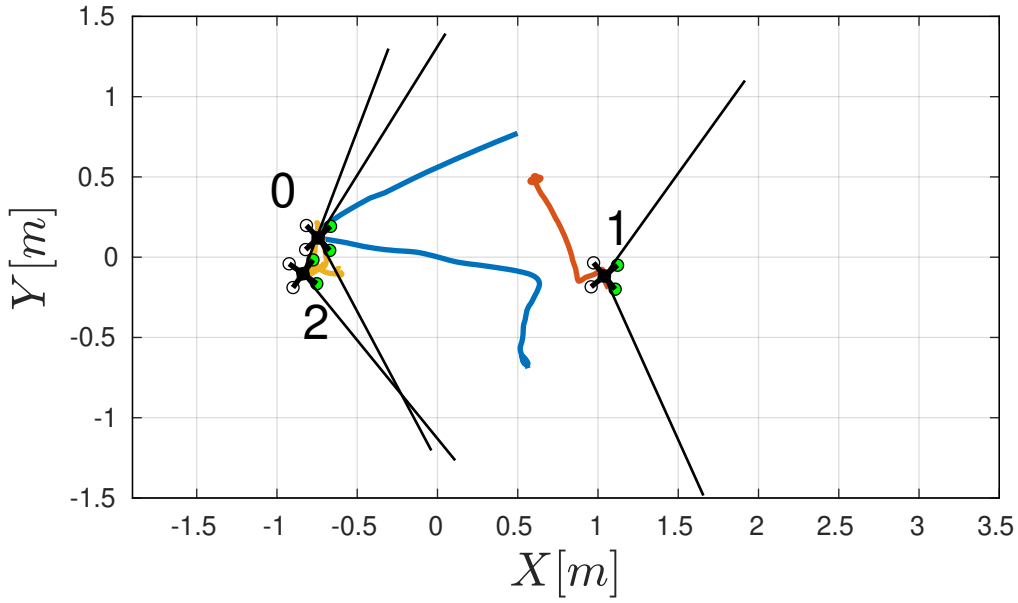


Figure 4.9: Coverage control with a limited field of view. The aerial platforms end up crashing with each other because of the lack of information.

the effect of the CBF on maintaining the current connection between the drones.

Case Study: Coverage Control

The topology maintenance approach proposed in this work can be integrated into a wide range of distributed multi-robot control algorithms. Among many possible options, such as [98, 99, 100, 101], this section considers coverage control as a case study and investigates how anisotropic neighbor detection affects performance and how the proposed approach guarantees the safe execution of the coverage task. The following test performs coverage control in a closed environment using only limited information coming from the field of view of the robot. The algorithm starts from a configuration where each robot can see the other two, generating a complete graph connecting the robots. Figure 4.9 shows

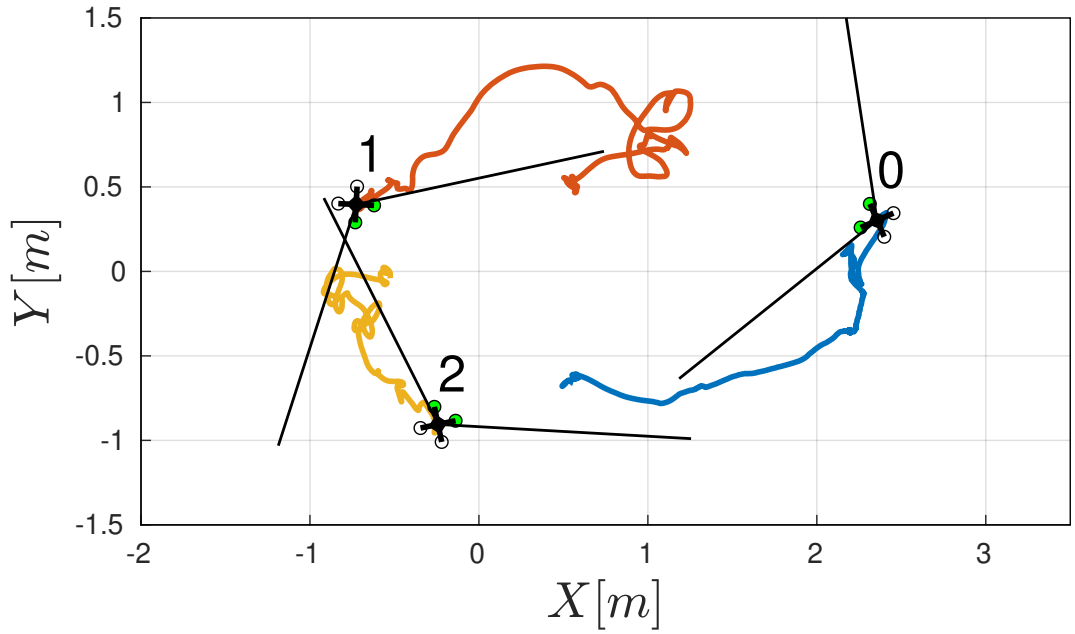


Figure 4.10: Coverage control. The drones start by facing each other and move in the environment according to the motion generated by the control barrier function. The trajectory of drones 0, 1, and 2 are depicted in blue, red, and yellow respectively.

the behavior of the drones without any constraint. It is shown that two aerial robots clash with each other since one robot has no information on the presence of the other, therefore the coverage algorithm instructs the robot to move towards what it considers "free space", causing the accident. Figure 4.10 displays the behavior of the controlled system with the effect of the proposed CBF.

The robots initially spread over the environment until the constraint becomes active. The modified control action generates a motion where the drones start circling in the counter-clockwise direction till they are set in the final displayed position, where every drone has the other two in the field of view. Figure 4.8 displays the behavior of h for the robots throughout the test. The data shows again that the field of view boundaries are enforced by constraining the requested motion. By maintaining the initial information flows on the robot's controller, the safety of the algorithm is improved since the robot knows all the necessary information to avoid crashing with the other robots. It can be noted that the cumulative application of the CBF to the motion of all three aerial robots generates significant oscillations in the values of h when the robot is inside the safe set. These oscillations are greatly reduced when the detected robot approaches a border of the field of view, due to the effect of the CBF.

4.1.5 Conclusion

In this section, we proposed a novel control formulation to maintain the topology of the directed graph describing the connection between agents in a multi-robot system. Specifically, we explored the case where the agents have a limited field of view due to local sensing and propose a CBF-based optimization controller that modifies a generic

desired input in a minimally invasive fashion. The CBF is formulated in order to enforce the maintenance of the current neighbors inside the field of view. The validity of the CBF is tested using aerial platforms by performing a scenario where the desired input purposely directs platforms toward an undesirable configuration. The data shows how the control is capable of stopping the undesired motion. The control layer is then inserted in the coverage control scenario where the robots monitor an area. The provided data shows the effectiveness of the proposed solution on the maintenance of the current connections and therefore improving the safe and proper execution of the coverage task.

In this work, we assume not to have information on the neighbors' velocity, therefore, we considered the robots moving toward the closest border as part of the CBF formulation. This assumption, alongside the assumption on the bounded speed and planar motion of the aerial platform, limits the performance at which the team of robots can accomplish the required task. Future work will focus on relaxing these assumptions by incorporating the whole system dynamics in order to increase the range of applications of the proposed controller. Furthermore, since the current validation is based on data from a motion capture system, we will focus on extending the proposed solution using only onboard robot sensor data.

4.2 Distributed Control of a Limited Angular Field-of-View Multi-Robot System in Communication-Denied Scenarios: A Probabilistic Approach

Building upon our prior work that guaranteed coordination by strictly enforcing visual contact, this section introduces a more flexible control framework for multi-robot systems. While effective, the previous approach could be overly conservative, limiting the agents' ability to operate efficiently. Real-world deployments in fields like search and rescue or surveillance are often constrained by limited fields of view and communication denials, making rigid network topologies impractical. In this work, we relax the constant connectivity requirement and address the challenge of temporarily losing neighbors from sight. Our solution integrates a particle filter to probabilistically track the location of out-of-view agents, managing the associated uncertainty. This is combined with a Control Barrier Function framework that arbitrates the exploration-exploitation dilemma, allowing robots to balance their primary mission with information-seeking actions to re-establish lost connections. We validated this approach in a simulated coverage task, analyzing performance across varying team sizes. Results show that despite the added challenge of dynamic topology, the system successfully completes the mission and achieves a final configuration comparable to an ideal, fully-connected scenario, demonstrating a significant advancement in robust, real-world coordination.

4.2.1 Introduction

One major challenge limiting the adoption of multi-robot systems is dealing with non-ideal sensors. For instance, using cameras with a restricted angular field of view makes it difficult to perceive neighbors and coordinate effectively. Studies investigating anisotropic sensory have been made when applied to environmental perception [102], while communication among the agents is usually assumed to be isotropic. Limitations on communication capabilities lead to even more challenging problems in deploying a robotic team for real-world operations. This challenge is usually addressed only considering robots with a limited communication range designed as a sphere around the sensor, without taking into consideration other fundamental aspects such as the antenna radiation pattern, which is not isotropic, dropped packets, and delays. Moreover, sharing more information increases both hardware and software complexity, thus leading to scalability limitations [32]. In addition, extreme conditions may occur where communication is completely denied by environmental constraints, as in case of underwater or subterranean operations, or artificially jammed, e.g. in military missions.

The contribution of this work is the definition of an optimization-based visual control for decentralized multi-robot systems. In particular, a probabilistic strategy is defined, in order to face the exploration-exploitation trade-off, i.e., choosing whether to search for neighbors and gaining information on their position or to exploit information to pursue the goal of the mission. A properly defined particle filter will be designed with the aim of

gaining information on the location of each neighbor and the related uncertainty. Subsequently, Control Barrier Functions (CBFs) will be designed constraining the motion of the controlled robot to keep a target neighbor inside its own field of view. Finally, slack variables will be used to handle the exploration-exploitation dilemma, softening CBFs constraints, thus allowing the robot to lose visual contact with the target in order to pursue the global goal of the task. The proposed methodology is tested when applied to coverage control of a team of quadrotors in simulations.

4.2.2 Related Works

Only few research works can be found in the literature going as far as to investigate such extreme conditions. A solution is proposed in [103] for flocking control of unmanned aerial vehicles (UAVs). The team is characterized as a continuous fluid and hydrodynamics laws are exploited to guarantee swarm cohesion, collision avoidance, and velocity consensus while taking into account latency within inter-agent communication and constraints resulting from the 3D antenna radiation pattern. Other interesting approaches make use of probabilistic information about the position of undetected neighbors, like Bayes multi-target filters and Probability Hypothesis Density (PHD) filters. A Gaussian Mixture PHD is proposed in [104] to build a localization system estimating positions of robots even when communication is lost for a long duration, evaluating the motion model of the system and selecting places where lost robots are more likely to be at each moment. Another probabilistic approach is presented in [105], where a collision avoidance method is proposed, accounting for both measurement and motion uncertainty. The authors exploit Probabilistic Safety Barrier Certificates (PrSBC) using Control Barrier Functions to define the space of probabilistically safe control actions.
Our work will combine CBFs with probabilistic analysis in order to coordinate the team and complete the mission without collisions among the agents.

4.2.3 Problem Description

Consider a team of N robots moving in a 2D obstacle-free bounded environment tasked with a coverage mission. The robots navigate only exploiting locally available information from an on-board camera and can not communicate with each other. We adopt the same notation framework introduced in Section 4.1.2 and visualized in Fig. 4.2 and Fig. 4.4, where the coordinate frames and geometric relationships are defined. Odometry data is not available for self-localization, but we assume robots are able to calculate their control input locally from a probability density function defining a target area to be reached in order to accomplish the global task. The above conditions can indistinctly apply to ground robots and drones assumed to fly at the same constant altitude and moving slowly enough that their pitch and roll angles are negligible. For this reason, we will mainly consider aerial robots in the rest of the section, but the theoretical discussion can equally be applied to ground robots. We assume each robot can detect the position of neighbors that are in the field of view of its camera. The field of view can be expressed as a limited angular sector centered at local X axis with amplitude β and radius $R_d \in \mathbb{R}_{>0}$,

as visualized in Fig. 4.4. We will indicate the area corresponding to the field of view of robot i as $F_i \subset \mathbb{R}^2$. Considering the i-th robot, the collection of all the other $N-1$ robots in the team will be indicated as $\mathcal{N}_i$. The 2D position of the neighbor j with respect to Σ_i is expressed as ${}^i\mathbf{p}_j = [{}^i x_j, {}^i y_j]^T$ (see Fig. 4.4). Furthermore, it is assumed that each robot has a unique ID and can visually recognize the ID of every detected neighbor. According to [106], ID recognition is possible when UAVs are equipped with blinking UV lights and a specific blinking frequency is assigned to each ID.
The state of each robot i is denoted with ${}^w\boldsymbol{\chi}_i = [{}^w x_i, {}^w y_i, {}^w \theta_i]^T$, where ${}^w x_i, {}^w y_i \in \mathbb{R}$ represent the coordinates of the 2D position of Σ_i in Σ_w while ${}^w\theta_i$ is the yaw angle. In case of a team of aerial robots, the altitude is excluded since we assume every robot flies at the same constant altitude. The robot is controlled according to the kinematic model ${}^i\dot{\chi}_i = {}^i\mathbf{u}_i = [{}^i v_x, {}^i v_y, {}^i\omega]^T$ where ${}^i v_x, {}^i v_y, w \in \mathbb{R}$ represent the linear and angular velocity input for the agent in Σ_i.

4.2.4 Neighbors' Position Estimation

In our work, we make use of particle filters to estimate the 2D position of neighbors. More in details, a particle filter is a recursive Bayesian state estimator that uses discrete particles to approximate the posterior distribution of an estimated state. Its algorithm is useful for online state estimation of a non-linear system according to the dynamic model of the robot and measurements taken (see [107]). Process and measurement noise distribution are also taken into account, resulting in the definition of a probability distribution of the real robot's state. The procedure for particle filter state estimation is presented in details in [108].

Multiple Particle Filters with Particles Deletion

In order to have an estimate on the position of its neighbors, each robot is required to run onboard a particle filtering algorithm for every other robot j in the team. As previously stated, we assume every robot has a unique ID, and this ID can be recognized by other agents when detected, therefore no effort regarding association between a detected drone and state estimates is required. Since no information can be obtained from the i-th agent for robots lying outside its field of view F_i, we make use of an additional intermediate step to improve the estimation accuracy, while the other steps are the same as the traditional procedure.

0. *Initialization*: A specified number of particles $P \in \mathbb{N}$ is generated according to a known distribution or uniformly distributed within the environment. Each particle represents an hypothesis of the state variables.

$$\chi(0)_k \sim p\left(\chi\left(0\right)\right) \tag{4.11}$$

In the above equation, $\chi(0)_k$ is the initial state of the generic particle k, $p\left(\chi\left(0\right)\right)$ is the initial distribution and the operator $\sim$ is used to denote that the particle is randomly obtained from the probability distribution.

1. *Prediction*: The state of each particle is propagated following the state transition model of the system $f\left(\chi\left(t-1\right)_k, u(t)\right)$, where $u(t)$ is the control input.The result is a new particles distribution.

$$\chi(t)_k = f\left(\chi\left(t-1\right)_k, u(t)\right) \tag{4.12}$$

   Since the kinematic model requires a velocity input, an estimate is provided according to the common goal of the team. Otherwise, if an estimate on the input of drone j can not be derived, the worst case scenario is considered, i.e. robot j is moving towards robot i at maximum speed.

2. *Weight Update*: The likelihood of sensor measurements $\gamma(t)$ is exploited to update the weight of each particle.

$$w(t)_k = p\left(\gamma(t)|\chi(t)_k\right) \tag{4.13}$$

   The measurement uncertainty was considered as a matrix $\Sigma_{MEAS} \in \mathbb{R}^{3\times3}$, which was assumed to be independent from the position of the robot inside the field of view in order to simplify the discussion.

3. *Particles Deletion*: In case robot i is not able to detect j, the only correction that can be made is deleting samples inside robot i's field of view F_i, since j would be detected if it was there. Particles inside the field of view are removed by assigning them a null weight:

$$w(t)_k = 0 \quad \text{if } j \notin F_i \text{ and } k \in F_i \tag{4.14}$$

4. *Resampling*: The particles are resampled according to their weights, in order to give more weight to particles that are more likely to match observed data. This means that the new set of particles will be more concentrated in regions of the state space that have the highest probability, and no samples will be placed inside F_i thanks to the particles deletion step.

5. *State Estimation*: The state estimate $\hat{\chi}(t)$ is calculated as a weighted sum of particles.

$$\hat{\chi}(t) = \sum_{k=1}^{P} \frac{w(t)_k \chi(t)_k}{w(t)_k} \tag{4.15}$$

   It is interesting to note that the particle filter provides an estimate of both position and orientation, even if our control strategy will only exploit the position estimate. Orientation can be helpful in some situations, but for that to happen, measurements must have accurate information about it, which visual sensors may lack.

In Fig. 4.11, drone 0 visually gets drone 1's relative position, but can only estimate the location of undetected neighbors.

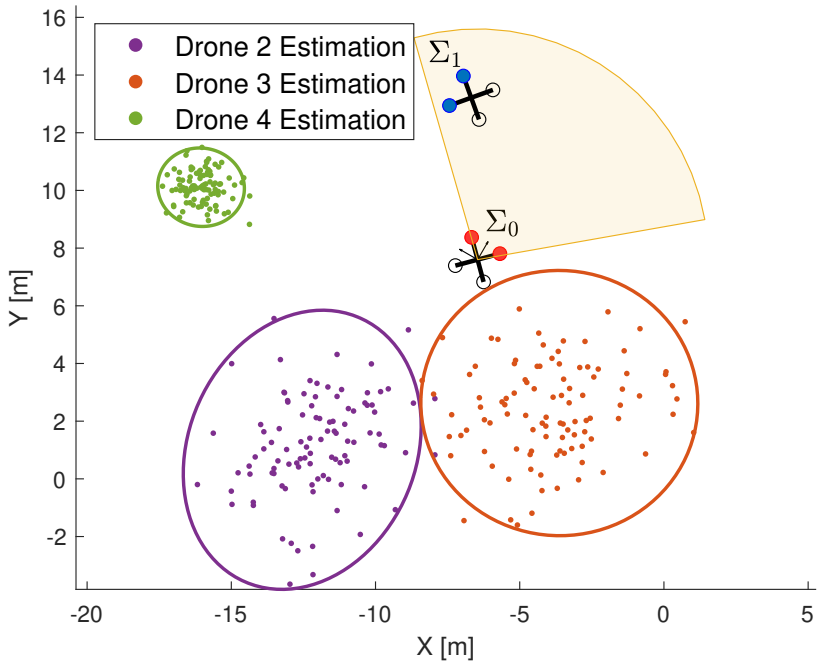


Figure 4.11: 95% Confidence Ellipses of undetected neighbors.

Samples Budget Re-Allocation

The adoption of several particle filters raises the problem of computational effort required to each robot, and makes the solution not scalable since a larger number of robots leads to the implementation of more filters. Our solution to overcome this problem and achieve a scalable architecture is defining a fixed total budget of samples P_T that have to be allocated to the neighbors at each time instant.

Let us assume, for now, that we can define a relative importance among the neighbors of robot i, and therefore we can sort them from the most to the least important. The importance parameter will be discussed in Section 4.2.5. The key idea is to assign more particles to neighbors we are more interested in tracking based on the aforementioned importance, resulting in a more accurate evaluation. Conversely, the state estimation of less important robots can be performed with lower accuracy, so fewer particles are required. Therefore, after sorting neighbors, the number of required particles P_j is calculated according to their relative importance as a fraction of the total budget of particles P_T. Then, another resampling step is performed in order to draw a new samples set with P_j elements, without modifying the current probability distribution. In this way, the evaluation on neighbors' position can be considered scalable and only depends on P_T, regardless of the number of robots in the team.

The main motivation for adopting particle filters lies in their ability to discard particles within the field of view and propagate only those outside it, thereby naturally handling the resulting non-Gaussian belief. Additionally, particle filters allow to maintain a fixed total number of samples, which provides a controlled computational complexity.

4.2.5 Uncertainty Evaluation

In the following, particles distribution will be investigated to gain information on the probability associated with the location of an undetected neighbor. Since our focus is on the 2D relative position of drone j with respect to Σ_i, namely ${}^i\mathbf{p}_j = [{}^i x_j, {}^i y_j]^T$, particles distribution can be addressed as a bivariate normal distribution, whose key parameters are the mean point ${}^i\overline{\mu}_j \in \mathbb{R}^2$ and the covariance matrix ${}^i\overline{\Sigma}_j \in \mathbb{R}^{2\times 2}$. Approximating the particles distribution as a Gaussian distribution is needed to lower the required computational effort. As a matter of fact, fitting the samples set to a non-uniform or a Gaussian mixture distribution would make the computational burden too heavy to be carried out by a generally low performance platform as an aerial robot. Moreover, considering particles as normally distributed reflects reality when all particles are placed outside drone i's field of view, therefore none of them is deleted. The distribution becomes non-uniform, instead, when some particles are deleted, but we can consider them as a small fraction of the samples set, thus not influencing the overall distribution significantly. Furthermore, a confidence interval can be derived from the distribution according to a desired confidence level drawn as an ellipse. Therefore, the real position of the undetected robot will be located inside the ellipse with the desired confidence level. Let us consider a generic ellipse defined by

$$\left(\frac{x - x_C}{a}\right)^2 + \left(\frac{y - y_C}{b}\right)^2 = \alpha \tag{4.16}$$

where $q_C = [x_C, y_C]^T$ is the central point, $a, b \in \mathbb{R}_{>0}$ are the major and minor semi-axis respectively, and $\alpha \in \mathbb{R}_{>0}$ is the scale of the ellipse. Since we are considering particles sampled from a multivariate Gaussian distribution, particles' x-coordinates and y-coordinates are normally distributed too. Therefore, the left hand side of (4.16) represents the sum of squares of normally distributed samples, also known as a Chi-Squared distribution. This allows us to evaluate the Chi-Squared likelihood to define the scale α of the ellipse, according to Pearson's Chi-Squared test [109]. Parameters defining the confidence ellipse can be easily calculated from the mean point and the covariance matrix. In particular, the center q_C corresponds to the mean point $\overline{\mu}$, and major and minor semi-axis, a and b respectively, can be calculated from the eigenvalues of the covariance matrix as $a = \sqrt{\alpha\lambda_1}$ and $b = \sqrt{\alpha\lambda_2}$, where $\lambda_1, \lambda_2 \in \mathbb{R}$, with $\lambda_1 \geq \lambda_2$, are the eigenvalues of the covariance matrix. Finally, the inclination of the major axis with respect to the x-axis is calculated as

$$\psi = \arctan \frac{v_1(y)}{v_1(x)} \tag{4.17}$$

where $v_1(x), v_1(y) \in \mathbb{R}$ are the x and y components of the eigenvector $\mathbf{v}_1$ corresponding to the largest eigenvalue λ_1, respectively. The calculated ellipses are shown in Fig. 4.11. The distance $d_{ij} \in \mathbb{R}_{>0}$ between robot i and the elliptical area related to each neighbor j is measured to evaluate the relative importance of each robot as mentioned in Section 4.2.4. To calculate the distance from robot i to j-th ellipse, let us consider the parametric version

of (4.16), generalized to a rotated ellipse:

$$\begin{aligned} {}^{i}x &= {}^{i}(\overline{\mu}_x)_j + a_j \cos\left({}^{i}\psi_j\right)\cos\theta - b_j \sin\left({}^{i}\psi_j\right)\sin\theta \\ {}^{i}y &= {}^{i}\left(\overline{\mu}_y\right)_j + a_j \sin\left({}^{i}\psi_j\right)\cos\theta + b_j \cos\left({}^{i}\psi_j\right)\sin\theta \end{aligned} \tag{4.18}$$

The nearest point to robot i lying on j-th ellipse, namely ${}^{i}p_j^{cr}$, can be obtained from the inclination of the line linking j to i:

$$\theta_j^{cr} = \arctan \frac{{}^{i}x_j}{{}^{i}y_j}. \tag{4.19}$$

Substituting (4.19) in (4.18), we obtain coordinates of ${}^{i}p_j^{cr}$. Finally, d_{ij} is calculated as the Euclidean distance between p_i and p_j^{cr}:

$$d_{ij} = \|{}^{i}p_j^{cr} - {}^{i}p_i\| \tag{4.20}$$

The distance d_{ij} will also be exploited in the next sections as a trade-off parameter for the exploration-exploitation problem, defining the behavior of the controlled robot to search for undetected neighbors or to fulfill the global task.

4.2.6 CBFs With Field of View Constraints

Here, we leverage the Control Barrier Function (CBF) framework introduced in Section 2.2 to constrain the motion of a controlled robot, ensuring a target neighbor remains within its field of view. Specifically, we define a safe set $\mathcal{C}$ representing the configurations where the target is visible. The control input is then computed by solving the optimization-based controller formulation (2.13), which minimally modifies the desired input $\mathbf{u}^*$ to satisfy the safety constraints.

The problem of balancing between exploration and exploitation is addressed by softening or hardening these constraints, effectively determining whether the agent must strictly adhere to the safe set or is permitted to exit it. To stabilize the system and guide the robot back into the safe set when necessary, we employ the Control Lyapunov Function (CLF) methodology, also detailed in Section 2.2, making the equilibrium set equivalent to the safe set $\mathcal{C}$.

Field of View Constraints

The set of constraint equations forcing a robot to keep a target neighbor inside its field of view is derived from the previous section 4.1.3 with the addition of a further constraint on the maximum distance, and only depends on the 2D relative position of robot j, namely

${}^i\mathbf{p}_j$. In particular:

$$h({}^i\mathbf{p}_j) = \begin{bmatrix} \tan(\beta/2) & 1 \\ \tan(\beta/2) & -1 \\ {}^i x_j & {}^i y_j \\ -{}^i x_j & -{}^i y_j \end{bmatrix} {}^i\mathbf{p}_j + \begin{bmatrix} 0 \\ 0 \\ -D_s^2 \\ R_d^2 \end{bmatrix}. \tag{4.21}$$

Considerations on how we can derive a control input for robot i go out of the scope of this work, therefore we refer the reader to Section 4.1.3 for a detailed explanation on the formulation of the optimization problem (4.3) from the constraint equations (4.21).
It is important to highlight that the chosen CBF (4.21) can also force the controlled robot to go back into the safe set if outside. This means that the input $\mathbf{u}$ resulting from the QP problem will carry the agent inside the set $\mathcal{C}$, i.e. in a configuration where robot j lies inside F_i.

Theorem 1. *Let $V(\chi) : \mathbb{R}^m \to \mathbb{R}$ be:*

$$V(\chi) = \begin{cases} 0 & \text{if } \chi \in \mathcal{C} \\ -h(\cdot) & \text{if } \chi \notin \mathcal{C} \end{cases} \tag{4.22}$$

Consider the CBF $h({}^i\mathbf{p}_j)$ defined in (4.21)*. Then, $V(\chi)$ is a CLF for the system.*

Proof. The candidate CLF (4.22) is continuously differentiable and positive definite by construction. We should demonstrate that the derivative $\dot{V}(\chi)$ is negative definite or semidefinite. Let us focus on the case $\chi \notin \mathcal{C}$:

$$\dot{V}(\chi) = -\dot{h}(\chi) \leq \alpha(h(\chi)) \tag{4.23}$$

From Section 4.1 we have $\alpha(h(\chi)) = \lambda h(\chi)$, where $\lambda \in \mathbb{R}_{\geq 0}$ is a positive constant. Since we know that $h(\chi) < 0$ if $\chi \notin \mathcal{C}$, we can combine the chain of inequalities and finally get

$$\dot{V}(\chi) < 0 \tag{4.24}$$

which ensures that $V(\chi)$ is a CLF and asymptotically stabilizes the set $\mathcal{C}$. □

Slack Variables for Exploration-Exploitation

The task of keeping every robot $j \in \mathcal{N}_i$ inside F_i can potentially become impossible to solve, especially in case of teams with a high number of robots. Moreover, even if this condition could be satisfied, the set of feasible control actions would become very restricted and achieving the global goal of the mission could be highly complicated. For this reason, we face this problem as an exploration-exploitation trade-off, where robot i is required to search for j only when the distance to the j-th ellipse is low, indicating that a collision could potentially happen. Information on the estimated position and uncertainty ellipses are gained from the particle filter as described in Section 4.2.5.
The implementation of this solution is carried out integrating slack variables into the

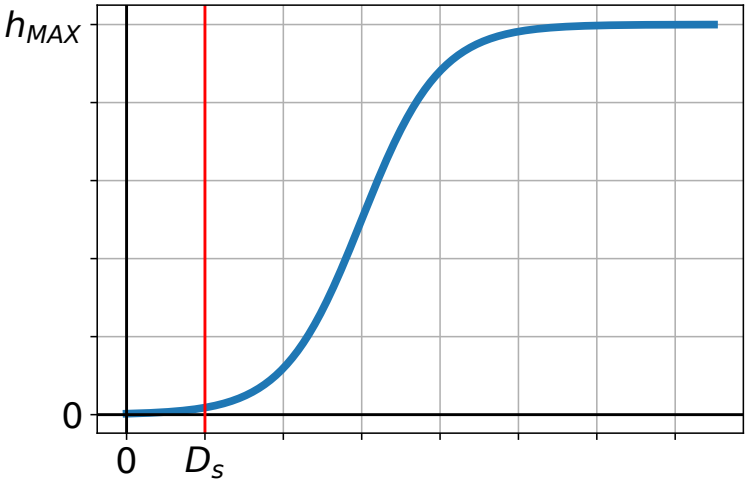


Figure 4.12: Value of the slack variable ϵ

optimization problem, softening CBFs constraints related to neighbors placed far away from i, therefore there is no danger of collision. We define the slack variable $\epsilon(d_{ij}) \in \mathbb{R}_{\geq 0}$ as a sigmoid function of the distance d_{ij} calculated in (4.20). Consider the following sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{4.25}$$

We define the slack variable as $\epsilon(d_{ij}) = \sigma(d_{ij} - 3D_s)$ where D_s is the safety distance. As we can see from Fig. 4.12, $\epsilon \to 0$ as d_{ij} decreases to D_s, while assuming a high value when the distance is greater and there is no danger of collision. In addition, the slope of the function contributes to balancing exploration and exploitation, assigning more weight to exploitation as the slope increases. The set of saturation values h_{MAX} is defined as the modulus of the lowest value that can be assumed by (4.21) according to the environment size, except for the safety constraint defined by the third row of $h({}^i\mathbf{p}_j)$, which is always kept as a hard constraint in order to ensure collision avoidance, thus $\epsilon_3 = 0$.
Inserting slack variables into the optimization problem (4.10), it becomes:

$$\begin{aligned} \mathbf{u}(\chi) = \underset{\mathbf{u}\in\mathbb{R}^m}{\text{argmin}} \quad & \frac{1}{2}\|\mathbf{u} - \mathbf{u}^*\|^2 \\ \text{s.t.} \quad & \dot{h}(\chi, \mathbf{u}) + \epsilon \geq -\alpha(h(\chi)). \end{aligned} \tag{4.26}$$

Thus, the integration of ϵ has the effect of softening constraints related to robot j, therefore i can move towards the goal of the global mission when d_{ij} is safe $\forall j \in \mathcal{N}_i$. Otherwise, when the j-th ellipse indicates a possibly dangerous position, the j-th set of constraints becomes hard and forces the robot back into the safe set thanks to Theorem 1, in order to gain information on j-th position and increase the accuracy of the particle filter.

Hierarchical Optimization

One possible drawback of stacking several inequality constraints into the optimization problem (4.26) could be infeasibility. As a matter of fact, especially in case of large teams, robot i could be in a condition where the distance to more than one ellipses is critical, thus having several hard constraints in the optimization problem. To solve this issue, we assign a priority to each set of constraints in the same way as we sorted neighbors for the particles budget re-assignment in Section 4.2.4, i.e. based on the distance d_{ij} to the j-th ellipse calculated in (4.20). The adopted approach to hierarchically solve the optimization problem is the one proposed in [110], whose goal is to hierarchically minimize the violation $\|\kappa_j\|$ of the j-th level of constraints. More in details, a cascade of QP problems is performed, starting from minimizing $\|\kappa_1\|$ which gives an optimal value κ_1^*, then proceeding in minimizing the violation of the following levels of constraints:

$$\begin{aligned} \kappa_j = \underset{\chi,\mathbf{u}}{\operatorname{argmin}} \quad & \|\kappa_j\|^2 \\ \text{s.t.} \quad & \dot{h}(\chi,\mathbf{u}) + \kappa_1^* \geq -\alpha(h(\chi)) \\ & \vdots \\ & \dot{h}(\chi,\mathbf{u}) + \kappa_{j-1}^* \geq -\alpha(h(\chi)) \\ & \dot{h}(\chi,\mathbf{u}) + \kappa_j \geq -\alpha(h(\chi)) \end{aligned} \tag{4.27}$$

This formulation ensures that, for each new level of constraints, the new value κ_j will not affect the prior constraint levels and therefore ensures a strict hierarchy.
The calculated values of minimized violations $\kappa = [\kappa_1, \ldots, k_{N-1}]^T$ is then inserted into (4.26) as a further set of slack variables. In conclusion, the main optimization problem becomes:

$$\begin{aligned} \mathbf{u}(\chi) = \underset{\mathbf{u}\in\mathbb{R}^m}{\operatorname{argmin}} & \frac{1}{2}\|\mathbf{u}-\mathbf{u}^*\|^2 + \frac{1}{2}\|\kappa\|^2 \\ \text{s.t.} & \dot{h}(\chi,\mathbf{u}) + \epsilon + \kappa \geq -\alpha(h(\chi)). \end{aligned} \tag{4.28}$$

The integration of κ into the optimization problem affects the robot's behavior enforcing it to hierarchically search for undetected neighbors, aiming at first detecting neighbors who are expected to be in more dangerous positions.

4.2.7 Experimental Evaluation

Here, we present an application of our solution in the widely studied use case of coverage control. Then, we apply our methodology to coverage control of a team of aerial robots in a virtual environment and compare our results with those obtained employing robots equipped with isotropic sensors and ideal communication and we analyze how the number of robots in the team affects the performances of our solution and check its capability of handling large teams. Finally, we test the effectiveness of our solution when the team is instructed to follow intersecting trajectories. Robots are considered to be equipped with two kinds of sensors. The first one is an anisotropic sensor and is exploited to detect other

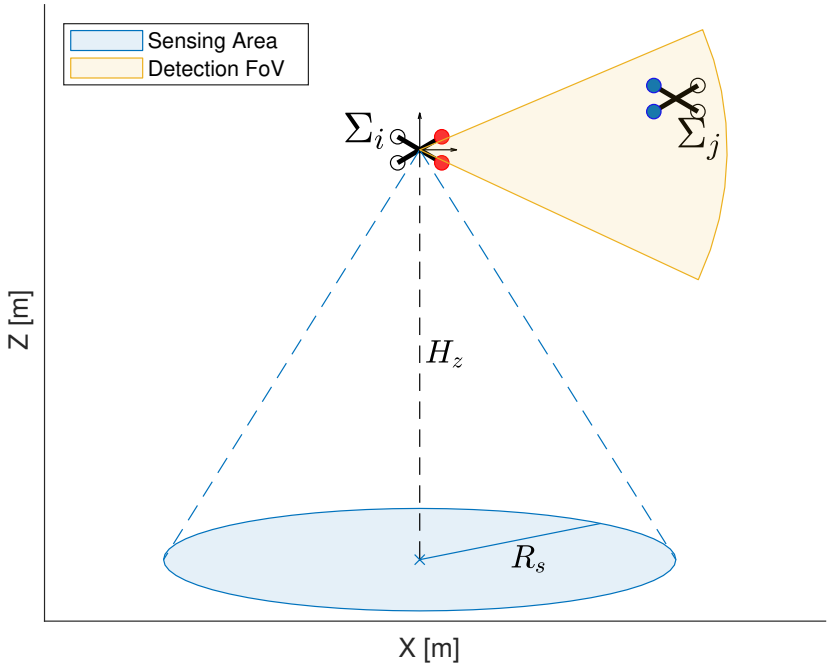


Figure 4.13: Setup for experimental evaluation.

robots. Its limited field of view corresponds to the aforementioned angular sector F_i. The second one, instead, is an isotropic sensor exploited for retrieving information from the environment. Its sensing area is considered to be a circle of radius equal to $R_s \in \mathbb{R}_{>0}$ centered in robot i's position. A graphical representation is provided in Fig. 4.13 for an aerial robot.

First Case Study: Coverage Control

The proposed probabilistic approach can be employed along a wide range of decentralized control algorithms. As a case study we investigate how the presented solution can guarantee the safe execution of the coverage task described in Section 2.1. In our work, we make use of estimated positions to locally build a rough approximation of the real Voronoi partitioning of the environment based on information from particle filters. The desired control action $\mathbf{u}^*$ is calculated with the coverage control algorithm as described in Section 2.1 with the only difference of the angular velocity calculated to face the centroid of its Voronoi cell, in order to move the robot towards the centroid while rotating along the z-axis until facing the target location. Finally, the desired velocity is inserted into the optimization problem (4.28). It is interesting to note that self-localization is not strictly necessary to control the team, since all the data needed to calculate the control input for the i-th robot is expressed relative to Σ_i.

- **Comparison with Ideal Scenario**

  Tests were performed in virtual environments. We employed a team of 6 aerial robots in the RotorS Gazebo simulator framework [111], controlled exploiting the ROS middleware. Drones were considered to fly at a constant altitude $H_z = 3.5$ m sensing a circular area with radius $R_s = 6$ m on the ground thanks to a camera facing downwards with a 120° field of view amplitude. Their maximum linear

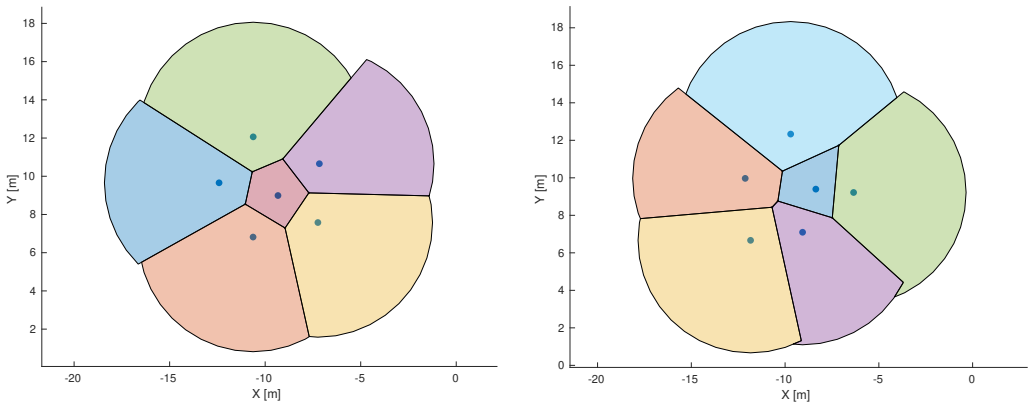


Figure 4.14: Comparison between final configuration reached when performing coverage control with isotropic (left) and anisotropic (right) sensors adopting the proposed solution.

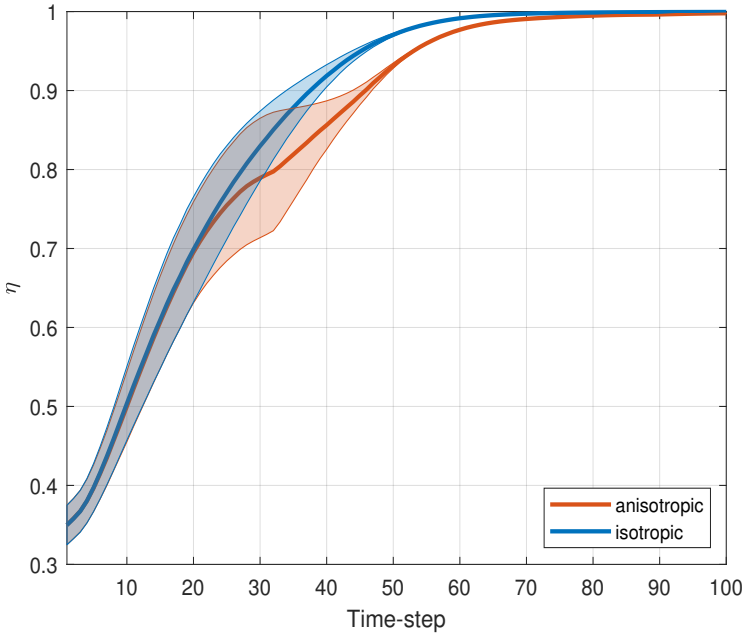


Figure 4.15: Comparison in coverage performance.

velocity is limited to 1.0 m/s, thus their roll and pitch angles can be neglected. Detection and relative localization of neighbors was emulated communicating to the i-th drone the relative position of j only when $j \in F_i$, with F_i defined as the angular sector of radius $R_d = 10$ m and amplitude $\beta = 120°$. Measurement uncertainty was modeled as a diagonal matrix $\Sigma_{MEAS} = diag(0.1, 0.1, 0.5)$, indicating a uniform and accurate measure of the position, but a lower accuracy on the orientation. A safety distance $D_s = 2.0$ m was defined for the safety constraints of the CBFs. We assumed every robot knows the starting position of each one of its neighbors. The team was tasked with a coverage operation in an obstacle-free large-scale environment of 40×40 m^2. A probability function $\phi(q)$ was defined as a Gaussian distribution with mean point $q_m \in \mathbb{R}^2$, highlighting the higher importance of areas near q_m. The control input for robot j needed in the prediction step (4.12) of the particle filter was estimated by robot i as a proportional law driving j towards q_m:

$$ {}^i\mathbf{u}_j^{est} = -K_p \left({}^ip_j - {}^iq_m \right) \tag{4.29} $$

The chosen control input roughly represents the expected behavior of robot j, which, according to the coverage control strategy, should move towards the area of interest whose mean point is ${}^iq_m$. Since ${}^i\mathbf{u}_j^{est}$ will not be accurate, the uncertainty on prediction will be modeled accordingly in order to exploit probabilistic analysis to avoid collisions. The aim of this first set of experiments was to compare performances in terms of covered surface and time required to accomplish the mission with respect to the ideal scenario of robots equipped with isotropic sensors.
The effectiveness was evaluated running 5 sets of simulations comparing performances of our solution with the ideal scenario of a coverage operation employing isotropic sensors. In particular, for each run, a random starting position was generated for each drone, then the mission was performed a first time with $\beta = 120°$ and afterwards with $\beta = 360°$. In both cases, log data was retrieved about the time required to reach the final configuration and area coverage efficiency $\eta \in [0, 1]$ calculated as:

$$ \eta = \int_A \phi(q)\, dq \tag{4.30} $$

where $A \subset \mathbb{R}^2$ is the total surface covered by the team as represented in Fig. 4.14. First of all, results demonstrated that the robotic team successfully accomplished the task without any collision being reported. Furthermore, results shown in Fig. 4.15 show that in both cases the team reaches a configuration where $\eta = 1$, i.e. the area of most interest according to $\phi(q)$ is completely covered. In addition, the time required to reach a value of $\eta = 0.99$ was evaluated, in order to have quantitative data indicating how much slower the fulfillment of the task was. Results show an average increase of 18.6% in task duration when adopting our strategy with anisotropic sensors in comparison with the ideal case of isotropic sensors.
From the obtained results we can state that our methodology can be successfully employed to overcome limitations on communication among the agents and their limited sensory capability, with the only drawback of a small increase in the duration of the mission due to the necessity of each robot to possibly stop and search

for information.

- **Analysis of Performances with Large Teams**
  This further set of tests focused on the analysis of the computational effort required

Table 4.1: CPU Time vs Number of Robots

Number of Robots	AVG CPU Time	Standard Deviation
6	138.8 ms	15.83
9	126.0 ms	16.26
12	125.7 ms	14.18

to each agent depending on the total number of robots employed in the mission. The aim of this investigation is evaluating the scalability of the proposed solution, and check if the fixed total budget of particles is capable of making the algorithm independent from the number of robots. Teams with 6, 9 and 12 robots were employed, and the time required by the single agents for computation was measured. The simulation setup and robot's parameters were the same as the previous set of experiments, in particular F_i has been defined as an angular sector with amplitude $\beta = 120°$ and radius $R_d = 10$ m, a reasonable value for considering a precise neighbors localization (the authors in [106] define a maximum detection range of 15 m). Performances were evaluated meaning the average time required by every robot along the entire duration of the operation and the standard deviation among these values. Results are reported in Table 4.1. The analysis of these results focuses on the relative values of average CPU time more than the absolute ones. As a matter of fact, the simulation can not reflect the real computation time since the detection and relative localization of neighbors was only emulated, in addition to hardware differences. For this reason, the analysis has been performed over a comparison of data related to teams with different number of agents, highlighting no significant differences in the average computation time required, even when numerous teams were employed. From these results, we can prove that the computational complexity of the particle filtering algorithm does not depend on the number of robots thanks to the fixed budget of particles to be continuously re-allocated, while the increasing number of cascaded QP problems (4.27) implies a negligible increase in the total computation time. In conclusion, the analysis suggests the scalability of the control strategy and indicates that it is nearly independent from the number of robots in the team.

Second Case Study: Intersecting Trajectories

Finally, we tested the behavior of a team of 4 UAVs when instructed to reach a desired configuration, which could lead to intersecting trajectories being followed by each robot. The aim of this last set of experiments was to check whether the agents were able to avoid collisions even without any information about the target position of their teammates. The

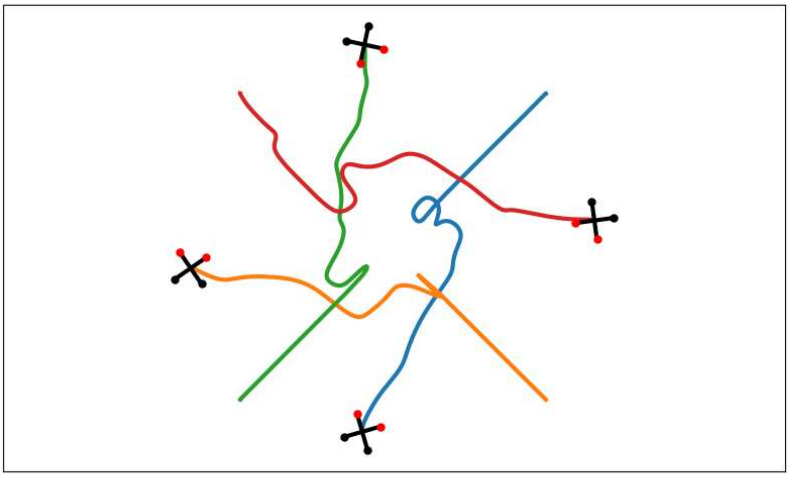


Figure 4.16: Intersecting trajectories for 4 UAVs

simulation setup was the same as previous tests, with the only difference on the estimated control input for the prediction step (4.12). Since the behavior of robot j is unknown, we considered the dangerous case of j moving towards i at half the maximum speed:

$$ {}^{i}\mathbf{u}_{j}^{est} = -\frac{v_{MAX}}{2}\frac{{}^{i}p_{j}}{\|{}^{i}p_{j}\|} \tag{4.31} $$

The desired control law ${}^{i}\mathbf{u}_{i}^{*}$ is again calculated as a proportional law towards the target. As we can see from Fig. 4.16, each UAV is capable of deviating from $\mathbf{u}^{*}$ in order to avoid collisions, then proceed to the target location occasionally searching for neighbors. The successful behavior of the team confirmed the effectiveness of our solution even in challenging scenarios with potential collisions.

4.2.8 Conclusions

This work presents a distributed approach for controlling a multi-robot system in communication-denied scenarios, particularly those with robots having a limited camera field of view for neighbor position retrieval. The key contribution is an optimization-based control ensuring collision avoidance while accomplishing a global task. The exploration-exploitation dilemma is addressed, requiring robots to balance information seeking about neighbors' positions with achieving the mission's global goal. Each robot predicts neighbor positions using a particle filter and defines confidence ellipses to address uncertainty. Control Barrier Functions constrain robot motion to keep a target in its field of view, and confidence ellipse distances are used to assess the opportunity to soften constraints. Simulations demonstrate successful collision-free task completion, and scalability is confirmed with varying robot numbers. Future work will focus on eliminating the need for IDs, evaluating overall area probabilities for robot presence, implementing an error model for neighbor localization, and addressing obstacles in the environment.

4.3 Robust Trajectory Generation and Control for Quadrotor Motion Planning with Field-of-View Control Barrier Certification

Many approaches to multi-robot coordination are susceptible to failure due to communication loss and uncertainty in estimation. Moving beyond the reactive control formulations discussed in the previous section, we present a real-time communication-free distributed navigation algorithm integrated into a Model Predictive Control framework and certified by control barrier functions. This approach models and controls the onboard sensing behavior to keep neighbors in the limited field of view for position estimation. The method is robust to temporary tracking loss and directly synthesizes control to stabilize visual contact through control Lyapunov-barrier functions. The main contributions of this work are a continuous-time robust trajectory generation and control method certified by control barrier functions for distributed multi-robot systems and a discrete optimization procedure, namely, MPC-CBF, to approximate the certified controller. In addition, we propose a linear surrogate of high-order control barrier function constraints and use sequential quadratic programming to solve MPC-CBF efficiently.

4.3.1 Introduction

Multi-robot systems, such as those used in search and rescue [112], active target tracking [113], and collaborative transportation [114], demand real-time distributed coordination solutions robust to communication compromises. Communication is vulnerable to adversarial attacks and faces challenges such as dropped messages, delays, and scalability [31]. In contrast, onboard sensing to estimate neighbors' states is robust to compromised communication. However, one major challenge is dealing with imperfect perception. For instance, onboard cameras typically have a restricted angular field of view, leading to a trade-off between task completion and neighbor detection. Although 360° cameras offer an omnidirectional field of view, object tracking on 360° images remains challenging due to significant distortions and stitching artifacts [115]. In addition, the limited frame rate and uncertainty pose challenges for practical applications. For instance, a robot may lose track of a neighbor due to image blur caused by the vehicle motion or inaccurate estimation due to measurement noise.

In this work, we present a real-time robust trajectory generation and control strategy for navigation tasks in distributed multi-robot systems that respects visual contact in communication-denied environments. Keeping neighbors in the field of view during navigation is challenging due to the abovementioned limitations. When estimation uncertainty is present or it is infeasible to track all neighbors, temporary compromises of field-of-view constraints are inevitable. A robust controller is demanded to regain the visual contact between robots in such scenarios.

Our robust control strategy utilizes Control Barrier Functions (CBFs) [116] to maintain visual contact with neighbors and regain it after temporary loss. We consider

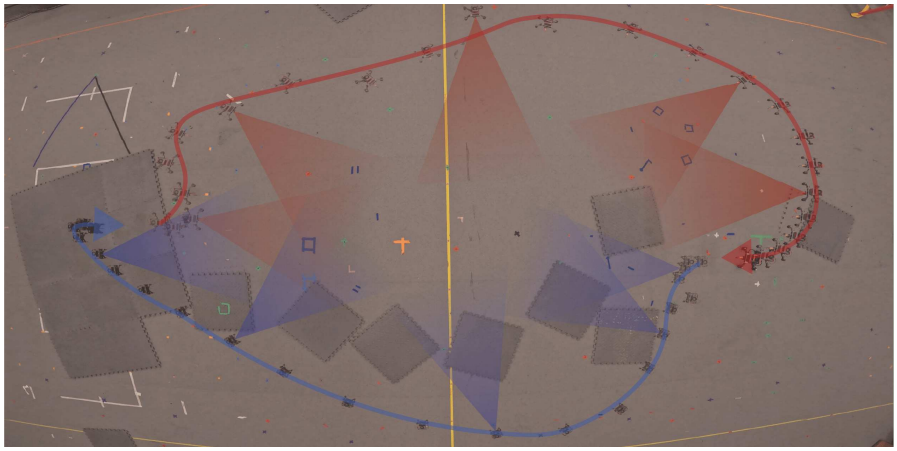


Figure 4.17: Long exposure top view of 2 quadrotors navigating with distributed controller respecting field-of-view constraints. The red and blue triangles are the fields of view of UAV1 and UAV2. The sensing ranges extend beyond triangles and are omitted in the figure. Curves represent the robot routes.

a continuous-time trajectory and control generation problem certified by CBFs. We consider a double integrator model, which requires high-order control barrier functions (HOCBFs) to satisfy the field-of-view constraints. Tracking robustness follows from the Lyapunov-like property of HOCBFs. We propose a discrete optimization framework, model predictive control with control barrier functions (MPC-CBF). Since imposing HOCBF constraints for all times in the horizon is intractable, MPC-CBF applies constraints at sampled time stamps to approximate the certified solution. We introduce a linear surrogate for HOCBF constraints and solve MPC-CBF via sequential quadratic programming (SQP) using a quadratic programming (QP) solver. Our framework jointly generates a certified continuous-time trajectory and a controller in real time using piecewise splines. Our algorithm provides inputs up to an arbitrary order of derivatives. Our contributions can be summarized as follows:

- a real-time distributed controller that maintains visual contact between robots in communication-denied areas and tolerates temporary tracking loss during navigation;
- a continuous-time spline-based trajectory and control generation method certified by control barrier functions;
- an optimization framework, namely MPC-CBF, that imposes the HOCBF constraints at sampled time stamps and approximates the certified solution. The open-source code of our implementation is available at `https://github.com/LishuoPan/fovmpc`.

We demonstrate our algorithm in simulations with up to 10 robots and in physical experiments with 2 custom-built UAVs, shown in Fig. 4.17.

4.3.2 Related Work

Control barrier functions provide sufficient and necessary conditions to guarantee safety [39]. Despite success in collision avoidance [117], lane keeping and adaptive cruise control [118], CBFs synthesize reactive control that can be short-sighted behaviors in planning tasks. For instance, CBF-based controllers can cause deadlocks in multi-robot navigation [119],

and aggressive trajectory and heading controls that lead to more failures in visual maintenance and goal reaching in visual contact navigation. CBFs have recently been applied to limited field-of-view problems: in [120] for only static gates in drone racing, in [121] for pursuit settings to track a moving target, and in [122] for multi-robot navigation with triangular sensing. However, the approach in [122] requires centralized intervention and omnidirectional sensing. In this work, we combine planning with CBFs to overcome such disadvantages.

Attempts have been made to combine MPC with CBFs in a discrete-time formulation [123]. Our continuous-time controller improves system responsiveness and stability. In continuous-time formulations, a multi-layer controller [124] solves optimal control with CBF constraints. Our approach provides additional smoothness and derivatives up to an arbitrary order. A spline-based trajectory generation method imposes CBF constraints utilizing polygonal cells [125]; however, it only solves the trajectory and requires separate control synthesis. Instead, our framework solves trajectory and control *concurrently*. Our approach has the following advantages:

1. It unifies trajectory and control generation: control inputs can be computed directly from the trajectory.

2. It provides smooth control inputs up to an arbitrary derivative order.

3. It delivers *continuous-time* trajectory and control, thus responding better to delays and stabilizing agile systems.

4. It performs in real time.

4.3.3 Preliminaries

Bézier Curve

We use piecewise splines $f(t)$ to impose smoothness requirements in the trajectory generation problem and obtain a trajectory with derivatives up to an arbitrary defined order. The i-th Bézier curve in the piecewise splines $f_i : [0, \tau_i] \to \mathbb{R}^d$ is parameterized by time, with duration τ_i. The Bézier curve of arbitrary degree h with duration τ_i is defined by $h+1$ control points $\boldsymbol{\mathcal{U}}_i = [\boldsymbol{u}_{i,0}; \ldots; \boldsymbol{u}_{i,h}]$. We first construct Bernstein polynomials $\boldsymbol{B}_v^h \in \mathbb{R}$ of degree h [126]:

$$\boldsymbol{B}_v^h = \binom{h}{v} \left(\frac{t}{\tau}\right)^v \left(1 - \frac{t}{\tau}\right)^{h-v}, \forall t \in [0, \tau], \tag{4.32}$$

where $v = 0, 1, \cdots, h$. A d-dimensional Bézier curve is defined as $f_i(t) = \sum_{v=0}^{h} \boldsymbol{u}_{i,v} \boldsymbol{B}_{i,v}^h$ with $\boldsymbol{u}_{i,v} \in \mathbb{R}^d$. The finite set of control points $\boldsymbol{\mathcal{U}} = [\boldsymbol{\mathcal{U}}_0; \ldots; \boldsymbol{\mathcal{U}}_{P-1}]$ uniquely characterize a piecewise spline of P Bézier curves and act as decision variables in the trajectory generation problem. The duration of the entire piecewise spline is $\tau = \sum_{i=0}^{P-1} \tau_i$.

Neighbors State Estimation

We adopt particle filters for neighbor position estimation as in the previous section. The neighbor's position belief is represented by $N_p \in \mathbb{N}$ weighted particles, each indicating a possible position. Particles evolve following a transition function, update weights with new measurements, and are resampled accordingly. When the target is outside the field of view, we reduce the weight of particles inside the sensing region $\mathcal{F}_i$ by setting $w_j^k \leftarrow \varepsilon w_j^k$, where $\varepsilon \in [0, 1)$. Unlike our previous work in Section 4.2, our method accommodates occasional missed detections by uniformly scaling weights of particles within $\mathcal{F}_i$, preserving the normalized distribution.

4.3.4 Problem Formulation

Consider N homogeneous robots in a communication-denied workspace $\mathcal{W}$. Let $\mathcal{R}(\mathbf{r}_i)$ denotes the convex set of points representing robot i at position $\mathbf{r}_i \in \mathbb{R}^3$. Robots generate trajectories and controls concurrently in a decentralized manner to reach goals while maintaining visual contact and avoiding collisions without communication. Each robot estimates others' positions via the field of view of an onboard camera. We assume that the initial teammate positions are shared via a pre-mission communication. Our optimization problem solves trajectory and control by minimizing the control effort, and the distance to the goal, subject to dynamics, initial state, control continuity, safety corridor, and CBF constraints.

Robot model

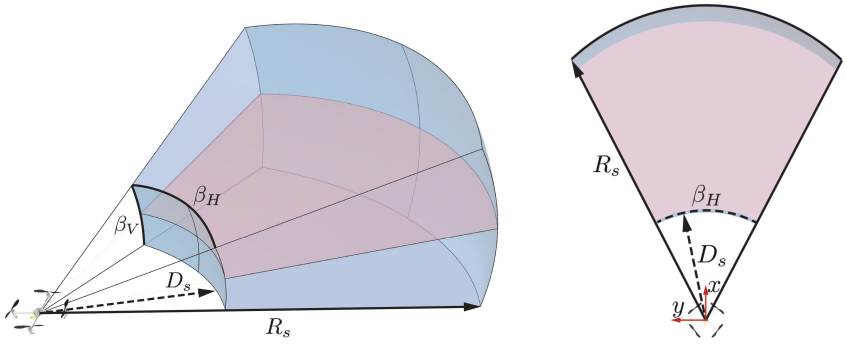
We consider a double integrator, for faster computation and a unified planning and control framework using Bézier curve. The state includes position, yaw, and corresponding first-order derivatives $\mathbf{x} = [\mathbf{r}; \phi; \dot{\mathbf{r}}; \dot{\phi}] \in \mathbb{R}^8$, where $\mathbf{r} \in \mathbb{R}^3$, $\phi \in \mathbb{R}$.

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}, \tag{4.33}$$

where the control input $\mathbf{u} = [\mathbf{u}_r; u_\phi] \in \mathbb{R}^4$ is the acceleration. The system output is $\mathbf{y} = [\mathbf{r}; \phi] \in \mathbb{R}^4$. We denote velocity by $\mathbf{v} = [\dot{\mathbf{r}}; \dot{\phi}] \in \mathbb{R}^4$. Physical limits are given by minimum velocity and acceleration $\mathbf{v}_{\min},\ \mathbf{a}_{\min} \in \mathbb{R}^4$, maximum velocity and acceleration $\mathbf{v}_{\max},\ \mathbf{a}_{\max} \in \mathbb{R}^4$, respectively. $A = [\mathbf{0}, \mathbf{I}; \mathbf{0}, \mathbf{0}] \in \mathbb{R}^{8\times8}$, $B = [\mathbf{0}; \mathbf{I}] \in \mathbb{R}^{8\times4}$, with $\mathbf{0}, \mathbf{I} \in \mathbb{R}^{4\times4}$ denoting the zero and identity matrices. At time t_0, we generate a trajectory $\mathbf{x}(t|t_0)$ and control $\mathbf{u}(t|t_0)$ over horizon τ.

Sensing model

Each robot senses via an onboard camera aligned with the body frame, positive x-axis. We model the robot i's sensing region $\mathcal{F}_i$ as a truncated spherical sector as shown in Fig. 4.18, limited by a maximum perception range $R_s > 0$, and a minimum safety distance from the other robots $D_s > 0$. Horizontal and vertical fields of view are given by angles



(a) Robot sensing region **(b)** Top view

Figure 4.18: The sensing region $\mathcal{F}$ of a robot is modeled as truncated spherical sector. β_H, β_V are the horizontal and vertical field of view angles. R_s is the sensing range and D_s is the safety distance. The blue volume (or red plane in 2D) is the region where the neighbor can be safely detected.

$\beta_H, \beta_V \in [0, 2\pi)$, respectively. A neighbor's position is observable when inside the field of view. Robot i measures the relative position of neighbor j in its body frame, i.e., ${}^i\mathbf{r}_j = \mathbf{r}_j - \mathbf{r}_i$. We model measurement uncertainty as a zero-mean multivariate Gaussian noise with covariance matrix $R_{\mathrm{m}} \in \mathbb{R}^{3\times3}$.

4.3.5 HOCBF Design

We require a robot to maintain a safety distance, visual contact, and perception range with its neighbors. We formulate the following CBF for robot i in the form $b({}^i\mathbf{r}_j) \geq 0$, $\forall j \in \mathcal{N}_i$. Here, we denote the neighbors of robot i as $\mathcal{N}_i$ (we consider all the other robots; however, a sensing range can be enforced if desired). In this work, we only focus on 2D motion, which can be applied to ground robots or aerial vehicles that fly at the same altitude. Thus, the sensing region is a planar angular sector defined by β_H (see Fig. 4.18b). The safety distance and range CBFs are defined as follows:

$$b_{\mathrm{sr}}({}^i\mathbf{r}_j) = \begin{bmatrix} {}^ix_j & {}^iy_j \\ -{}^ix_j & -{}^iy_j \end{bmatrix} \begin{bmatrix} {}^ix_j \\ {}^iy_j \end{bmatrix} + \begin{bmatrix} -D_s^2 \\ R_s^2 \end{bmatrix}, \forall j \in \mathcal{N}_i, \tag{4.34}$$

We extended the field-of-view CBFs in [2] to include $\beta_H \in [\pi, 2\pi)$ and our CBFs are defined as follows:

$$b_{\mathrm{fov}}({}^i\mathbf{r}_j) = \begin{cases} \begin{bmatrix} \tan(\beta_H/2) & 1 \\ \tan(\beta_H/2) & -1 \end{bmatrix} \begin{bmatrix} {}^ix_j \\ {}^iy_j \end{bmatrix}, & \text{if } \beta_H \in [0, \pi) \\ \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} {}^ix_j \\ {}^iy_j \end{bmatrix}, & \text{if } \beta_H = \pi \\ \begin{bmatrix} \tan(\pi - \frac{\beta_H}{2}) & \mathrm{sign}({}^iy_j) \end{bmatrix} \begin{bmatrix} {}^ix_j \\ {}^iy_j \end{bmatrix}, & \text{if } \beta_H \in (\pi, 2\pi) \end{cases} \tag{4.35}$$

The CBFs $b_{\text{sr}}({}^i\mathbf{r}_j)$ force robot i to maintain a minimum safety distance D_s and a maximum distance equal to the sensing range R_s from robot j, $\forall j \in \mathcal{N}_i$. Meanwhile, $b_{\text{fov}}({}^i\mathbf{r}_j)$ force the robot to keep the neighbor robot j inside the 2D visual cone with amplitude β_H. Combining them, we obtain:

$$b({}^i\mathbf{r}_j) = \left[b_{\text{sr}}({}^i\mathbf{r}_j); b_{\text{fov}}({}^i\mathbf{r}_j)\right] \geq 0, \forall j \in \mathcal{N}_i, \tag{4.36}$$

constraining robots to keep all their neighbors inside the field of view while moving. Note that b has a relative degree $q = 2$ with respect to system dynamics (4.33). Therefore, we use HOCBFs to guarantee constraint satisfaction. Choosing $\alpha_1(b({}^i\mathbf{r}_j)) = \gamma_1 b^3({}^i\mathbf{r}_j)$ and $\alpha_2(\psi_1({}^i\mathbf{r}_j)) = \gamma_2\psi_1^3({}^i\mathbf{r}_j)$, and recalling the HOCBF constraint definition in (2.18) for $q = 2$:

$$\sup_{\mathbf{u}\in U}[L_f^2 b(\cdot) + L_g L_f b(\cdot)\mathbf{u} + O(b(\cdot)) + \alpha_r(\psi_1(\cdot))] \geq 0,$$

we obtain the HOCBF constraint:

$$L_f^2 b({}^i\mathbf{r}_j) + L_g L_f b({}^i\mathbf{r}_j)\mathbf{u} + 3\gamma_1 b^2({}^i\mathbf{r}_j) L_f b({}^i\mathbf{r}_j) + \gamma_2(L_f b({}^i\mathbf{r}_j) + \gamma_1 b^3({}^i\mathbf{r}_j))^3 \geq 0. \tag{4.37}$$

Notably, we choose $\alpha_1(\cdot)$ and $\alpha_2(\cdot)$ as odd power functions, which belong to extended class $\mathcal{K}$ functions. Therefore, the designed HOCBF (4.36) is also a HOCLBF [116] and brings the system back into the safe set $\mathcal{C}$ if not already within (see [116, Theorem 2]). We use this property in our controller to tolerate constraint violations and enable the robot to regain visual contact with its neighbors after temporary tracking loss.

4.3.6 Trajectory and Control Generation with Safety Certification

Our algorithm generates the continuous-time trajectory and control certified by control barrier functions, utilizing piecewise splines. Our optimization problem solves for the piecewise h-th order Bézier curves. The trajectory is defined as the piecewise Bézier curves and their first-order derivatives. The control inputs $\mathbf{u}(t)$ are defined as their second derivatives. We choose a sufficiently large h to generate a smooth control $\mathbf{u}(t)$. To satisfy the visual contact requirement, we impose the HOCBF constraints in (4.37), for any given t in the horizon. The general form of our problem is formulated as follows:

$$\begin{aligned}
\underset{\boldsymbol{u}}{\text{argmin}}\ & \mathcal{J}_{\text{cost}} && \text{(4.38a)}\\
\text{s.t. } & \dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t) && \text{(4.38b)}\\
& \frac{d^j f(0)}{dt^j} = \frac{d^j \mathbf{r}(t_0)}{dt^j},\ \forall j \in \{0, \ldots, C\} && \text{(4.38c)}\\
& f \text{ continuous up to derivative } C && \text{(4.38d)}\\
& A^{\text{cbf}}\mathbf{u}(t) + \boldsymbol{b}^{\text{cbf}}({}^i\hat{\mathbf{r}}_j(t|t_0)) \geq 0,\ \forall t \in [t_0, t_0+\tau] \quad \forall j \in \mathcal{N}_i && \text{(4.38e)}\\
& \mathbf{a}_{\min} \preceq \mathbf{u}(t) \preceq \mathbf{a}_{\max},\ \forall t \in [t_0, t_0+\tau] && \text{(4.38f)}\\
& \mathbf{v}_{\min} \preceq \mathbf{v}(t) \preceq \mathbf{v}_{\max},\ \forall t \in [t_0, t_0+\tau], && \text{(4.38g)}
\end{aligned}$$

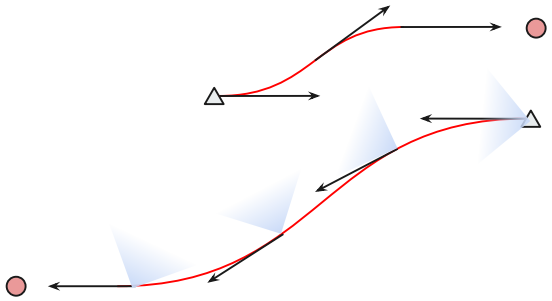


Figure 4.19: Robot navigates to its goal (red dot) with predicted field of views (blue triangles). MPC-CBF imposes constraints at sampled steps.

where $\preceq$ stands for element-wise less than or equal to, t_0 is the current time stamp, and C is the highest order of derivatives required for continuity. The constraint (4.38e) is equivalent to (4.37), where $A^{\text{cbf}} = L_g L_f b$, $\boldsymbol{b}^{\text{cbf}} = L_f^2 b + (2\mu+1)\gamma_1 b^{2\mu} L_f b + \gamma_2(L_f b + \gamma_1 b^{(2\mu+1)})^{(2\mu+1)}$. Note, ${}^i\hat{\mathbf{r}}_j(t|t_0) = \hat{\mathbf{r}}_j(t_0) - \mathbf{r}_i(t)$, as we can only obtain the current estimation of the neighbor $\hat{\mathbf{r}}_j(t_0)$ in a communication-denied setting.

Theorem 2. *Consider the HOCBF in* (4.36) *and the set* $\mathcal{C} := \mathcal{C}_1 \cap \mathcal{C}_2$. *Let* α_1, α_2 *be differentiable extended class* $\mathcal{K}$ *functions. If* $\mathbf{x}(t_0) \in \mathcal{C}$, *the controller* $\mathbf{u}(t)$ *from* (4.38), $\forall t \in [t_0, t_0+\tau]$ *renders* $\mathcal{C}$ *forward invariant. Otherwise,* $\mathbf{u}(t)$ *from* (4.38), $\forall t \in [t_0, t_0+\tau]$ *stabilizes system* (4.33) *towards the set* $\mathcal{C}$.

Proof. $\mathbf{u}(t)$ is Lipschitz continuous as it is defined as the second-order derivative of the Bézier curve with sufficient continuity. The constraint (4.38b) requires the state transition to obey the system model in (4.33). The constraints (4.38e) render the system safe in the horizon or stabilize the system to a safe set $\mathcal{C}$ following directly from HOCLBFs properties (see [116, Theorem 2]). □

Solving the above optimization, however, is intractable, as (4.38e) imposes constraints for all t in continuous time, yielding an infinite number of constraints. Instead, we propose a discrete optimization scheme to approximate the solution, depicted in Fig. 4.19. Our approach imposes the HOCBF constraints only at time stamps sampled at fixed intervals over the horizon. The trajectory is replanned in real-time and the most recent optimized trajectory is executed only up to the first sampled time step. Since our optimization scheme acts similarly to an MPC with continuous-time control inputs, we name our algorithm model predictive control with control barrier functions, or MPC-CBF for short.

Trajectory and Control Prediction Model

We introduce the notation $\hat{(\cdot)}(k|t_0)$, which represents the prediction of $(\cdot)(k|t_0)$, given information at time t_0 and horizon $k \in \{0, \cdots, K-1\}$, where $(K-1)\delta = \tau$. Here δ is the duration of each discrete time step. The prediction of system output $\hat{\mathbf{y}}(k|t_0)$ is the optimized piecewise Bézier curve resulting from (4.38); its first and second-order derivatives $\hat{\mathbf{v}}(k|t_0)$ and $\hat{\mathbf{u}}(k|t_0)$ can be computed in closed form. By definition, the

predicted trajectory $\hat{\mathbf{x}}(k|t_0) = [\hat{\mathbf{y}}(k|t_0); \hat{\mathbf{v}}(k|t_0)]$ is the solution of (4.33) given control inputs $\hat{\mathbf{u}}(k|t_0)$.

HOCBF Constraints and Relaxation

To satisfy the safety requirements, we approximate HOCBF constraints in (4.38e) using a sampling-based approach. We constrain $\hat{\mathbf{u}}(k|t_0)$ at discrete time steps over horizon by

$$A^{\text{cbf}}\hat{\mathbf{u}}(k|t_0) + \boldsymbol{b}^{\text{cbf}}({}^i\hat{\mathbf{r}}_j(k|t_0)) \geq 0, \ \forall j \in \mathcal{N}_i \\ \forall k \in \{0, \dots, K-1\}. \tag{4.39}$$

As samples increase, constraints (4.39) approach HOCBF constraints in (4.38e). Note that the prediction $\hat{\mathbf{y}}(k|t_0)$ depends on the decision variables, hence ${}^i\hat{\mathbf{r}}_j(k|t_0)$ does too. The constraint (4.39) is nonlinear, due to the nonlinearity of A^{cbf} and $\boldsymbol{b}^{\text{cbf}}$. In Section 4.3.7, we propose a linear approximation of the constraint and solve the problem with SQP.

More neighbors increase the number of constraints, which leads to infeasibility. We relax (4.39) with slack variables for distant robots, which do not pose a danger of collisions. We define slack variables $\epsilon_j \geq 0$, for $j \in \mathcal{N}_i$ (see Sec. 4.38).

Collision Avoidance Constraints

The safety distance HOCBF cannot guarantee collision avoidance in the horizon, as the robot can only estimate the current relative position of its neighbors ${}^i\hat{\mathbf{r}}_j(t_0)$ without knowing their plans. We use a separating hyperplane approach, similar to [127], to guarantee collision avoidance in belief space. A function $L(\mathcal{A}, \mathcal{B})$ computes a separating half-space $\hat{\mathcal{H}}_r := \{\mathbf{r} \in \mathcal{W} \mid \boldsymbol{w}_r^\top \mathbf{r} + b_r \leq 0\}$, where $\boldsymbol{w}_r$ and b_r are the weights and bias of the half-space, respectively, and $\mathcal{A}$ and $\mathcal{B}$ are the convex hulls representing the robots. We compute Voronoi-cell separation between $\mathbf{r}_i$ and $\mathbf{r}_j$ as $\hat{\mathcal{H}}_r$. By buffering the half-space by an offset $b_r' = b_r + \max_{\boldsymbol{y} \in \mathcal{R}(\mathbf{0})} \boldsymbol{w}_r^\top \boldsymbol{y}$, we obtain that the safety corridor consists of $\mathcal{H}_r$ for robot i. The Bézier curve f_i generated at the negative side of $\mathcal{H}_r$ guarantees collision avoidance with its neighbor's belief. We can write this constraint in the form

$$A_i^{\text{col}}\boldsymbol{u}_{i,j} + \boldsymbol{b}_i^{\text{col}} \leq 0, \forall i \in \{0, \dots, P-1\} \\ \forall j \in \{0, \dots, h\}. \tag{4.40}$$

Output and Derivatives Continuity

To guarantee continuity of the system output and its derivatives, we need to impose continuity between the splines, thus adding the following constraints,

$$\frac{d^j f_i(\tau_i)}{dt^j} = \frac{d^j f_{i+1}(0)}{dt^j}, \forall i \in \{0, \dots, P-2\} \\ \forall j \in \{0, \dots, C\}. \tag{4.41}$$

Physical Limits

We require limits on the derivatives due to physical constraints. The derivatives of Bézier curves are confined within the convex hull of the derivative's control points. This approach, however, is overly conservative [128]. In [129], the duration of the Bézier curve is iteratively rescaled until the physical constraints are satisfied. Inspired by [130], we propose an approach that leverages our discrete optimization scheme. We bound the values of $\hat{\mathbf{v}}(k|t_0)$, $\hat{\mathbf{u}}(k|t_0)$ in the horizon,

$$\mathbf{v}_{\min} \preceq \hat{\mathbf{v}}(k|t) \preceq \mathbf{v}_{\max},\ \forall k \in \{0, \dots, K-1\}, \tag{4.42}$$

$$\mathbf{a}_{\min} \preceq \hat{\mathbf{u}}(k|t) \preceq \mathbf{a}_{\max},\ \forall k \in \{0, \dots, K-1\}. \tag{4.43}$$

Cost Functions

We optimize the predicted trajectory and control inputs considering different objectives according to the task.

- *Goal Cost*: The trajectory should navigate the robot towards its goal $\mathbf{y}_d \in \mathbb{R}^4$. We penalize the squared distance between the last κ sampled predictions $\hat{\mathbf{y}}(k|t_0)$ and the goal,

$$\mathcal{J}_{\text{goal}} = \sum_{k=K-\kappa}^{K-1} \omega_k \left\| \hat{\mathbf{y}}(k|t_0) - \mathbf{y}_d \right\|_2^2, \tag{4.44}$$

  where ω_k is the weight for k-th sample.
- *Control Effort Cost*: We minimize the weighted sum of the integral of the square of the norm of derivatives,

$$\mathcal{J}_{\text{effort}} = \sum_{j=1}^{C} \theta_j \int_{t_0}^{t_0+\tau} \left\| \frac{d^j}{dt^j} f(t) \right\|_2^2 dt, \tag{4.45}$$

  where θ_j is the weight of the order of derivatives.
- *Priority Cost*: We prioritize tightening the HOCBF constraints for the nearest neighbors to maintain visual contact and prevent impending collisions. We derive a confidence ellipsoid $\mathcal{R}_j^{95}$ containing the real position $\mathbf{r}_j$ with 95% probability from the particle filter. We find the distance d_{ij} between robot i and $\mathcal{R}_j^{95}$ following the solution in Section 4.2. Sorting neighbors based on the distance d_{ij} (from the closest to the farthest one), we obtain an ordered set $\overline{\mathcal{N}}_i$, and prioritize the satisfaction of the HOCBF constraints on robots that are believed to be closer.
  Priority assignment is achieved by adding slack variables ϵ_j with exponentially decaying weights $\xi_j = \Omega \cdot \gamma_s^j$ as a cost function, where $\Omega \in \mathbb{R}_{>0}$ is the cost factor and $\gamma_s \in (0, 1)$ is the decay factor. Therefore, the cost can be defined as

$$\mathcal{J}_{\text{prior}} = \sum_{j \in \overline{\mathcal{N}}_i} \xi_j \epsilon_j. \tag{4.46}$$

  Slack variables let robot i temporarily lose visual contact with a distant neighbor

j, but re-establish it when the uncertainty grows and the ellipsoid $\mathcal{R}_j^{95}$ is close.

4.3.7 Solving MPC-CBF Optimization

As mentioned in Sec. 4.3.6, the HOCBF constraints in (4.39) are nonlinear. We propose a linear surrogate of HOCBF constraints and use the SQP to solve the proposed problem with a QP solver. The SQP iteratively solves MPC-CBF. In each iteration, we use the prediction from the previous QP as the surrogate of the states to compute A^{cbf} and $\boldsymbol{b}^{\text{cbf}}$. This decouples the dependency between the prediction and decision variables, so that A^{cbf} and $\boldsymbol{b}^{\text{cbf}}$ can be treated as constants. The initial QP is solved with HOCBF constraint only on the observable state at time t_0 and predicts $\hat{\mathbf{x}}_0(k|t_0)$ and $\hat{\mathbf{u}}_0(k|t_0)$. Here, the subscription indicates the QP iteration index. However, the predicted trajectory and control inputs do not necessarily satisfy HOCBF constraints in the horizon. For the ν-th QP iteration, we substitute ${}^i\mathbf{r}_j(k|t_0)$ with the predicted ${}^i\hat{\mathbf{r}}_{j,\nu-1}(k|t_0)$ from the previous QP, for $\nu = 1, \ldots, \mathcal{V}-1$. The ν-th QP is formulated as follows:

$$\underset{\boldsymbol{\mathcal{U}}}{\text{argmin}}\ \mathcal{J}_{\text{effort}} + \mathcal{J}_{\text{goal}} + \mathcal{J}_{\text{prior}} \tag{4.47a}$$

$$\text{s.t. } \frac{d^j f(0)}{dt^j} = \frac{d^j \mathbf{r}}{dt^j},\ \forall j \in \{0, \ldots, C\} \tag{4.47b}$$

$$\frac{d^j f_i(T_i)}{dt^j} = \frac{d^j f_{i+1}(0)}{dt^j},\ \forall i \in \{0, \ldots, P-2\},\ \forall j \in \{0, \ldots, C\} \tag{4.47c}$$

$$A_i^{\text{col}} \boldsymbol{u}_{i,j} + \boldsymbol{b}_i^{\text{col}} \leq 0,\ \forall i \in \{0, \ldots, P-1\},\ \forall j \in \{0, \ldots, h\} \tag{4.47d}$$

$$A^{\text{cbf}} \hat{\mathbf{u}}(k|t_0) + \boldsymbol{b}^{\text{cbf}}({}^i\hat{\mathbf{r}}_{j,\nu-1}(k|t_0)) + \epsilon_j \geq 0,\ \forall j \in \mathcal{N}_i,\ \forall k \in \{0, \ldots, K_r - 1\} \tag{4.47e}$$

$$\mathbf{v}_{\min} \preceq \hat{\mathbf{v}}(k|t_0) \preceq \mathbf{v}_{\max}, \forall k \in \{0, \ldots, K-1\} \tag{4.47f}$$

$$\mathbf{a}_{\min} \preceq \hat{\mathbf{u}}(k|t_0) \preceq \mathbf{a}_{\max}, \forall k \in \{0, \ldots, K-1\} \tag{4.47g}$$

$$\epsilon_j \geq 0,\ \forall j \in \mathcal{N}_i. \tag{4.47h}$$

Note that the robot estimates neighbors' positions only at t_0. Adding visual constraints for the entire horizon based on this estimate leads to an overly conservative plan. Instead, we satisfy HOCBF constraints (4.47e) only up to K_r steps in the horizon.

4.3.8 Simulation Results

We define two sets of instances. In "Circle" instances, robots are initialized uniformly on a circle with antipodal goals; their start and goal headings face the circle's center. In "Formation" instances, robots are initialized in grids and demanded to move forward; start and goal headings are set to 0° yaw.

To reflect the uncertainty in the system dynamics, Gaussian noise is added to the system output and velocity, i.e., $\mathbf{y}(k|t_0) \sim \mathcal{N}\left(\hat{\mathbf{y}}(k|t_0), \sigma_{\mathbf{y}}^2\mathbf{I}\right)$, $\mathbf{v}(k|t_0) \sim \mathcal{N}\left(\hat{\mathbf{v}}(k|t_0), \sigma_{\mathbf{v}}^2\mathbf{I}\right)$, where $\mathcal{N}(\boldsymbol{\mu}, \sigma\mathbf{I})$ denotes a multivariate Gaussian distribution with mean $\boldsymbol{\mu}$ and a diag-

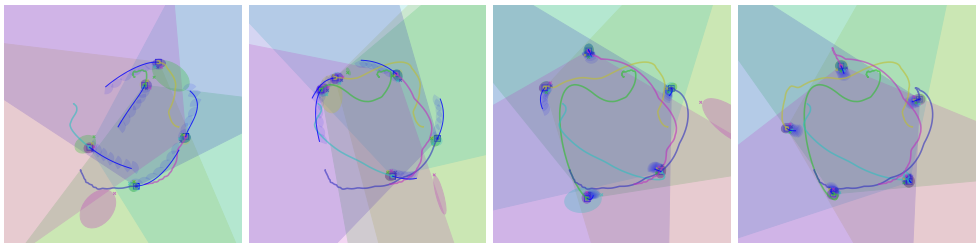
onal covariance matrix $\sigma\mathbf{I}$. We set $\sigma_{\mathbf{y}} = 0.001$, and $\sigma_{\mathbf{v}} = 0.01$. In this work, we fix the height of the robots. We set different β_H to demonstrate the property of our algorithm. We limit the acceleration in range $[-10, 10]\text{m/s}^2$ in the x-y plane, and velocity in range $[-3, 3]\text{m/s}$ for "circle" instances and $[-0.5, 0.5]\text{m/s}$ for "formation" instances. We set the yaw acceleration and yaw rate limits as $[-\pi, \pi]\text{rad/s}^2$ and $[-\frac{5}{6}\pi, \frac{5}{6}\pi]\text{rad/s}$ respectively. To expedite the computation and respect the visual contact constraints, we set $K_r = 2$, $\mathcal{V} = 2$ in the SQP solver. We set the number of pieces $P = 3$ for the piecewise spline, the degree of Bézier curves $h = 3$ with duration $\tau_i = 0.5\text{s}$, for $i = 1, 2, 3$, and require the highest order of continuity $C = 3$. In the MPC-CBF algorithm, we set the discrete sample interval $\delta = 0.1\text{s}$. In simulation, the replanning happens every 0.1s. For the particle filtering, we set the number of particles to $N_p = 100$ (initialized uniformly randomly in the workspace), the process covariance to $0.25\mathbf{I}$, the measurement covariance R_{m} to $0.05\mathbf{I}$, and the penalty factor for particles inside the field of view to $\varepsilon = 0.1$. The cost factor of slack variables is $\Omega = 1000$. The collision shape of the robot is defined as an axis-aligned bounding box in range $[-0.2, 0.2]\text{m}$ for both x-y dimensions. For the baseline, we implemented the controller from Section 4.2, extending it to an HOCBF with a double integrator under the same velocity and acceleration limits; a PD controller provides the desired input, and the control loop runs every 0.1s in simulation.

Simulation in Circle Instances

We use the following criteria for evaluation:
Success Rate: success is defined as all robots reach their goal areas and stay within them without collisions.
Makespan: time at which the last robot reaches its goal area.
Percentage of Neighbors in FoV: average percentage of neighbors the robot keeps in visual contact over the makespan.
Goals may not satisfy the visual-contact requirement. Our method compensates for visual contact and thus could lead to deviations from the goal. Thus, we consider the robot to complete its task when it reaches a goal area and stays within.

We set cost coefficients $\omega_k = 10$ for $k = K - \kappa, \ldots, K - 1$, $\theta_j = 1$ for $j = 1 \ldots C$, and $\kappa = 3$. The snapshots in Fig. 4.20 are typical routing of our control strategy in the "circle" instance with 5 robots and a $\beta_H = \frac{2}{3}\pi$ field of view. As a demonstration of controller robustness, note that the robot colored with green trajectory, when it loses visual contact with neighbors in Fig. 4.20(a), changes its position and heading in Fig. 4.20(b) to regain detection. The sensitivity of heading adjustments is controlled by the slack variable decay factor γ_s. A small γ_s prioritizes tracking the closest neighbor in the belief space, resulting in more aggressive heading responses. A large γ_s tends to track more neighbors, leading to less aggressive heading responses. An aggressive heading response can lead to an inefficient strategy due to frequent heading changes. An insensitive heading tends to overlook impending collisions, leading to actual collisions.

Figure 4.21 reports quantitative performance of our control strategy with different β_H in $[\frac{2}{3}\pi, \frac{4}{3}\pi, 2\pi]$ and slack variable decay factors γ_s in $[0.1, 0.2]$. Note that, $\beta_H = 2\pi$ always



(a) Time = 4.4s **(b)** Time = 6.8s **(c)** Time = 11.0s **(d)** Time = 15.0s

Figure 4.20: Snapshots for 5 robots in the circle instance. The ovals are 95% confidence ellipsoids of estimation (the source of estimations is indicated by colors). The predicted output is depicted as blue curves and purple field of views. The path is shown as a solid line.

satisfies the field-of-view constraints. We show *task success rate*, *makespan* (excluding failure instances), and *percentage of neighbors in FoV*. We notice that for a small number of robots (fewer or equal to 6 robots in the "Circle" instance), our controller has a similar success rate compared to the baseline. As we increase the number of robots, our algorithm outperforms the baseline with different β_H and achieves successful executions when the baseline failed to complete the task at all. The reason is that the baseline, as a reactive controller, responds more aggressively to the imminent collisions and does not maintain a large safety distance. As a result, it is more sensitive to estimate uncertainty and exposes robots to a higher risk of collision once the neighbor detections are lost. For a small β_H, the control challenge is to maintain neighbors in the field of view, thus providing the latest estimation to avoid collision, while reaching the desired goal. As the field of view increases, maintaining visual contact with neighbors becomes easier, and robots tend to take the shortest path, leading to a decrease in the *makespan*. However, it leads to deadlocks and potential collisions due to uncertainty in the estimation. The challenge of deadlocks in multi-robot planning falls outside the current scope. Modern MAPF-based path/trajectory planning addresses deadlock problems even in large-scale operations [127] and limited communication scenarios [131]. Despite the drop in *task success rate*, we note the *percentage of neighbors in FoV* remains above 60% as we scale up the number of robots even with $\beta_H = \frac{2}{3}\pi$. Our control strategy maintains the same level of visual contact with neighbors compared to the baseline controller while improving the success rate. Additionally, we report that less aggressive heading adjustments, i.e., a larger γ_s, prevent robots from repetitively searching for neighbors, resulting in a higher success rate in "Circle" instances.

Simulation in Formation Instances

In "formation" instances, we initialize all robots in grids 1m apart in the x-y direction. All robots are initialized with 0 yaw. Goals are 12m to the right of the starts, each with a yaw of 0. We set the cost coefficients $\omega_k = 300$ for $k = K - \kappa, \ldots, K - 1$, $\theta_j = 1$ for $j = 1 \ldots C$, and $\kappa = 3$. In Fig. 4.22, we show a typical result of MPC-CBF on 4 robots. Robots form and maintain visual contact with neighbors at all times, demonstrating the controller robustness.

We summarize the quantitative results in Fig. 4.23. Since robots move in the same

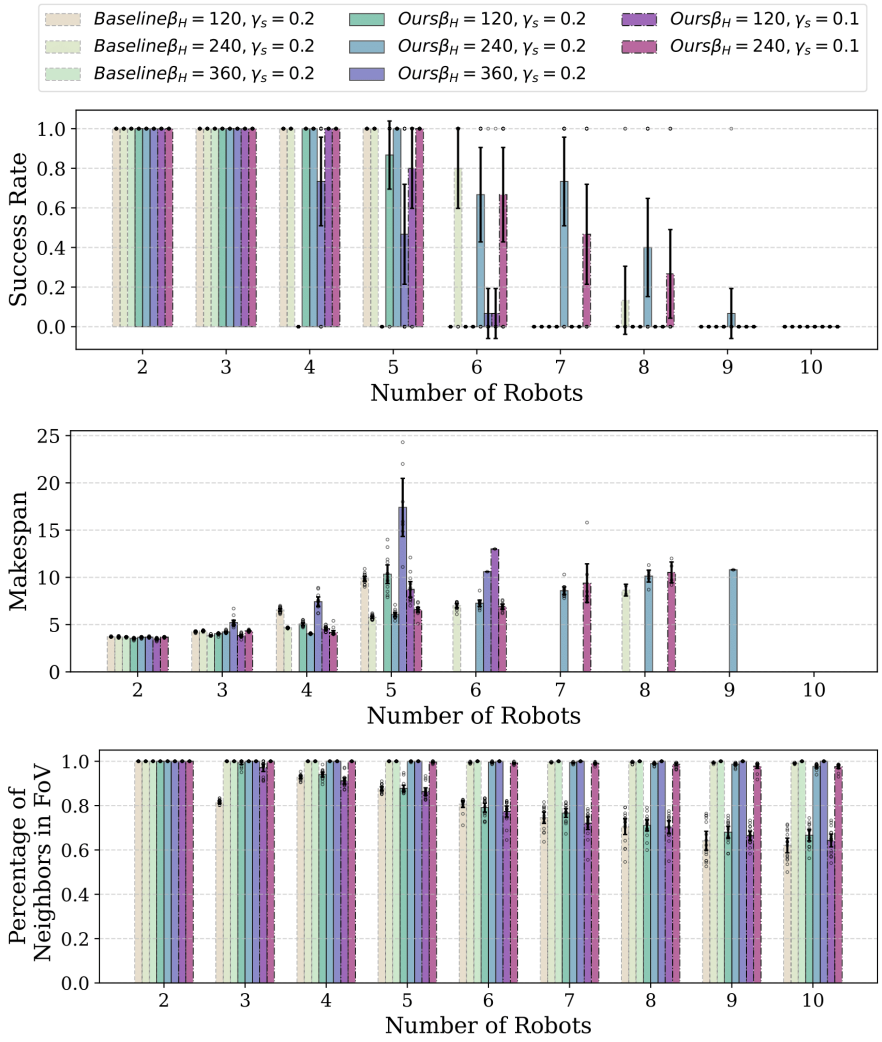
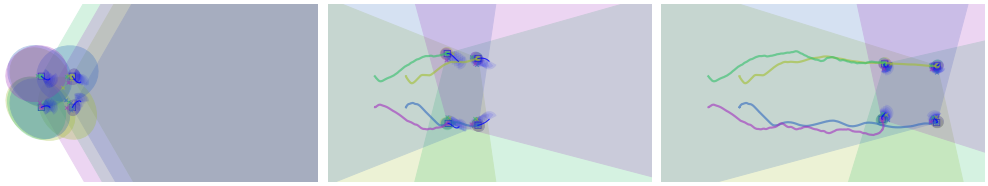


Figure 4.21: Performance of our algorithm across β_H, γ_s, and different robot counts in "circle" instances. Bars show means, error bars show 95% confidence intervals over 15 trials.

direction, unlike "Circle" instances, tasks result in fewer collisions and deadlocks. We notice an improved scalability and success rate compared to "Circle" instances. Overall, our controller outperforms the baseline across different numbers of robots. With a small field of view, the baseline satisfies the field-of-view constraints while compromising the goal-reaching capability. In contrast, our control strategy retains goal-reaching capability while maintaining visual contact with neighbors. With an omnidirectional field of view, the baseline suffers from collision due to estimation uncertainty as the number of robots increases. Increasing the field of view improves the success rate for both the baseline and our approach. In addition, a larger γ_s works better for "formation" instances. The *makespan* does not show significant changes with different β_H or γ_s, mainly because the task is less challenging regarding navigation (the *makespan* metric is omitted due to the space limit). The *percentage of neighbors in FoV* remains above 60% even with $\beta_H = \frac{2}{3}\pi$. Our controller maintains equivalent visual contact quality without compromising the goal-reaching capability. We report the runtime of the proposed perception control stack with different numbers of robots in Table 4.2.



(a) Time = 0.0s **(b)** Time = 10.0s **(c)** Time = 27.0s

Figure 4.22: 4 robots navigating in a formation instance. Their start and goal yaws are set as 0. The ovals are 95% confidence ellipsoids of estimation. The robot forms and maintains visual contact with its neighbors.

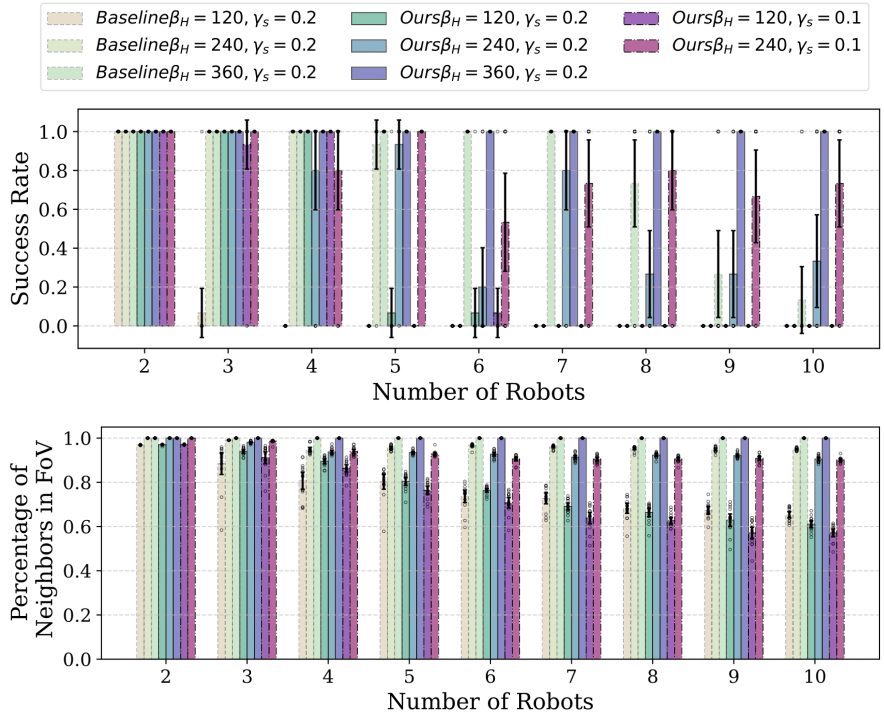


Figure 4.23: Performance on different numbers of robots in "formation" instances. The statistics are obtained in the same way in the "circle" instances.

Sensitivity Analysis

Sample interval δ. We approximate the HOCBF constraints in the horizon using a sampling-based approach. The forward invariance property, however, can be violated due to large sample intervals. We conduct experiments with different sample intervals δ over a 0.2s horizon on the "Circle" instance with 4 robots and compared with baseline and MPC-CBF without HOCBF constraints in the horizon, i.e., $K_r = 1$. All methods use slack variables to avoid optimization failure, but with a large slack cost $\Omega = 1e{+}20$. We use the *Percentage of Neighbors in FoV* to evaluate the efficacy of the set forward invariance. From Table 4.3, we note that all algorithms cannot maintain visual contact at all times. Reactive controllers, such as baseline and MPC-CBF, $K_r = 1$, perform significantly worse compared to MPC-CBF with horizon HOCBF constraints. The reactive controllers have more aggressive path and heading adjustments, leading to more failures in visual maintenance. Decreasing sample intervals δ has an insignificant influence on performance, but increases the runtime. We choose $\delta = 0.1$s, as it imposes denser CBF constraints without a significant runtime penalty.

Instance/number of Robot	2[ms]	3[ms]	4[ms]	5[ms]	6[ms]	7[ms]	8[ms]	9[ms]	10[ms]
Baseline "Circle"	1.06	2.09	3.46	4.81	6.12	7.26	8.91	9.84	11.16
Baseline "Formation"	0.76	2.04	3.43	4.87	5.59	6.94	8.44	9.79	10.56
MPC-CBF "Circle"	5.87	10.04	13.05	18.55	21.95	23.64	28.30	31.15	35.98
MPC-CBF "Formation"	5.21	7.84	10.69	14.32	17.74	19.39	22.78	25.46	28.65

Table 4.2: The runtime of baseline and MPC-CBF in "Circle" and "Formation" instances. The statistics are averaged over 200 iterations.

Methods	Percentage of Neighbors in FoV	Runtime
Baseline	92.98%	12.87ms
MPC-CBF, $K_r = 1$	90.87%	46.98ms
MPC-CBF, $\delta = 0.05$s	97.86%	118.17ms
MPC-CBF, $\delta = 0.1$s	98.01%	68.98ms
MPC-CBF, $\delta = 0.2$s	97.98%	57.57ms

Table 4.3: The percentage of neighbors in FoV and the runtime of baseline and MPC-CBF as the HOCBF constraint sample intervals change. The statistics are averaged over 15 trials.

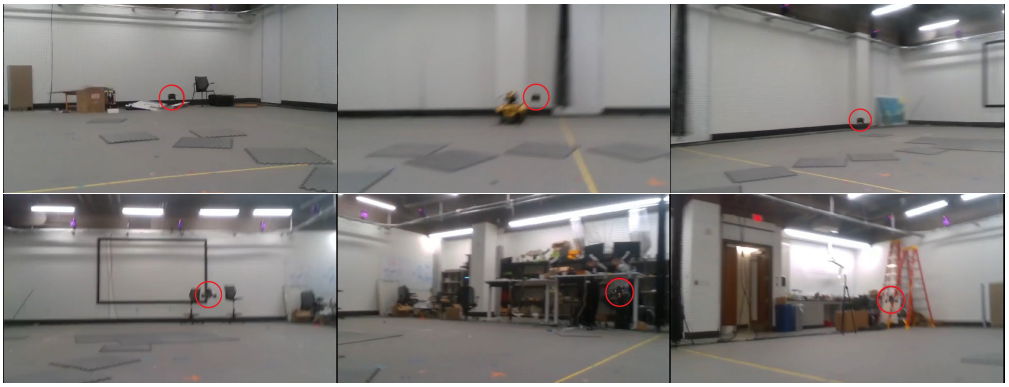
Detection Noises and Delays. We evaluate the efficacy of our algorithm under noisy and delayed detections. We use the same experiment setup as the above 4 robots experiment. A Gaussian noise $\mathcal{N}(\mathbf{0}, \sigma^2_{\text{noise}}\mathbf{I})$ is added to the detected neighbor state. From Table 4.4, we notice that the *Percentage of Neighbors in FoV* decreases as delay and noise increase. Despite performance degradation, our algorithm maintains 100% success rate on the 4 robots experiment.

Setup	Percentage of Neighbors in FoV	Success Rate
Delay = 0.05s	91.52%	100.00%
Delay = 0.1s	84.85%	100.00%
Delay = 0.2s	74.23%	100.00%
$\sigma_{\text{noise}} = 0.1$	98.17%	100.00%
$\sigma_{\text{noise}} = 0.2$	97.29%	100.00%

Table 4.4: The percentage of neighbors in FoV and the success rate as the detection delay and noise change. The statistics are averaged over 15 trials.

4.3.9 Physical Experiment

We build PX4-based UAVs and track their position and orientation with Vicon during the experiment. The motion capture can be replaced by onboard state estimation, such as GPS and a magnetometer during outdoor flights or VIO/LIO during indoor flights, providing a communication-free setup for localization. The UAV is equipped with a Jetson Xavier and a RealSense D435 camera, where $\beta_H \approx \frac{1}{4}\pi$. We model the quadrotor as a double integrator and send position, velocity, and acceleration commands to its onboard flight controller. This model mismatch may violate forward invariance in physical experiments. The quadrotor model requires a third-order CLF-CBF [132], which we leave for future work. We enforce field-of-view constraints in the x-y plane, noting they can be extended to 3D by adding constraints in the x-z plane. We use onboard Apriltag detection in a single-robot experiment, shown in the accompanying video. In the multi-robot experiment, due to unstable Apriltag detection, we simulate detection capabilities using Vicon. In other words, the robot receives the relative location (detection) of its neighbor



(a) Time = 4.2s **(b)** Time = 10.8s **(c)** Time = 14.3s

Figure 4.24: The visual contact constraints are satisfied throughout the flight. The other robot is circled in red.

from Vicon when that neighbor is in its sensing region. We note that relying on Vicon deviates from the decentralized assumption. Onboard detection (e.g., [133]) would introduce noise, delay and missed detections. We discussed the influence of noise and delay in Sec. 4.3.8. Our algorithm handles the missed detections as discussed in Sec. 4.3.3. We demonstrate visual contact during navigation with a two-robot experiment: two UAVs maintain visual contact with each other at all times, as shown in Fig. 4.24.

4.3.10 Conclusion

We address online distributed coordination without communication. Robots navigate to their goals while estimating the locations of neighbors by maintaining visual contact. The proposed strategy is robust to temporary tracking loss and able to regain tracking. We propose MPC-CBF, a discrete optimization framework to approximate the certified solution. In addition, we propose an efficient SQP to solve MPC-CBF with a QP solver. We verify the efficacy and scalability of our algorithm with 10 robots in simulation and physical experiments with 2 custom-built UAVs with onboard cameras. In future work, we aim to extend to non-planar motion, and develop an adaptive controller that is resilient to model perturbation caused by external forces, such as downwash [134].

4.4 Dual-Layer Control Architecture for Cooperative Environmental Monitoring in Multi-Robot Systems

While previous sections focused on maintaining connectivity for data exchange, this section addresses a fundamental sensing challenge: anisotropic target tracking. Unlike the communication maintenance explored earlier, where intermittent links were sufficient, target tracking with limited-field-of-view sensors requires continuous, cooperative observation to resolve position ambiguity. To address this, we propose a strategy that pairs agents into coordinated sub-groups. By enforcing precise geometric formations within these local clusters, agents can triangulate targets that would be unobservable or poorly localized by a single sensor, thereby bridging the gap between loose connectivity and precise cooperative sensing.

4.4.1 Introduction

In the majority of applications, UAVs serve as mobile sensors whose job is to gather data, primarily in the form of camera images. However, sensors' non-ideality arises some issues in deploying UAVs in a real environment, particularly when they are equipped with low-cost hardware platforms. In this work, we consider UAVs only equipped with a monocular camera, being assigned with the task of monitoring the environment in order to detect and localize events of interest. More in details, the use of monocular cameras does not yield information regarding the distance of detected targets; thus a suitable strategy needs to be designed for target localization.
The solution we propose in this Section aims at exploiting cooperation among UAVs to monitor the environment. Specifically, we define sub-groups of the entire team, each one composed by n UAVs capable of collectively localizing a target combining their image data. The control architecture is designed as a combination of formation control for coordination of robots within the same sub-group, and coverage control for coordination among the whole team. In the following sections, we describe how a two-level semi-centralized controller is designed to achieve the mentioned behavior, and we present the experimental evaluation we conducted to test the effectiveness of the solution.
To summarize, we can formulate the two-fold contribution of our work as follows:

- Design of a formation control algorithm based on Control Barrier Functions (CBFs) and Controly Lyapunov Functions (CLFs) to keep robots in a desired formation, allowing for formation adjustments to avoid collisions with obstacles.
- Design of a cooperative coverage control to provide coverage of areas of interest from multiple UAVs, in order to combine their monocular camera images to extract the position of possible targets within the region.

4.4.2 Related Works

Monocular and Cooperative Target Tracking

The problem of target tracking with a monocular camera has been addressed in [135], where authors propose a solution to retrieve the position of a target based on pinhole imaging, allowing to map three-dimensional objects to a two-dimensional plane, in combination with a geometric model to find the relationship between pixels in the image and points in the real world. Instead, the authors in [136] propose a target localization strategy to retrieve depth data from sequential images. By exploiting an Extended Kalman Filter for robot localization, this solution allows to triangulate the target's position by observing it repeatedly from different spots. A different approach has recently emerged with the use of Deep Learning techniques, as pointed out in [137]: recent considerable progress in learning-based techniques, specifically Computer Vision, has brought a wide range of solutions to the problem of depth estimation, typically outperforming traditional solutions. In addition, learning-based methods have been also applied to cooperation settings. An example is the solution in [138], which leverages Graph Neural Networks to select the control action from local observations and communication only, thus making it suitable for large-scale scenarios.

Formation Control

Formation control is another extensively studied topic within the multi-robot research community. A distributed solution is proposed in [139] to find collision-free trajectories via constrained optimization, allowing for reconfiguration. A similar approach is also used in [140] for collaborative object transport. Artificial potential fields can also be employed for this purpose. An example is [141], where their employment allows a multi-robot system to be shaped as an arbitrarily defined polygon. Finally, already mentioned Deep Learning has also been applied to this scenario to train robots to organize into formations while avoiding collision with other agents and obstacles [142].

4.4.3 Problem Description

Consider a team of $N \in \mathbb{N}$ aerial robots flying at the same constant altitude $\bar{z} \in \mathbb{R}^+$, thus moving in a 2D environment $\mathcal{Q} \subset \mathbb{R}^2$ possibly populated with obstacles. We denote as $\mathcal{O} = \{\mathbf{o}_1, \ldots, \mathbf{o}_{N_{\text{obs}}}\} \subset \mathbb{R}^2$ the positions of $N_{\text{obs}} \in \mathbb{N}$ obstacles. For the sake of simplicity, following the discussion in [143], we assume the UAVs move slowly enough to neglect dynamic behaviors, following the single integrator control law $\dot{\boldsymbol{\chi}}_i = \mathbf{u}_i$ where $\boldsymbol{\chi}_i = [x_i, y_i, \theta_i]^\top$ is the state of the i-th robot, and $\mathbf{u}_i \in \mathbb{R}^3$ represents the vector of first-order input control, encompassing both linear and angular velocities. We assume the team is split into $C \in \mathbb{N}$ clusters of $n \in \mathbb{N}$ UAVs, and each UAV is statically assigned to a specific cluster from the beginning of the task execution. For simplicity, we assume N is a multiple of n, ensuring $C = N/n \in \mathbb{N}$. Then, we denote as $\mathbf{G}_k \in \mathbb{R}^2$ the centroid of the k-th cluster, and we define a virtual agent $\boldsymbol{\psi}_k \in \mathbb{R}^2$ to be tracked by $\mathbf{G}_k$, as

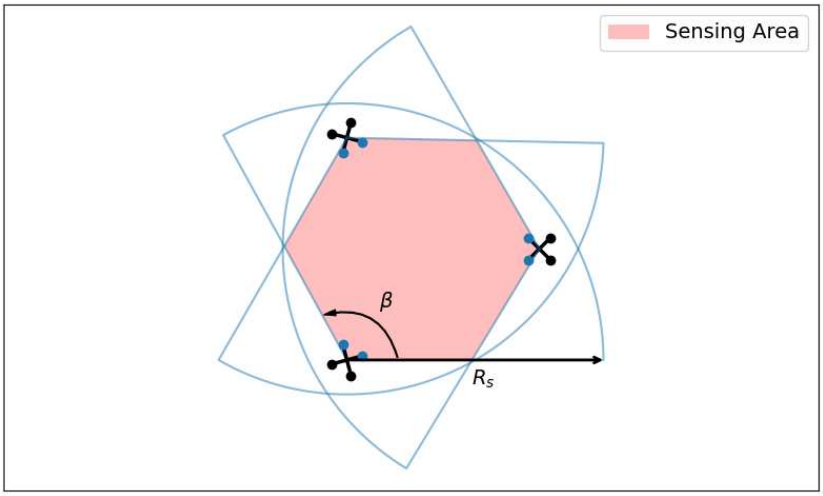


Figure 4.25: Sensing model: the sensing area of the cluster results from the intersection of each UAV's field of view.

we will detail in the following sections. Consequently, $\mathscr{P} = \{\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_C\}$ will refer to the set of virtual agents' positions. We consider robots with communication capabilities within a communication range $R_c > 0$, and sensing capabilities by means of an on-board monocular camera. For simplicity, the area on the ground captured by the camera is modeled as a circular sector with radius $R_s > 0$ and amplitude $\beta > 0$. However, the use of a monocular camera does not allow feature localization on the ground, necessitating cooperative triangulation among robots in a cluster. Therefore, we consider the sensing region of a cluster as the intersection of the fields of view of the UAVs belonging to that sub-group, as depicted in Fig. 4.25. In this region, features can be detected by each single UAV and localized through cooperation among them. We assume robots are equipped with odometry sensors for self-localization, and we will indicate as $\mathbf{p}_j$ the position of robot j relative to the common global reference frame, while ${}^i\mathbf{p}_j$ will refer to the position of j relative to the inertial reference frame of robot i. Finally, we will use the notation $\mathbf{p}_j^k$ to indicate that robot j is a member of the k-th cluster.

4.4.4 Robots Control Layer

The inner control layer governs the UAVs within a generic k-th cluster to track the virtual agent $\boldsymbol{\psi}_k$, whose motion is detailed in the following section. Specifically, UAVs within a cluster are required to keep their virtual agent centered in their field of view and arrange themselves such that the cluster centroid coincides with the virtual agent. To simplify the discussion, in the remainder of the Section we consider a constant number of UAVs per cluster, specifically $n = 3$. These requirements translate into a desired formation for cluster k, where UAVs are located on a circle of radius $R_s/2$, angularly spaced by 120°. This configuration allows the UAVs to frame the same area from different angles, ensuring complete coverage for the cooperative detection and localization of targets. Additionally, robots are required to avoid collisions with obstacles and other agents while maintaining communication connectivity with members of their own cluster.

CBFs for Safety Constraints

The requirements described above are enforced using Control Barrier Functions (CBFs) and Control Lyapunov Functions (CLFs) presented in Section 2.2. First, for each drone i in the k-th cluster, we define the following barrier functions $\mathbf{h}(\cdot)$ with respect to every neighbor $j \neq i$ in the same cluster:

$$\mathbf{h}({}^i\mathbf{p}_j^k) = \begin{bmatrix} h_1({}^i\mathbf{p}_j^k) \\ h_2({}^i\mathbf{p}_j^k) \end{bmatrix} = \begin{bmatrix} \|{}^i\mathbf{p}_j^k\|^2 - D_s^2 \\ -\|{}^i\mathbf{p}_j^k\|^2 + R_c^2 \end{bmatrix}. \tag{4.48}$$

The first component quantifies the margin between the inter-robot distance and a safety distance $D_s \in \mathbb{R}^+$; the second component compares the distance against the communication range R_c. Imposing $\mathbf{h}(\cdot) \geq \mathbf{0}$ constrains the distance between i and j to remain above the safety threshold but within the communication range. Similarly, we define two additional components to ensure collision avoidance with robots from other clusters $c \neq k$ and every obstacle $\mathbf{o} \in \mathcal{O}$:

$$\begin{bmatrix} h_3({}^i\mathbf{p}_j^c) \\ h_4({}^i\mathbf{o}_j) \end{bmatrix} = \begin{bmatrix} \|{}^i\mathbf{p}_j^c\|^2 - D_s^2 \\ \|{}^i\mathbf{o}_j\|^2 - D_s^2 \end{bmatrix}. \tag{4.49}$$

We then define the class $\mathcal{K}$ function $\alpha(\cdot)$ as:

$$\alpha(\mathbf{h}(\cdot)) = \gamma \mathbf{h}(\cdot), \tag{4.50}$$

where $\gamma \in \mathbb{R}^+$ is a positive constant. For a generic distance-based constraint h, the time derivative can be expressed as:

$$\dot{h}({}^i\mathbf{p}_j) = \frac{\partial h({}^i\mathbf{p}_j)}{\partial {}^i\mathbf{p}_j} {}^i\dot{\mathbf{p}}_j = \begin{bmatrix} -2{}^ix_j & -2{}^iy_j \end{bmatrix} {}^i\dot{\mathbf{p}}_j = \mathbf{A}_c {}^i\dot{\mathbf{p}}_j. \tag{4.51}$$

Since we assume the UAVs move slowly with respect to the control frequency, we treat the relative motion of robot j with respect to i as negligible for the control derivation. Furthermore, considering static obstacles implies $\dot{h}_4(\cdot) = 0$. Thus, we can approximate (4.51) as:

$$\dot{h}({}^i\mathbf{p}_j) \approx \mathbf{A}_c {}^i\mathbf{u}_i. \tag{4.52}$$

Finally, the optimization problem (2.13) is formulated as:

$$\begin{aligned} \mathbf{u}_i^* = \underset{\mathbf{u}_i \in \mathbb{R}^3}{\text{argmin}} \quad & \frac{1}{2}\|\mathbf{u}_i - \mathbf{u}_{\text{ref}}\|^2 \\ \text{s.t.} \quad & \mathbf{A}_c {}^i\mathbf{u}_i \geq -\gamma \mathbf{h}(\cdot). \end{aligned} \tag{4.53}$$

CLFs for Desired Configuration

After defining the set of safety constraints that robots must satisfy to avoid collisions with obstacles and other robots, and to preserve communication with neighbors, we define how the system is controlled to reach the desired configuration. This requirement translates

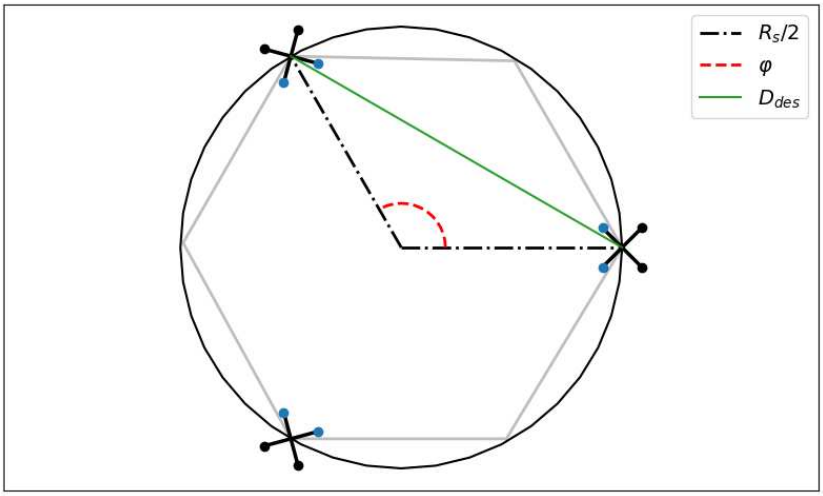


Figure 4.26: Desired distance D_{des} among robots, calculated as the chord linking two points with angular distance φ located on a circumference with radius $R_s/2$.

to an objective distance that each member of the k-th cluster should maintain from one another. Specifically, $D_{\text{des}} \in \mathbb{R}^+$ corresponds to the chord length linking two points on a circle of radius $R_s/2$ with an angular separation of $\varphi = 120°$:

$$D_{\text{des}} = 2\frac{R_s}{2}\sin\frac{\varphi}{2} = R_s \sin\frac{\varphi}{2}. \tag{4.54}$$

We then define the Control Lyapunov Function (CLF) as:

$$\nu({}^i\mathbf{p}_j^k) = (\|{}^i\mathbf{p}_j^k\| - D_{\text{des}})^2. \tag{4.55}$$

It is straightforward to verify that ν is a positive definite and continuously differentiable function, making it a suitable CLF candidate. Following the same logic applied to the CBFs, the time derivative is given by:

$$\dot{\nu}({}^i\mathbf{p}_j^k) = \frac{\partial \nu({}^i\mathbf{p}_j^k)}{\partial {}^i\mathbf{p}_j^k}{}^i\dot{\mathbf{p}}_j^k \approx \mathbf{B}_c{}^i\mathbf{u}_i. \tag{4.56}$$

Stabilization to the desired state is achieved by defining the extended class $\mathcal{K}$ function $\zeta(\nu(\cdot))$ as:

$$\zeta(\nu(\cdot)) = \kappa\nu(\cdot), \tag{4.57}$$

where $\kappa \in \mathbb{R}^+$ is a positive constant. Integrating the constraint into the optimization problem (4.53), the problem evolves into:

$$\begin{aligned} \mathbf{u}_i^* = \underset{\mathbf{u}_i\in\mathbb{R}^3,\varepsilon\in\mathbb{R}^+}{\text{argmin}} \quad & \frac{1}{2}\|\mathbf{u}_i - \mathbf{u}_{\text{ref}}\|^2 + p\varepsilon^2 \\ \text{s.t.} \quad & \mathbf{A}_c{}^i\mathbf{u}_i \geq -\gamma\mathbf{h}(\cdot), \\ & \mathbf{B}_c{}^i\mathbf{u}_i \leq -\kappa\nu(\cdot) + \varepsilon. \end{aligned} \tag{4.58}$$

Note that a slack variable $\varepsilon \in \mathbb{R}^+$ (penalized by weight p) has been introduced into the CLF constraint. This relaxation prioritizes strict adherence to safety constraints

(hard constraints) while allowing temporary deviations from the desired formation (soft constraint) to avoid collisions with obstacles and other robots.

Finally, we define the desired reference control input $\mathbf{u}_{\text{ref}}$. As previously mentioned, the desired velocity drives each robot such that the centroid of the k-th cluster tracks a virtual agent $\boldsymbol{\psi}_k$, whose motion is detailed in the following section. Let $\mathbf{G}_k \in \mathbb{R}^2$ be the centroid of cluster k, whose position can be calculated independently by each UAV within the sub-group thanks to the continuous communication channel ensured by the CBFs. Approximating the cluster as a rigid body, we calculate the linear velocity of each UAV as the linear velocity required for $\mathbf{G}_k$ to reach $\boldsymbol{\psi}_k$. Conversely, the angular velocity is calculated to keep $\boldsymbol{\psi}_k$ centered in the field of view. Hence, the desired reference velocity relative to the inertial reference frame of robot i is formulated as:

$$
{}^{i}\mathbf{u}_{\text{ref}} = K_p \begin{bmatrix} {}^{i}\psi_{k,x} - {}^{i}G_{k,x} \\ {}^{i}\psi_{k,y} - {}^{i}G_{k,y} \\ \arctan({}^{i}\psi_{k,y} / {}^{i}\psi_{k,x}) \end{bmatrix}, \tag{4.59}
$$

where $K_p \in \mathbb{R}^+$ is a proportional gain.

4.4.5 Clusters Coordination Layer

As previously mentioned, the overall control strategy employs a two-level approach: the lower level coordinates UAVs within a cluster, while the higher level coordinates the clusters themselves. Having presented the lower control level, this section details the higher level responsible for moving the virtual agent $\boldsymbol{\psi}_k$. The proposed method is semi-centralized; only one member of cluster k calculates and communicates $\boldsymbol{\psi}_k$ to the other members. Although every robot is capable of computing the virtual agent's control, a centralized designate ensures consistency among members of the same cluster, preventing mismatches that could arise from decentralized calculation. Furthermore, the existence of a communication link among intra-cluster robots is guaranteed by the CBF constraints defined in the previous section.

Our approach adapts standard Voronoi-based coverage control to the virtual agent framework. We exploit the Voronoi algorithm to partition the environment based on the virtual agents' positions $\boldsymbol{\psi} \in \mathscr{P}$. Since UAVs lack global knowledge, we employ the limited-range decentralized partitioning presented in Section 2.1.1. The resulting generic Voronoi cell $V_k \subset \mathcal{Q}$ comprises regions of the environment closer to the k-th virtual agent than to any other, restricted to the sensing area of $\boldsymbol{\psi}_k$. Since UAVs asymptotically converge to the configuration detailed in the previous section, locating themselves on a circle centered at $\boldsymbol{\psi}_k$ with radius $R_s/2$, we approximate the virtual sensing region of a generic cluster as the closed ball $B(\boldsymbol{\psi}_k)$ centered at $\boldsymbol{\psi}_k$ with radius $R_s/2$. The formulation for the Voronoi cell from (2.7) becomes:

$$
V_k = \{\mathbf{q} \in \mathcal{B}(\boldsymbol{\psi}_k) \mid \|\mathbf{q} - \boldsymbol{\psi}_k\| \leq \|\mathbf{q} - \boldsymbol{\psi}_j\|, \forall \boldsymbol{\psi}_j \in \mathscr{P} \setminus \{\boldsymbol{\psi}_k\}\}. \tag{4.60}
$$

Assuming robots from different clusters can exchange the positions of their virtual agents,

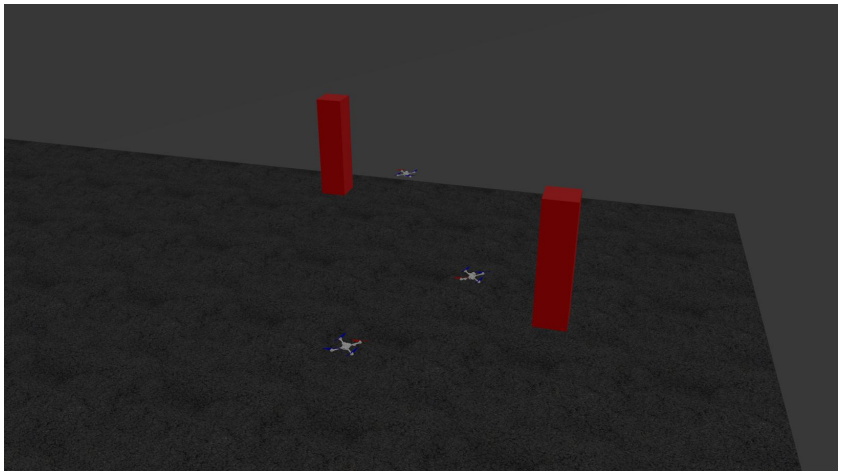


Figure 4.27: Gazebo virtual environment with 3 UAVs and 2 obstacles.

the limited Voronoi cell V_k can be constructed using locally available information. Following traditional coverage control, the control input for $\boldsymbol{\psi}_k$ can be computed according the proportional law:

$$\mathbf{u}_{\boldsymbol{\psi}_k} = K_p(\mathbf{C}_{V_k} - \boldsymbol{\psi}_k), \tag{4.61}$$

which iteratively drives $\boldsymbol{\psi}_k$ towards the centroid of its Voronoi region. This eventually leads the system to a Centroidal Voronoi Configuration (CVC), optimizing the total area covered. The position of $\boldsymbol{\psi}_k$ is continuously shared with other members of cluster k, enabling each robot to calculate its own desired reference input via (4.59).

4.4.6 Experimental Evaluation

Here, we present the experimental evaluation of our two-level control strategy. The validation was conducted in two phases: first, we evaluated the behavior of a single cluster to verify that the designed Control Barrier Functions (CBFs) and Control Lyapunov Functions (CLFs) successfully achieved the desired formation while ensuring collision avoidance. In the second phase, we assessed the overall performance of the full multi-robot team and the coordination between clusters, measuring the coverage effectiveness of the system in complex environments.

Single-Cluster Tests

In the first set of experiments, we controlled a single cluster composed of 3 UAVs to cooperatively track a virtual agent while maintaining a desired formation and avoiding obstacles. The virtual simulation environment was established using Gazebo, with virtual UAVs provided by the RotorS simulation framework [111] and controlled via ROS middleware. Obstacles were instantiated within the environment (see Fig. 4.27) at randomly generated locations for each simulation run. Similarly, UAVs were initialized at random locations within the safe set $\mathcal{C}$, ensuring that initial inter-agent distances were greater than the safety threshold but within the communication range, thereby satisfying the initial CBF constraints.

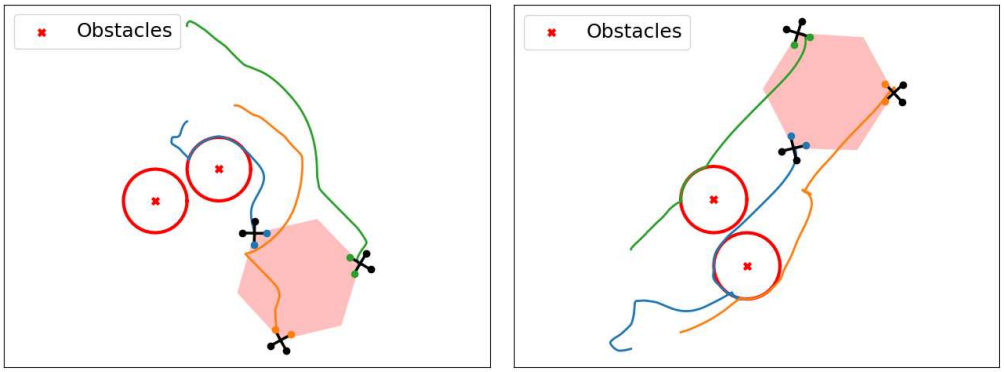


Figure 4.28: Trajectories of 3 UAVs achieving a desired formation while maintaining safety distances from obstacles.

The safety parameters were set to $D_s = 2.0\,\text{m}$ and $R_c = 10.0\,\text{m}$. The sensing field of view was modeled as a circular sector with radius $R_s = 10.0\,\text{m}$ and angular amplitude $\beta = 120°$. The weights in the optimization problem (4.58) were set to $\gamma = 5.0$ and $\kappa = 0.1$ to prioritize safety over formation maintenance, with the slack variable penalty weight set to p (implicitly handling ε). For this phase, the robots were tasked with covering an environment where the probability density function $\phi(\mathbf{q})$ was defined as a simple bivariate Gaussian distribution. This distribution was characterized by a mean vector $\boldsymbol{\mu} \in \mathbb{R}^2$, randomly generated for each run, and a covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{2\times2}$, fixed as the identity matrix.

While controlling a single cluster against a single Gaussian target trivially results in the virtual agent converging to the mean $\boldsymbol{\mu}$, this phase was crucial for validating the low-level control stack. The tests demonstrated the UAVs' ability to achieve the desired formation while strictly maintaining safety distances. As depicted in Fig. 4.28, the robots consistently attempt to maintain the desired geometric configuration during motion, modifying their formation only when safety constraints require obstacle avoidance.

The analysis of the CBF and CLF values during task execution, shown in Fig. 4.29, offers further insight. The constraint inequalities from optimization problem (4.58) are visualized as the residuals $\dot{h}(\cdot) + \gamma h(\cdot)$ and $-\dot{\nu}(\cdot) + \kappa\nu(\cdot) + \varepsilon$. For clarity, only the CBFs related to obstacle avoidance and the corresponding CLFs are plotted. The upper plot demonstrates that the CBF constraint is consistently satisfied, as the value never drops below zero. Conversely, the CLF curves in the lower plot remain near zero for the majority of the trajectory but are permitted to become negative when the robot must navigate around an obstacle. This behavior reflects the control design: the cluster is allowed to temporarily deviate from the intended formation (violating the soft CLF constraint) to ensure hard safety constraints are never compromised.

Multi-Cluster Coordination

Following the validation of intra-cluster coordination and collision avoidance, we conducted a second set of tests to evaluate the inter-cluster coordination and the global performance of the multi-robot system. The simulation setup remained consistent with

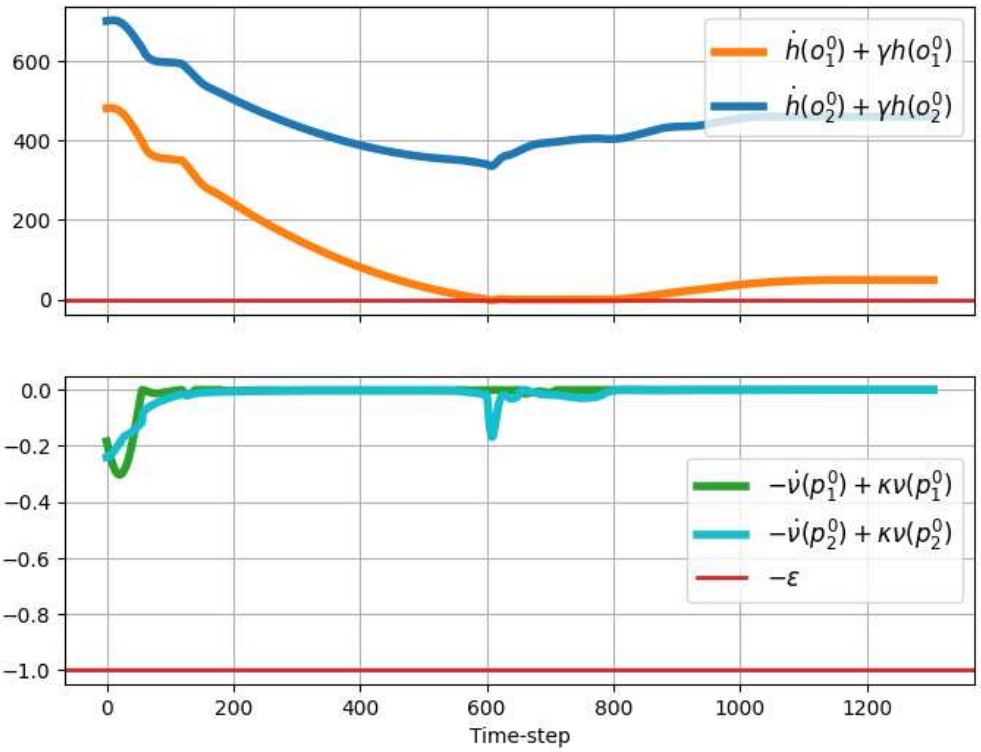


Figure 4.29: Values of the CBF and CLF constraints during task execution.

the previous phase, with the exception of the probability density function $\phi(\mathbf{q})$, which was modeled as a Gaussian Mixture Model (GMM) or a sum of bivariate Gaussian distributions to represent more complex environments.

In this setup, a leader UAV is designated within each cluster to compute the motion of the virtual agent and broadcast its position to neighbors. Communication is restricted to the range $R_c = 10.0\,\text{m}$. Agents from different clusters ($j \neq k$) exchange information regarding their respective virtual agents' positions, while agents within the same cluster k share gathered sensing data.

Figure 4.30 illustrates the trajectories and final configurations of the robots across several scenarios. In Fig. 4.30a and Fig. 4.30b, $\phi(\mathbf{q})$ is defined as a GMM, creating continuous regions of high event probability. In these scenarios, the UAVs move to cooperatively monitor the area of interest; their final positions represent a fast-equilibrium between the collective coverage objective and the inter-agent safety constraints.

In contrast, Fig. 4.30c presents a scenario where $\phi(\mathbf{q})$ is composed of four distinct Gaussian components. As the number of clusters equals the number of Gaussian peaks, the system naturally partitions such that each cluster monitors a specific component. Finally, an emergent behavior is observed in Fig. 4.30d, where $\phi(\mathbf{q})$ is defined as a single bivariate Gaussian distribution. Here, all three clusters are attracted to the same region. Consequently, the robots arrange themselves into a single global formation, redundantly facing the same region. This behavior suggests that for highly concentrated distributions, the system effectively merges into a larger "super-cluster," approximating a larger circular formation. This reflects a high degree of certainty regarding the target location, prompting the entire swarm to focus its resources on that specific area.

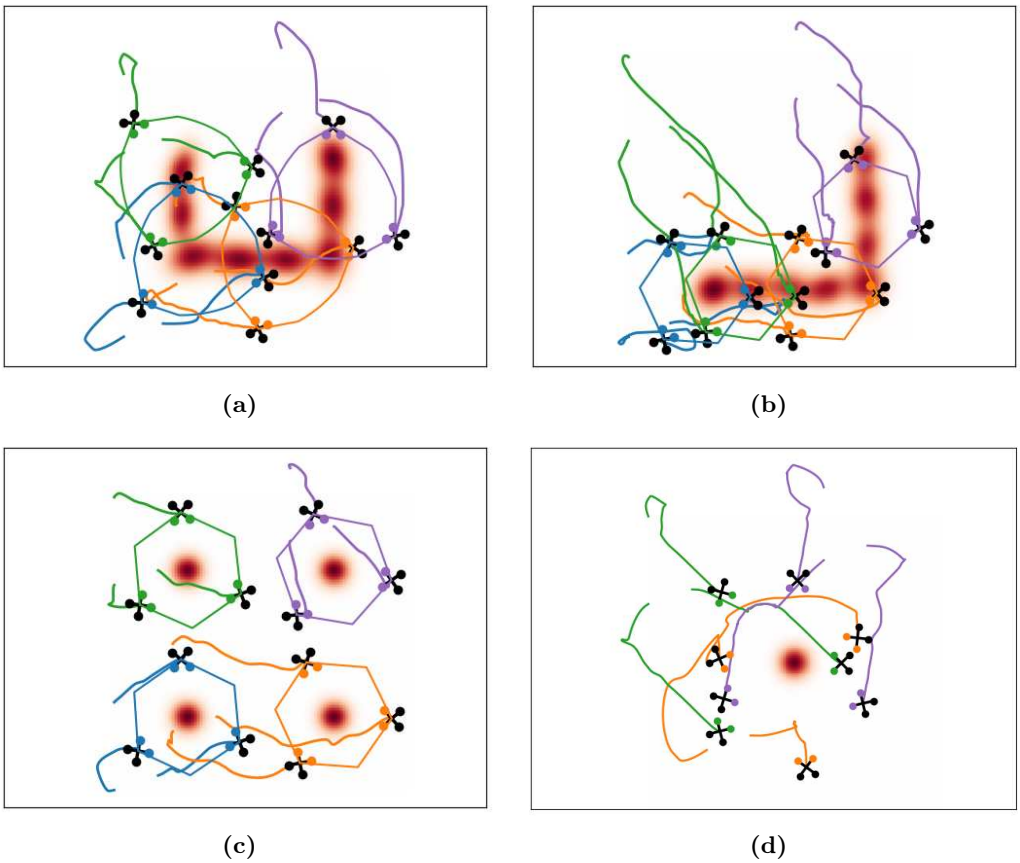


Figure 4.30: Results in different scenarios where $\phi(\mathbf{q})$ is defined as: (a, b) a Gaussian Mixture Model; (c) a sum of 4 Gaussian distributions; and (d) a single Gaussian function.

4.4.7 Conclusion

In this study, we have introduced a control methodology for a multi-robot system operating in unknown environmental conditions, ensuring a safety mechanism for collisions among drones and against potential obstacles. It guarantees formation-keeping within individual clusters of agents and enables cluster-level control to accomplish higher-level tasks–in this case, the surveillance of a moving agent. The adaptation to various conditions in simulations demonstrates the control method's capability to complete tasks even in the presence of obstacles. The use of the control barrier function as a safety constraint allows for the management of environmental stresses without collaterals to the swarm's configuration control or cluster control. We believe these results could pave the way for further real and practical applications, for example, where environmental stresses are caused not only by the presence of obstacles but also by atmospheric conditions such as the effect of wind on drones. Future work could also address the generalization of the number of drones, which in this study was limited to three due to numerical constraints, and a strategy for dynamic cluster allocation that would allow for automation in the case of a large swarm and a high number of points of interest to monitor.

Chapter 5

Enhancing Decentralized Control via Learning-Based Methods

In this chapter, we explore how machine learning techniques can enhance the adaptability, efficiency, and robustness of robotic systems across various operational scenarios. We begin by addressing surveillance tasks in Section 5.1, where Convolutional Neural Networks (CNNs) are employed to automatically extract road networks from aerial imagery, generating density maps that guide UAVs within a coverage control framework. Building on this visual perception capability, Section 5.2 introduces a learning-based control strategy that leverages CNNs to map local observations, including density, occupancy, and neighbor positions, directly to control commands, enabling effective coverage even in complex, non-convex environments. To address the challenges of state estimation under intermittent observations, Section 5.3 proposes a neural network architecture that mimics a particle filter, providing computationally efficient state and covariance estimation to support safe flocking via Control Barrier Functions (CBFs). Shifting focus to the monitoring of unknown spatial phenomena, Section 5.4 investigates the use of Gaussian Processes (GPs) within an ergodic control framework, balancing exploration and exploitation to adaptively refine the model of the underlying process. Finally, Section 5.5 presents a framework for single-robot safety under modeling uncertainty, utilizing Neural Ordinary Differential Equations (NODEs) to approximate dynamic residuals and integrate them into CBF constraints, laying the groundwork for future multi-robot extensions.

5.1 Distributed Multi-Robot Control for Streets Surveillance from Aerial Images with Neural Networks

We propose a distributed strategy for aerial robot teams to maximize street coverage in monitoring and search-and-rescue missions. A neural network extracts roads from aerial images, generating a probability density via Gaussian Mixture Models (GMMs) to guide robots toward detected streets. Each robot performs a Voronoi-based coverage mission, focusing on areas of interest. The neural network's performance was validated using Google Maps images, while the control architecture was tested in realistic simulations, demonstrating effectiveness.

5.1.1 Introduction

In this work we employ a team of UAVs to address the problem of streets surveillance. A similar problem was addressed in [144], where the authors propose a method to solve a formation control problem for monitoring an expanding flood area, splitting the team into two subgroups, one external and the other one internal, with the aim of tracking the boundary of the area of interest and monitoring the region itself, respectively. Monitoring tasks are often faced exploiting neural networks, whose employment is nowadays widely spread in nearly all fields of research, not just in robotics. An example is the work proposed in [145], where a U-Net, that is a convolutional neural network specially developed for image segmentation [146], is trained with the aim of detecting anomalies while a task is being executed by a robot. In [147] instead a multi-robot system exploits neural networks in order to gain information from the environment and coordinate with each other in order to accomplish a surveillance mission.
In this work we propose a distributed control strategy for a team of UAVs to reach a desired region of the environment and maximize its coverage. More in details, we consider the task of monitoring the streets in a selected area, moving robots above them and spreading in order to maximize the target surface. We make use of a convolutional neural network, the widely known U-Net, trained to extract roads from satellite images. Then, a proper probability density is defined fitting a Gaussian Mixture Model to the mask generated by the neural network, in order to define a higher probability in areas over the streets where UAVs are required to move. A similar strategy was adopted in [148], where neural networks are exploited in combination with Gaussian Mixture Models to accomplish a search and rescue task with a team of UAVs. However, the authors only consider curve-shaped 1-D target areas such as coastlines or streets modelling a 1-D Gaussian Mixture Model, forcing the team to assume a linear configuration, which may not be ideal for a surveillance task. Gaussian Mixture Models are also employed in [149] to model the probability distribution of the target's location and optimize path planning for a single UAV.
In our work, robots are controlled in a decentralized manner to perform Voronoi-based coverage, being attracted where the probability density is higher, thus achieving the goal of the mission. The proposed solution is finally tested selecting an area from Google Maps and generating high fidelity large-sized virtual environments (see Fig. 5.1), evalu-

Figure 5.1: Realistic simulation of the area to be monitored.

ating performances in terms of covered surface.

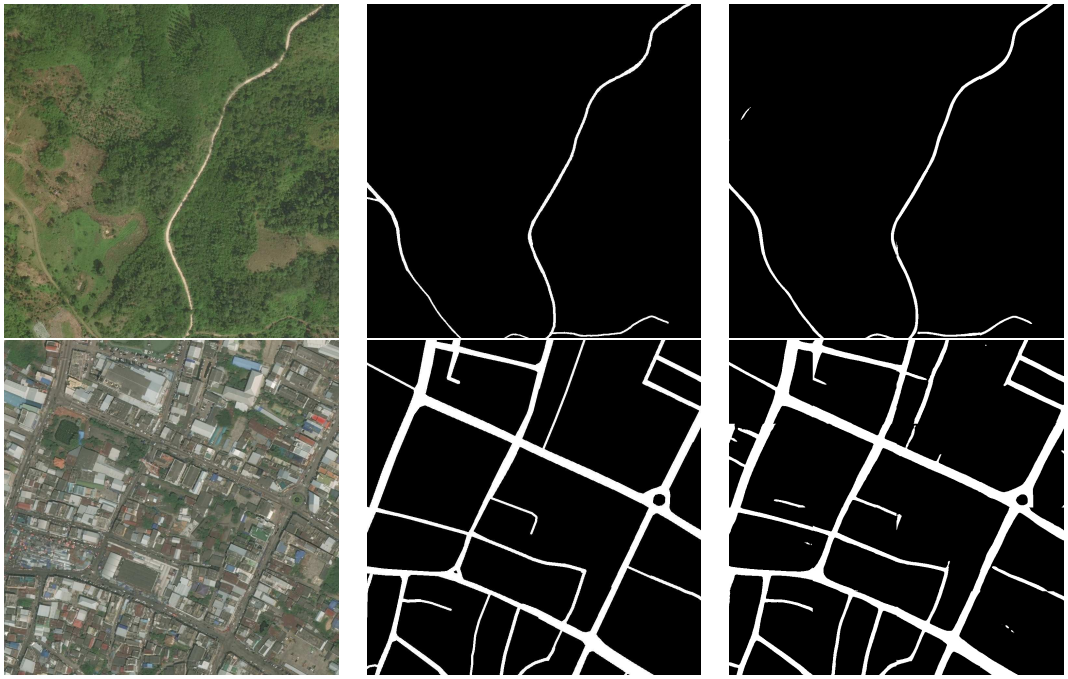
The contribution of this research is then the definition of an integrated control strategy that, starting from aerial images, is able to automatically detect the areas of interest (i.e., the roads) and subsequently deploy a fleet of UAVs in a way that optimizes the overall capability of observing the environment. The proposed methodology combines a convolutional neural network for image analysis, Gaussian Mixtures Models for describing the relative importance of different areas, and Voronoi-based coverage control for UAV deployment.

5.1.2 Problem Description

Consider a team of $n \in \mathbb{N}$ aerial robots flying at the same constant altitude, thus moving in a 2D environment. The altitude is assumed to be sufficient to avoid obstacles on the ground. We assume each robot to be modeled as a single integrator system, whose position $\mathbf{p}_i \in \mathbb{R}^2$ evolves according to $\dot{\mathbf{p}}_i = \mathbf{u}_i$, where $\mathbf{u}_i \in \mathbb{R}^2$ is the control input, $\forall i = 1, \ldots, n$. We consider the following assumptions:

- *Localization*: each robot is able to localize itself with respect to a global reference frame, shared among robots within the team.
- *Limited Sensing Capabilities*: each robot is able to detect and localize neighbors inside its limited sensing range, defined as a circle $B(\mathbf{p}_i, r)$.

Based on these assumptions, the problem addressed in this work can be described as the implementation of a control architecture for street surveillance employing a team of UAVs with limited sensing capabilities. The goal is to provide a straightforward workflow that enables a human operator to choose a location where the task must be carried out, and then operate the robotic team to maximize the monitored portion of the regions of interest.



(a) (b) (c)

Figure 5.2: Prediction test: (a) input image, (b) desired mask, (c) prediction of the trained U-Net.

5.1.3 Model Training

Here, we present the architecture of the neural network employed to extract roads from satellite images, and we detail its components and the strategy adopted for training and evaluation of the results.
As already mentioned, the neural network's architecture is modeled after the well-known U-Net [146], a convolutional neural network specially developed for image segmentation, thus perfectly suiting our use case. In the following, we will briefly present the structure of the network: for additional details, the reader is referred to [146]. First of all, an appropriate dataset has been retrieved from the DeepGlobe Road Extraction Challenge [150], whose goal was to advance research on roads extraction in order to enhance crisis response in disasters zones, especially in developing countries. This dataset contains 6226 satellite images, each one paired with a mask image for road labels.

Architecture of the U-Net

The developed network accurately reproduces the standard structure of a U-Net, consisting of a contracting path taking an image with shape $256 \times 256 \times 3$ and an expansive path, returning an image with the same shape as the input. The contracting path follows the typical architecture of a convolutional network, containing encoder layers that progressively reduce the input size while increasing the number of channels, in order to capture high-level features. Each encoder layer consists of the application of two 3×3 convolutions, each one followed by a rectified linear unit (ReLU) and a 2×2 max pooling operation for downsampling. The expansive pathway subsequently transforms the

(a) (b) (c)

Figure 5.3: Fitting a Gaussian Mixture Model to a satellite image: (a) input image; (b) predicted mask; (c) heatmap of the probability density generated by the GMM, with darker colors corresponding to higher values.

feature map obtained from the contracting path into an image of identical dimensions as the initial input. Every layer in the expansive path involves an upsampling, which reduces the number of channels while increasing the resolution of the feature map up to the shape of the input image. In addition, skip connections from the contracting path are exploited to allow upsampling layers to find and refine the features in the image. The output image therefore is a binary segmentation map whose shape is the same as the input. Depending on whether it is considered to belong to a street or not, each pixel is either white or black.

The described network was subsequently trained and tested on the aforementioned dataset in order to learn to operate roads extraction even in previously unseen satellite images.

Training and Evaluation

The dataset of labeled images was split into training, testing and validation sets, with a proportion of 70%, 10%, and 20%, respectively. The optimizer chosen for training was Adam [151], an extension to stochastic gradient descent method adaptively estimating first-order and second-order moments. The main parameters for the optimizer were the learning rate $\alpha = 0.0001$ and the exponential decay rates for the first and second moment estimates, namely $\beta_1 = \beta_2 = 0.99$. The loss function was calculated as the sum of the Dice loss [152], widely adopted for pixel segmentation, and the binary focal loss [153]. The Jaccard coefficient, also known as Intersection over Union (IoU), was finally calculated to test the accuracy of predictions on images from the testing set, evaluating how much the binary map generated by the model overlaps the label mask.
After the training phase, labeled images from the validation set were passed as input to the U-Net in order to test its behaviour with previously unseen images. Examples of results are shown in Fig. 5.2, where predicted masks are compared to the labels from the dataset. In addition to the visual results, the efficiency of the trained U-Net is also proved by the Jaccard coefficient, whose mean value on the validation set was 0.61, indicating good performances in the generation of binary masks from new images.

5.1.4 GMM Definition

In our work, we employ GMMs to construct a probability density function that effectively guides the robots into the desired regions as in Section 3.2.3. Specifically, to achieve coverage of street surfaces, we fit a GMM to the areas identified by the trained U-Net. The fitting process involves selecting the optimal set of parameters $(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{w})$ given the input data points. Parameter estimation is performed using the *Maximum Likelihood* method as described in [65], utilizing the *Expectation-Maximization* (EM) algorithm to iteratively converge on the optimal parameters. While the number of Gaussian components k can be manually selected, the algorithm can also determine an optimal number. To this end, the Bayesian Information Criterion (BIC), a likelihood-based metric for model selection, is calculated to compare fits across different model complexities. In our application, we use the foreground pixels (i.e., white pixels in Fig. 5.2c) from the U-Net-generated binary mask as the input samples for the EM algorithm.

Once the GMM is defined, the probability density function is computed as

$$\Phi(\mathbf{q}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{i=1}^{k} w_i \phi_i \left(\mathbf{q}, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right), \tag{5.1}$$

where ϕ_i is the contribution of the i-th Gaussian component given by (3.7). The result of this methodology is a non-uniform density map of the environment, where each point $\mathbf{q} \in Q$ is assigned a high probability value if it lies on a street. An example is illustrated in Fig. 5.3, which shows a binary mask generated by the U-Net from a satellite image, followed by the GMM fitted to the foreground pixels. By assigning high density values to street areas, this strategy creates an attractive field that draws the UAV team into positions directly above the streets, thereby optimizing their vantage points for monitoring tasks.

5.1.5 Distributed UAVs Control

Having detailed the neural network implementation and the generation of the binary mask to identify regions of interest, now we briefly describe the distributed control algorithm for the multi-robot system. Our approach builds upon limited-range Voronoi-based coverage control presented in Section 2.1.1, assuming the map is known *a priori* and shared among the agents. The centroid calculation for each Voronoi cell is modified to incorporate the probability density function defined in (5.1), resulting in:

$$\mathbf{C}_{V_i} = \frac{\int_{V_i} \mathbf{q}\Phi(\mathbf{q}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) d\mathbf{q}}{\int_{V_i} \Phi(\mathbf{q}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) d\mathbf{q}}. \tag{5.2}$$

Finally, the control input is computed to drive each robot toward the centroid of its cell with the proportional law 2.10. Notably, utilizing the probability density function (5.1), calculated by fitting a GMM to the binary mask, creates an attractive field toward the regions of interest, effectively drawing robots over the streets. Simultaneously, the Voronoi partitioning ensures an optimal distribution of agents to maximize the covered

street surface. Moreover, this entire process is executed using only locally available information, eliminating the need for explicit communication among agents or a central computing unit.

5.1.6 Experimental Evaluation

After presenting the methodologies employed in our work, we demonstrate how the entire workflow was validated through simulations. Specifically, we illustrate the process by which a human operator selects a target area from Google Maps, capturing a snapshot of the desired region to feed into the U-Net. Subsequently, we detail the creation of a realistic virtual environment that replicates the selected area, and the calculation of a Gaussian Mixture Model (GMM) by fitting the binary mask output from the neural network. The resulting probability density function is then exploited for the decentralized control of the UAVs. Finally, we present the experimental results, analyzing the evolution of the team's configuration and quantitative performance metrics.

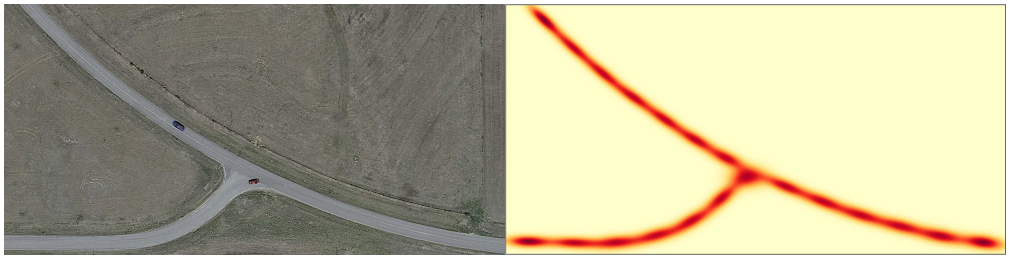
Simulation Setup

Virtual experiments were conducted using aerial robots within the RotorS Gazebo simulation framework [111], controlled via the ROS middleware. To provide visual feedback on the accuracy of our methodology, we created realistic virtual environments that faithfully reproduced the target regions. This enabled visual verification of the control strategy's ability to position UAVs effectively over road networks. The Gazebo worlds were generated by identifying specific areas on Google Maps and extracting 3D features using RenderDoc [154], a graphics debugger capable of capturing rendered frames. The extracted 3D models were imported into Blender, an open-source 3D graphics software, and converted into a format compatible with the Gazebo simulator. This process yielded realistic 3D environments where robot-environment interactions could be evaluated. An example is shown in Fig. 5.1, which reproduces the area surrounding the Colosseum in Rome.

We performed a series of tests in various environments to evaluate the impact of both the number of robots and the sensing range on the team's coverage capabilities. In every simulation, robots were controlled in a strictly decentralized manner, with no direct communication allowed between agents. Communication was restricted solely to the simulation engine to emulate sensing capabilities, providing neighbor position data only when other agents fell within the sensing range.

Road Extraction and GMM Fitting

As previously mentioned, the workflow initiates with the selection of the region to be monitored. After identifying the location and generating the corresponding Gazebo world, we captured a satellite image snapshot from Google Maps to serve as input for the U-Net. Figure 5.4a displays one such example used to test performance across different



(a) **(b)**

Figure 5.4: Test region: (a) Snapshot of the area from Google Maps; (b) heatmap of the calculated GMM, with darker colors corresponding to higher probability values.

scenarios. This specific image depicts a $300 \times 200\,\mathrm{m}^2$ area featuring a simple roadway in a flat environment. The aerial image was resized to 256×256 pixels to match the input requirements of the neural network. The trained U-Net model then processed the resized image to identify streets, generating a binary mask where white pixels represent road surfaces and black pixels represent the background.

Finally, the set of white pixels served as the dataset for fitting the GMM. For this specific environment, we employed $k = 20$ Gaussian components, a number determined by evaluating the Bayesian Information Criterion (BIC) against alternative component counts. This choice balanced the trade-off between fitting algorithm execution speed and the quality of the GMM in modeling the sample set. The resulting GMM is visualized as a heatmap in Fig. 5.4b, clearly showing that the probability density peaks in areas where streets are detected, thereby highlighting these regions for the UAV team.

It is worth noting that all steps described up to this point are performed offline by a central computer. Applying the trained U-Net model to the satellite image and fitting the GMM typically takes a few seconds. However, as this operation is a pre-mission prerequisite, it does not impact the real-time control loop of the robots.

Analysis of Results

Following the GMM fitting, the effectiveness of the proposed solution was tested by deploying the robotic team and evaluating its behavior. To achieve statistical significance and understand how different parameters influence performance, we ran multiple simulations for each environment. Specifically, we tested teams of $n = 12, 16,$ and 20 UAVs, with varying sensing ranges $r \in \{7.5, 15, 30\}\,\mathrm{m}$. For simplicity, the sensing range for neighbor detection was assumed equal to the coverage sensing range. Random starting positions were assigned for each run, and drones flew at a constant altitude $H_z = 1.0\,\mathrm{m}$, sufficient to treat the environment as obstacle-free. The GMM parameters $(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{w})$ were communicated to each robot only at the start of the task; thereafter, communication was denied except for emulated detection.

Initial analysis was conducted by visually inspecting the final configuration of the multi-robot system. An example is shown in Fig. 5.5, where a team of 16 UAVs with sensing range $r = 30\,\mathrm{m}$ successfully achieves the desired configuration, positioning themselves directly above the streets to maximize the monitored surface. Another notable result

Figure 5.5: Final positions of a team of 16 UAVs.

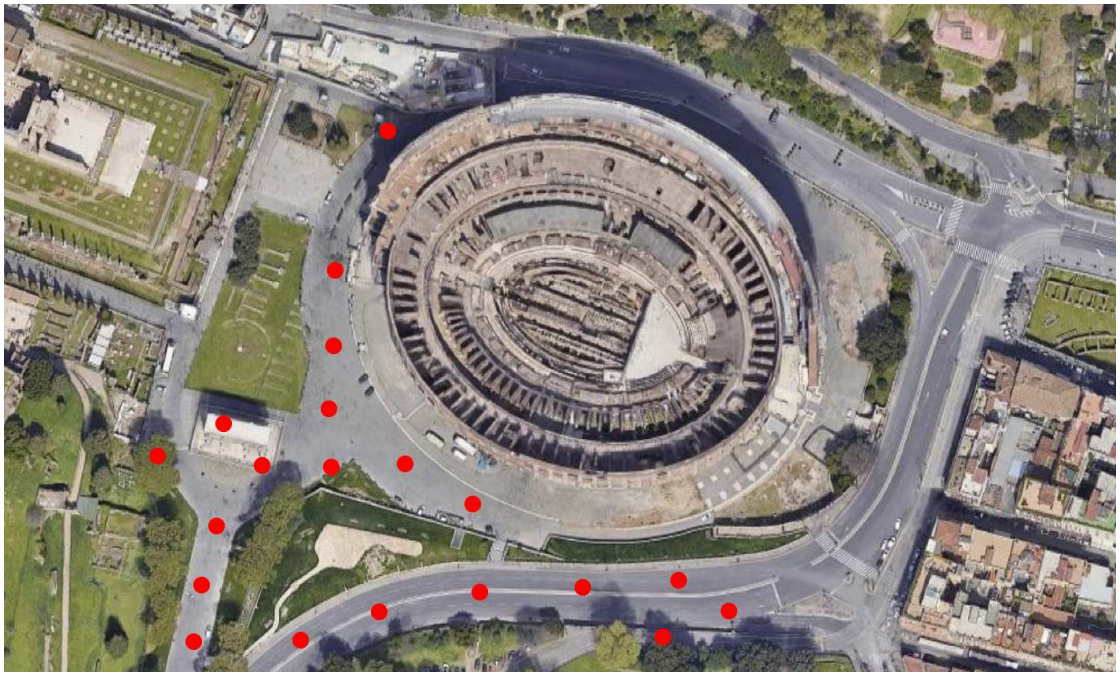
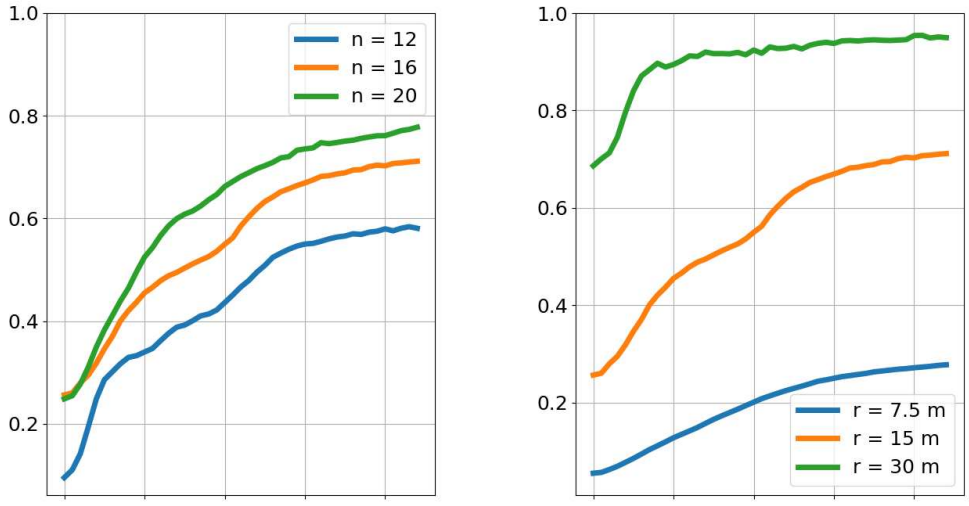


Figure 5.6: The team of UAVs is unable to fully cover such a large area.



(a) **(b)**

Figure 5.7: Coverage effectiveness: (a) with $r = 15\,\mathrm{m}$ and varying n; (b) with $n = 16$ and varying r.

is the total absence of collisions, demonstrating the effectiveness of the limited Voronoi partitioning method in intrinsically integrating collision avoidance into the trajectory planning. Conversely, Fig. 5.6 illustrates a scenario where a team of 20 UAVs was tasked with monitoring a larger area of $440 \times 275\,\mathrm{m}^2$. In this case, the target area exceeds the team's capacity, leaving a portion of the environment uncovered.

Beyond qualitative analysis, we performed a quantitative examination by retrieving log data from each simulation run. We evaluated the coverage effectiveness $\eta \in [0, 1]$, defined as:

$$\eta = \frac{\sum_{i=1}^{n} \int_{V_i} \Phi(\mathbf{q}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) d\mathbf{q}}{\int_{Q} \Phi(\mathbf{q}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) d\mathbf{q}}. \tag{5.3}$$

This metric compares the total probability mass sensed by the team against the total probability mass over the entire environment (which sums to 1). Consequently, coverage effectiveness efficiently quantifies the team's performance in monitoring the regions of interest. The evolution of η over time is depicted in Fig. 5.7 for varying numbers of robots and sensing ranges. The results indicate that η monotonically increases, confirming that robots are attracted toward the regions of interest. Furthermore, the value converges to a maximum specific to each configuration, determined by the team size and sensing capabilities. As expected, complete street surveillance is achievable only when a sufficient number of robots is deployed relative to the environment's size and complexity.

Based on these results, we conclude that the developed strategy successfully addresses the street surveillance problem with a team of quadrotors, driving robots to cover road networks without inter-agent collisions.

5.1.7 Conclusion

In this work we presented a distributed approach to control a team of UAVs to perform a surveillance task. Specifically, the goal of this work was to detect roads from a satellite image and drive the UAVs above them for monitoring. Roads extraction has been possible employing a U-Net, a widely known neural network for image segmentation, which we trained on a dataset of labeled satellite images. Then, suitable probability density of the environment was defined, fitting a GMM to the output binary mask in order to emphasize the higher importance of roadways. Finally, robots were controlled with a distributed Voronoi-based strategy driving the aerial robots towards areas of interest without requiring an explicit communication among them. Our methodology was tested choosing an area from Google Maps and creating a realistic 3D environment to run a series of simulations. The UAV team was able to arrange itself above streets trying to maximize the covered surface, and achieved great performances in terms of coverage effectiveness.
Future works will focus on integrating the detection of other features, indicating areas to be avoided by robots. Another interesting improvement would be integrating the U-Net on-board to the UAVs, allowing them to detect streets by themselves instead of only relying on the probability density calculated offline.

5.2 Decentralized Learning-Based Coverage Control for Multi-Robot Systems with Obstacle Awareness: A CNN-Driven Approach

Building upon the previous chapter, where Convolutional Neural Networks (CNNs) were utilized primarily for extracting geometric features such as street boundaries, this work expands the learning architecture to encode comprehensive local state information directly for decision-making. We present a fully decentralized learning-based solution for multi-robot coverage control, where robots with limited sensing capabilities must monitor events of interest by positioning themselves in regions of high likelihood. In this framework, the CNN processes high-dimensional local observations to generate continuous control inputs, effectively bridging perception and action. To ensure robust operation, the system employs imitation learning from a safety-guaranteed expert, facilitating obstacle-aware behavior without sacrificing coverage performance. Extensive simulations and real-world experiments with mini-quadrotors demonstrate the effectiveness of this proposed approach compared to traditional methods.

5.2.1 Introduction

Researchers have extensively focused their attention on coverage control as presented in Section 2.1 for optimal sensors placement. However, this problem becomes more complicated when dealing with non-ideal scenarios such as limited sensing and communication. Lack of information about team-mates' positions makes it impossible to build a global Voronoi partitioning, thus requiring some approximations. A *Boundary Scan* algorithm is proposed by [155] to sequentially build a limited Voronoi cell from the positions of robots inside the sensing region. This approach is adopted by [37] as shown in Section 2.1.1 and combined with Lloyd's algorithm to provide a solution for limited-range coverage control, yielding to a sub-optimal solution of the problem. [156] make a step further, investigating limited partitioning for heterogeneous robots, whose sensing ranges are different from each other, while [157] restrict the sensing region to a limited angular sector. All these works assume the operational space to be convex, and would fail in the presence of obstacles. Although considering non-convex environments adds complexity to the problem at hand, a solution would be highly beneficial for a real-world deployment of multi-robot systems. Along these lines, a method is proposed by [77] that combines Lloyd's algorithm with a local path planner, resulting in a dual-layer architecture in which the local planner calculates collision-free paths to reach targets defined by the upper level. As in almost any other research areas within the field of robotics, learning-based solutions have also been developed for coverage operations. A Deep Reinforcement Learning (DRL) approach was proposed by [52], integrating Deep Deterministic Policy Gradient (DDPG) with Prioritized Experience Replay (PER) in order to efficiently train a policy with data collected from human demonstration. The learned policy maps measurements from a laser sensor, together with position and velocity of the robot, directly to the action to be executed, making navigation safe even in cluttered environments. Other works propose

to overcome sensing limitations through communication, enabling access to non-local information. An example is provided by [158] with Graph Neural Networks (GNNs), proven to be successful in coordinating a team of robots to complete a coverage control task in non-convex environments, also avoiding getting stuck in local peaks of likelihood density. However, all the mentioned works highly rely on communication to exchange information and effectively coordinate. Instead, another interesting solution exploiting Reinforcement Learning (RL) and local information was developed by [159] and applied to a field coverage task. A robot-centered state space was defined, in order to learn a policy from local observation, represented as a grid around the robot's current position. Vision-based navigation is another solution, as proposed by [160] and our previous work in Section 5.1, but it usually requires large amounts of real-world images to create the dataset for training.
In this work, we address the problem of coverage control in non-convex (or crowded with obstacles) environments, where robots have only access to information in their limited surrounding area. More in details, we develop a Convolutional Neural Network (CNN)-based controller mapping local information to control inputs for the robot. Following a centralized training and decentralized execution framework, similar to the approach in [161], the controller is trained using global data but operates independently based on local inputs during deployment. Finally, we run extensive tests to assess the performances of the proposed methodology, both with simulations and hardware experiments, and compare those results with other solutions from the literature. The dataset of local information features and velocity labels is publicly released (see [162]), as well as the training and testing scripts are available at `https://github.com/ARSControl/cnn_coverage.git`.

5.2.2 Problem Statement

Consider N robots obeying single integrator dynamics:

$$\dot{\mathbf{p}}_i = \mathbf{u}_i, \quad i = 1, \dots, N. \tag{5.4}$$

We use the notation $\mathbf{p}_j^i = \mathbf{p}_j - \mathbf{p}_i$ to indicate the position of robot j relative to robot i. When not specified, the position is expressed relative to a common global reference frame. The team is tasked to cover an environment Q with an importance density field ϕ, which does not change over time but is not known *a priori*. Robots have only access to information available within their sensing region $B(\mathbf{p}, R)$, such as the position of teammates and obstacles, and can not share any other information through communication. We define the set of neighbors for robot i as $\mathcal{N}_i = \{\mathbf{p}_j^i \in \mathbb{R}^2 : \mathbf{p}_j \in B(\mathbf{p}_i, R)\} \subset \mathbb{R}^2$, representing the collection of relative positions of robots within robot i's sensing area. In this research work we focus on developing a robust, fully decentralized control algorithm that leverages only local information to effectively coordinate a multi-robot team in complex, non-convex environments. This approach aims to overcome the limitations of centralized systems and enable scalable, autonomous operation in diverse scenarios.

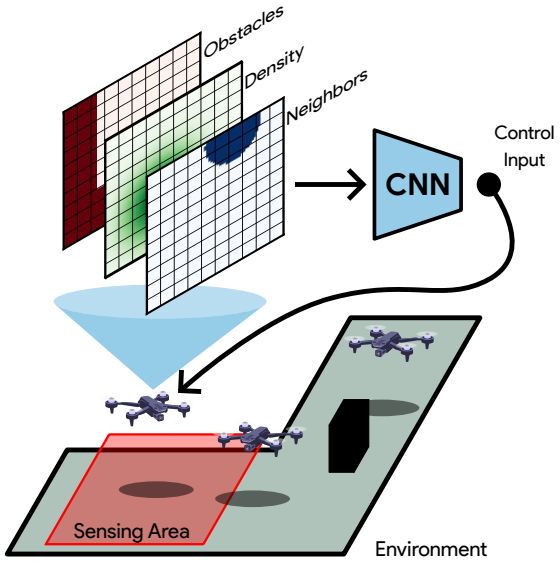


Figure 5.8: Setup of the proposed CNN-based navigation model from local information.

5.2.3 Background on CNNs

CNNs are a class of deep learning models that excel at processing and analyzing image data due to their ability to automatically extract and learn spatial hierarchies of features. The core components of a CNN include:

- *Convolutional Layers*, which apply filters to the input image to detect features such as edges, textures, and patterns.
- *Pooling Layers*, which reduce the dimensionality of the feature maps, retaining essential information while enhancing computational efficiency.
- *Fully Connected Layers*, which aggregate the extracted features to generate the final output or prediction.

For further technical and theoretical insights into CNNs, we direct readers to foundational works such as [163, 164].

5.2.4 Proposed Solution

Here, we present the solution we developed to solve the problem introduced in Section 5.2.2. The proposed architecture is depicted in Fig. 5.8. Each robot captures images of the environment within its limited sensing range, gathering key information: the positions of nearby robots, obstacles, the edges of the environment, and the probability density map that highlights areas requiring the most monitoring. This data is encoded into a 3-channel grid using on-board sensors, with each channel capturing a specific type of information. The information is then fed into a CNN to generate control inputs for the robot through a supervised learning approach. The CNN-based controller drives the robot toward high-priority areas while taking obstacles and nearby robots into account. With this architecture, each robot autonomously perceives its surroundings within a defined range and operates without the need for data exchange with other robots. As a result, each robot can independently cover areas of interest while avoiding collisions with obstacles and teammates.

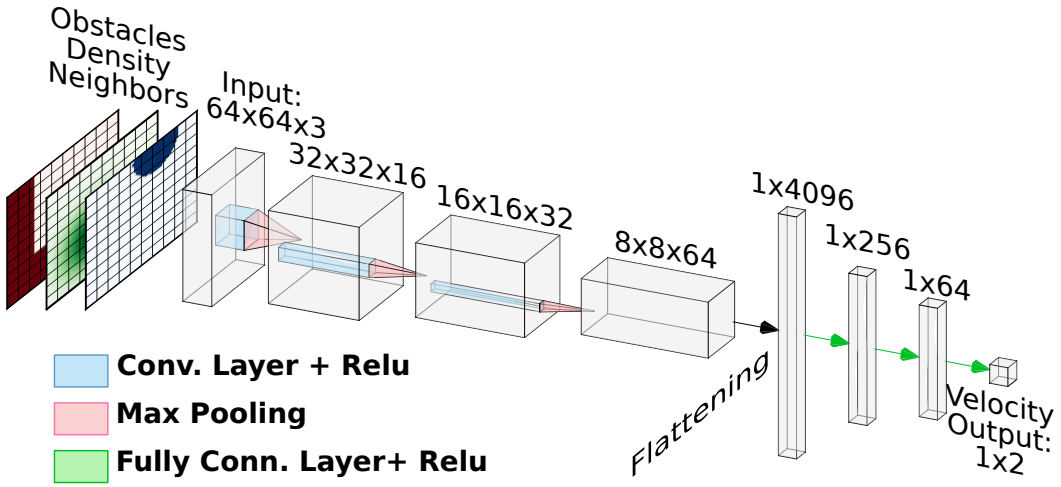


Figure 5.9: Architecture of the CNN

Grid-based Environment Representation

We now describe how information about the surrounding environment is encoded into an element suitable to be fed to the neural network. Since CNNs are mostly suited for image-like inputs, we define $\chi \in \mathbb{R}^{W \times W \times C}$ as our input features, representing local information as a grid-like object, with $C, W \in \mathbf{N}$ indicating the number of channels and cells, respectively. The size W is assumed to be the same for both width and height, thus forming a $W \times W$ grid of elements, corresponding to pixels in an image. This grid is obtained considering the square inscribed in the circumference of the sensing region $B(\mathbf{p}, r)$, whose side can be easily calculated as $r\sqrt{2}$ m. We define

$$\rho = \frac{W}{r\sqrt{2}} \left[\frac{\text{cells}}{\text{m}} \right] \tag{5.5}$$

as the resolution, indicating the relationship between real-world and images dimensions. Then, the image is built from locally available information by each robot encoding useful information, i.e., the local likelihood density, and neighbors and obstacles positions. Taking $C = 3$ channels, every piece of information can be assigned to a dedicated channel, resulting in a final combination as shown in Fig. 5.8. More in details, cells in the fist channel will be assigned with the value of the likelihood density, normalized to 255. Then, the second channel will contain information about the position of detected teammates $\mathbf{p}_j^i$, $\forall j \in \mathcal{N}_i$, encoded as a binary information, with 0 indicating a free cell and 255 an occupied cell. Similarly, the third channel includes boundaries and obstacles' positions, namely $\mathbf{p}_{obs}^i$. Thanks to this strategy, the developed ego-centric representation of information in the nearby area allows single robots to individually take actions, without needing any global knowledge.

Convolutional Neural Network Architecture

We now present the architecture of the neural network, as depicted in Fig. 5.9. The model processes input images of fixed dimensions, 64×64, a commonly used resolution in CNNs as it offers a balance between computational efficiency and the ability to cap-

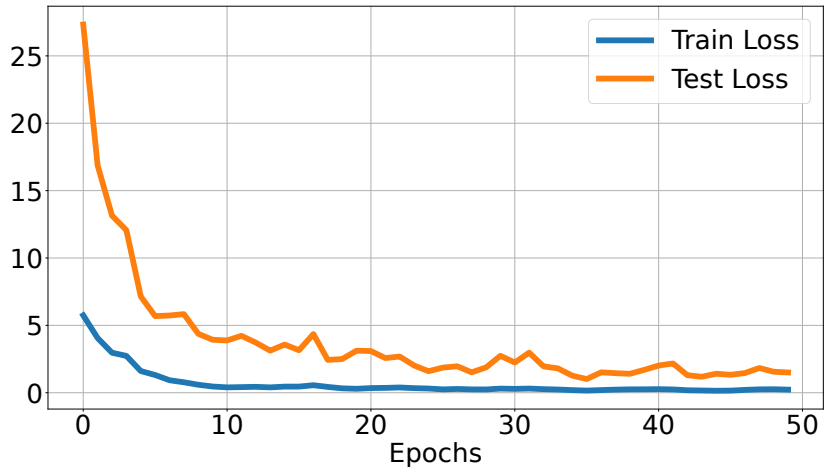


Figure 5.10: Training and Evaluation Loss.

ture sufficient detail in the image. This size allows the network to extract meaningful features while maintaining manageable processing requirements. The architecture follows a hierarchical structure consisting of three convolutional layers, three max-pooling layers, and three fully connected layers. Each convolutional layer is paired with the ReLU activation function to introduce non-linearity, enhancing the model's capacity to learn complex patterns. The convolutional layers progressively increase the number of output channels, starting with 16 in the first layer, 32 in the second, and 64 in the third. All convolutional layers employ 3×3 kernels, which are effective in capturing spatial hierarchies in the input images. Following each convolution, the corresponding feature map undergoes downsampling via max-pooling layers with 2×2 kernels, which reduce the spatial dimensions while retaining essential features. This pooling operation aids in reducing computational complexity and mitigating overfitting by providing a form of translational invariance. Once the feature extraction process is complete, the high-level features are passed through three fully connected layers. The first two layers, consisting of 4096 and 256 units, respectively, are followed by ReLU activation functions to further refine the learned representations. The final fully connected layer has 64 units, which ultimately produce the output of the model: the predicted 2D velocity of the robot. This architecture is specifically designed to balance complexity and efficiency, ensuring that the model captures critical visual features while maintaining robustness in its predictions.

Model Training and Evaluation

After defining the neural network structure and the input data format, we run several simulations of a coverage task as introduced in Section 2.1 performed by an expert algorithm in order to collect data for the training phase. The expert controller was designed in a centralized fashion, having access to the position of every robot in the team and the obstacles in the environment for the entire duration of the task. More in details, the controller followed Lloyd's algorithm (2.10), continuously partitioning the environment with a Voronoi tessellation and calculating the velocity input for each robot. Then, CBFs were exploited for obstacle avoidance, taking velocities calculated at the

previous step as the desired velocities $\mathbf{u}^*$ in (2.13) and returning a safe control input $\mathbf{u}$ for each robot. In order to encode spatial information, we defined $\mathbf{v} = \rho\mathbf{u}$ as the robot's displacement in terms of cells, with ρ calculated as in (5.5) in order to make the model suitable for usage with different values for the sensing range r or the number of cells W. At each time-step, we collected local information available to each robot, in the form of a discretized grid χ as described in Section 5.2.4, together with safe velocities $\mathbf{v_i}$, $\forall i \in \{1, \ldots, N\}$ calculated by the expert controller, creating a dataset of pairs $(\chi_i(t), \mathbf{v}_i(t))$ over several episodes.
The goal of centralized training is designing a learning-based controller minimizing the chances of getting stuck into local peaks of the probability distribution. In addition, integrating CBFs into the control algorithm aims at learning an obstacles-aware policy. However, it is important to note that the resulting learning-based controller does not guarantee safety during operations. For this reason, the evaluation phase described in the following section will also focus on the frequency of collisions happening in a cluttered environment.
Overall, simulations were conducted over 50 episodes, each of which had a duration of 50 time-steps, employing a team of $N = 16$ robots, resulting in a dataset composed by 40000 $(\chi_i(t), \mathbf{v}_i(t))$ pairs, publicly available at [162]. This dataset was split into training and test sets with a ratio of 80% and 20%, respectively.
For the training process, we used the Adam optimizer in order to minimize the mean squared error (MSE) (see [151]) between the expert controller's calculated velocity and the model's predicted velocity. We trained the model for 50 epochs with learning rate $\alpha_L = 0.001$. Results in Fig. 5.10 show both training and test losses decreasing and converging to values close to zero, indicating good performances of the trained model.

5.2.5 Experimental Evaluation

Here, we evaluate the proposed solution with several tests. First of all, we show the performances of the learning-based controller in reaching a complete coverage of the environment. Then, we compare our strategy with methods from the literature to showcase the improved collision-avoidance capabilities. Finally, we present a real-world experiment with a team of quadrotors.

Convergence Analysis

The first set of test we conducted had the goal of assessing the performances of the developed solution, showing its capability in driving robots towards areas of higher interest in the environment, while avoiding collisions with obstacles and other robots. First, we were interested in a qualitative analysis of the system's behavior. For this reason, we ran some simulations with $N = 17$ robots with sensing range $r = 3$ m, and $N_{OBS} = 4$ square obstacles placed in a 20×20 m^2 environment. The starting location of each robot was randomly generated for each run, while the obstacles have always been placed in the same position. The likelihood density ϕ was defined as a Gaussian Mixture Model with 4 components in random locations. Figure 5.11 shows two examples of those tests,

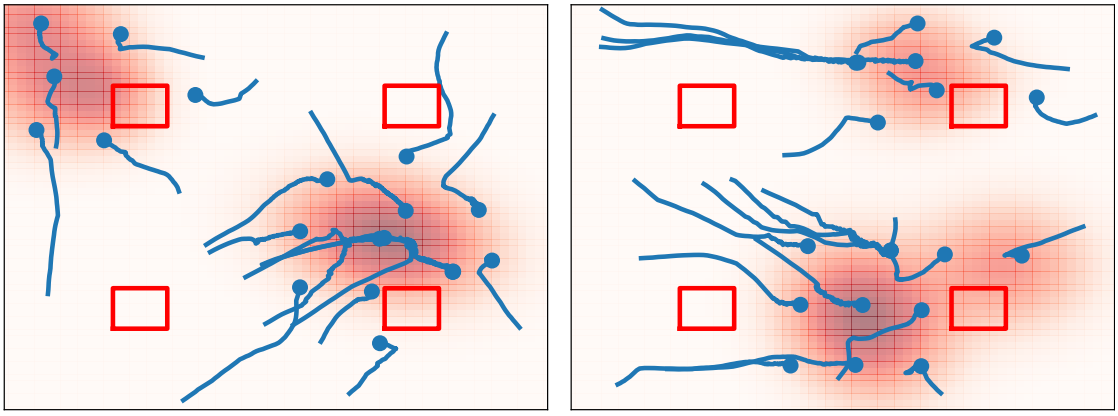


Figure 5.11: Trajectories followed by 17 robots employing the developed learning-based controller.

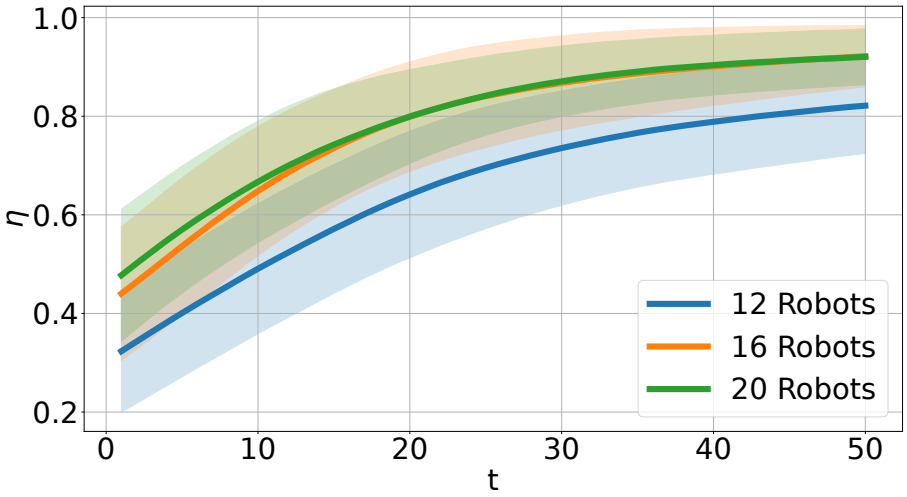


Figure 5.12: The figure presents the coverage effectiveness performance metric for teams of 12, 16, and 20 robots. As the number of robots increases, there is a clear improvement in the performance metric, with coverage approaching nearly 100% of the target area.

highlighting the ability of robots in reaching the regions of interest and avoiding obstacles while navigating.

After a qualitative evaluation, we also conducted a quantitative analysis of our method's performances. First of all, we define as $B(\mathcal{P}, r)$ the area collectively covered by the team, resulting from the union of each robot's sensing region:

$$B(\mathcal{P}, r) = \bigcup_{i=1}^{N} B(\mathbf{p}_i, r). \tag{5.6}$$

Then, we defined a performance metric $\eta \in \mathbb{R}_{\geq 0}$ as the coverage effectiveness, defined as:

$$\eta = \frac{\int_{B(\mathcal{P},r)} \phi(\mathbf{q}) \mathbf{dq}}{\int_{Q} \phi(\mathbf{q}) \mathbf{dq}}. \tag{5.7}$$

The coverage effectiveness η measures how much of the total information ϕ is collected by the team. The simulation setup was mostly the same as the previous tests, with the only difference of the obstacles being randomly placed instead of having fixed positions,

Table 5.1: Statistical evaluation of coverage effectiveness at time $T = 50$.

N	$\overline{\eta}$	σ
12	0.821	0.0969
16	0.921	0.062
20	0.920	0.056

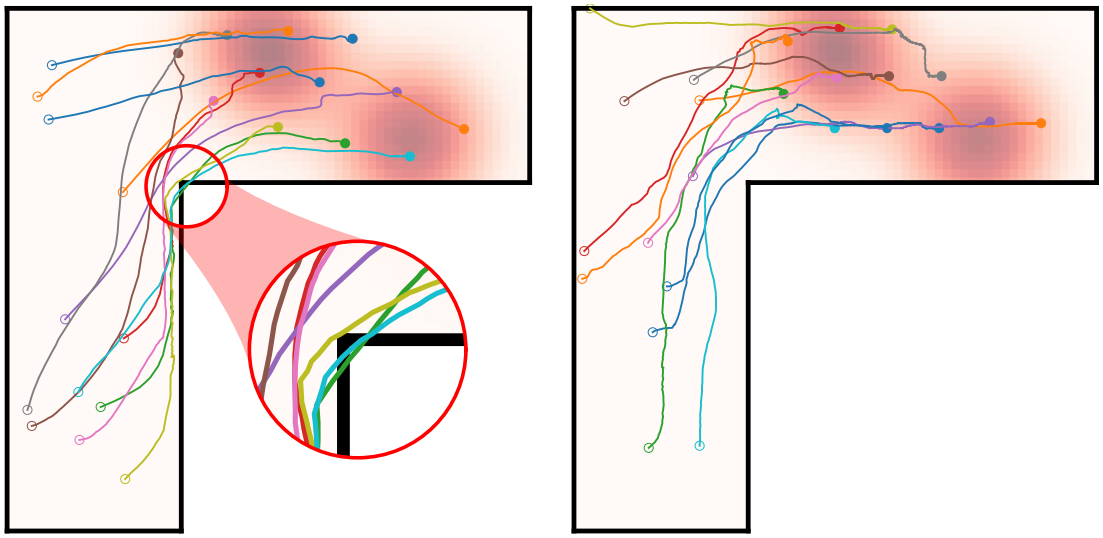


Figure 5.13: Navigation in an L-shaped environment with the solution in [37] (left) is prone to crashes, while it is successful when employing the proposed controller (right).

and a wider environment of 30×30 m^2. Focusing our interest on the differences in performances depending on the number of agents, we used teams composed by $N = \{12, 16, 20\}$ robots, running 50 episodes of the experiment for each configuration. We terminated each episode at a final time $T = 50$ time-steps, even if robots were still moving. The evolution of η over time is shown in Fig. 5.12 for the different configurations, and the final average values $\overline{\eta}$ are reported in Table 5.1 together with the standard deviation σ. As we can see from both Fig. 5.12 and Table 5.1, teams with 16 and 20 robots reach almost complete coverage of areas of interest, showing no differences regardless of the number of robots, while 12 robots seem to be too few to cover a 30×30 m^2 environment.

Comparison with Standard Approach

Subsequently, we focused on the capability of the controller to generate safe velocities in non-convex environments, evaluating the number of collisions happening over different episodes, and making a comparison with another method from literature. Keeping the same setup of previous simulations, with randomly generated robots, obstacles and likelihood density, we ran 50 episodes employing the proposed learning-based controller, and 50 episodes exploiting the solution from [37], where agents perform distributed coverage control with a limited sensing range. Each episode had a fixed duration $T = 50$ time-steps, and the number of episodes failing because of crashes was collected. Again, we

Table 5.2: Number of collisions c and ratio $c_{\%}$ over 50 episodes.

	[37]		Ours	
N	c	$c_{\%}$	c	$c_{\%}$
12	10	0.20	0	0.0
16	13	0.26	1	0.02
20	11	0.22	1	0.02

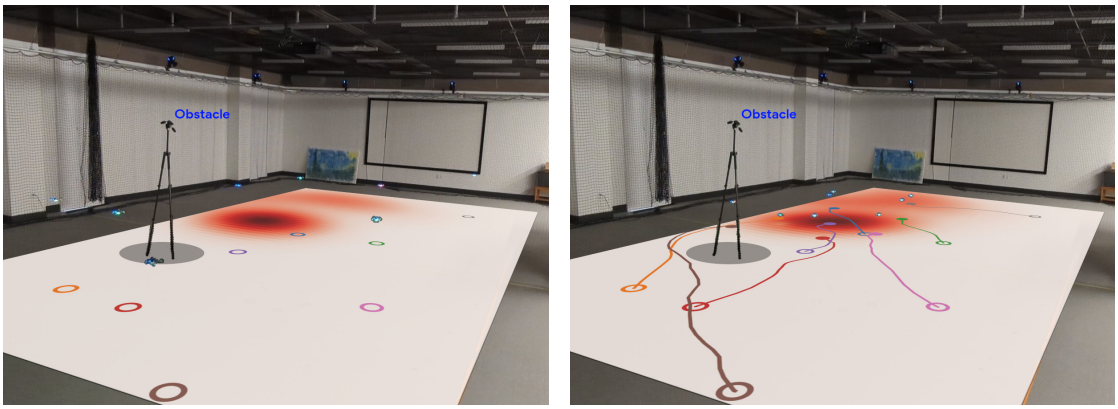


Figure 5.14: The figure shows eight mini-quadrotors autonomously covering a target area while avoiding collisions with each other and obstacles. The left panel depicts their initial positions with ground-truth on the surface. The right panel shows successful, collision-free coverage achieved through a coordination strategy without direct data exchange.

employed $N = \{12, 16, 20\}$ robots, running 50 episodes for each configuration. Collected data is reported in Table 5.2, showing the number of collisions c and the ratio $c_{\%} = c/T$ for both the evaluated methodologies. Furthermore, we also compared the behavior in an L-shaped environment (see Fig. 5.13), showing how non-convexity leads to crashes if exploiting the solution by [37], but can be handled with our proposed controller. As previously mentioned, the proposed solution does not guarantee a safe navigation, but results show drastically decreasing collisions and therefore enhanced safety. While ensuring safety through velocity filtering with CBFs is feasible, it falls outside the scope of this work. The action of the filter would be minimal, since the controller already tries to avoid obstacles, as we show in the following real-world experiments.

Hardware Tests

To rigorously evaluate the efficacy of our proposed solution, we conducted extensive tests using a real robotic platform. The experimental setup comprised a team of eight Crazyflie 2.1 mini-quadrotors (see [165] for more details). Due to hardware limitations, all the computations were done by an external computer, emulating the fully decentralized setup and limited sensing capabilities of robots, considered to have a sensing range $r =$ 2 m. The mini-quadrotors were tasked with covering an area of highest interest in a 11×6 m^2 environment while simultaneously avoiding collisions with each other and a

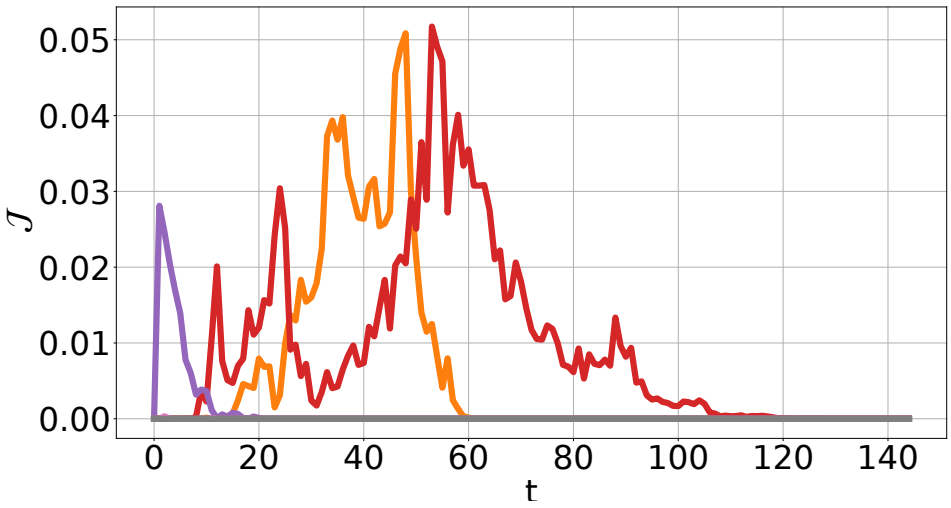


Figure 5.15: Each line shows the value of $\mathcal{J}$ from (5.8) for each robot in a representative run. Some lines remain zero since the model-generated velocity was already safe. Only three lines show minimal non-zero values, indicating the CBF filtering effect is minor and velocities were safety-oriented.

stationary obstacle, as illustrated in Fig. 5.14. The velocity output of the model was filtered with CBFs to ensure no platforms being damaged. Figure 5.14 demonstrates the performance of our proposed strategy. Starting from random initial positions, the quadrotor team successfully converged to cover the area of highest interest. Notably, this convergence was achieved without inter-robot data exchange, relying only on our proposed fully decentralized algorithm. In addition, we evaluated the action of CBFs during the experiment to assess how much the learned controller was already capable of generating safe velocities. For this purpose, we defined a cost function $\mathcal{J}$ recalling the QP in (2.13):

$$\mathcal{J} = \frac{1}{2}\|\mathbf{u} - \mathbf{u}^*\|^2. \tag{5.8}$$

Fig. 5.15 shows values of $\mathcal{J}$, indicating the action of CBFs, with $\mathcal{J} = 0$ indicating no changes to the desired velocity. As we can see, CBFs only act very lightly on 3 robots (the maximum value is $\mathcal{J}_{MAX} = 0.0517$), proving the controller is aware of the obstacles and capable of generating safe velocities. These results underscore the strategy's effectiveness in enabling autonomous, collision-free coverage in complex environments with obstacles.

5.2.6 Conclusion

In this Section, we presented a learning-based controller for a team of robots with limited sensing capabilities to achieve area coverage in a non-convex environment. Our solution makes use of a 3-channels representation of local information, which is processed by a CNN-based controller to generate the control input. Although the controller itself does not formally guarantee collision avoidance, numerous tests have proven its reliability and effectiveness. Future research directions will include extending this work to a 3D scenario, overcoming computational complexity of traditional Voronoi-based solutions.

5.3 Uncertainty-Aware Multi-Robot Flocking via Learned State Estimation and Control Barrier Functions

Information exchange is crucial for optimal coordination of robots, but a link may not always be available among agents to share data. Differently from the previous sections, where learning techniques were used in a perception-oriented way, this work uses learning in a different way and for different purposes, to support coordination by estimating missing information under limited communication. For this reason, this section presents a decentralized solution for flocking control, leveraging state and uncertainty estimation of undetected robots. A neural network is trained to mimic a state estimator, also providing information about the uncertainty of the estimate. This uncertainty is used to weigh the contribution of the estimate in taking actions for coordination. Using Control Barrier Functions and Control Lyapunov Functions, we define an optimization problem to find an optimal control input to reproduce collective motion observed in nature. We evaluate both the learned estimator and the control strategy with extensive simulations.

5.3.1 Introduction

Distributed solutions are promising due to their lack of a central controller, avoiding heavy computation and single points of failure. However, they rely on precise agent coordination, typically through communication. Inaccurate or delayed information limits traditional methods, especially with large teams or in wide environments.

Many research works assume the state of a neighbor can be measured (or communicated) within a sensing (or communication) range [166]. Some works make a step further and take measurements uncertainty into account. The authors in [167] propose event-triggered communication for disturbance identification and convergence guarantees. Bayesian state estimators are usually employed for this goal [168], but they suffer from scalability, easily becoming intractable in a multi-target scenario. A Gaussian Mixture Probability Hypothesis Density (GM-PHD) filter is employed in [104] to overcome this limitation in a formation control setting. However, knowledge about the desired formation is required when detections are missed in order to predict the action the tracked agent will perform.

Recently, interesting solutions for relative localization emerged. Ultra-Wideband (UWB) sensors provide long-range detections [169], while the UVDAR system [170] allows visual detection through Ultra-Violet (UV) sensors and is capable of measuring both position and heading of detected drones. Deep learning has also been used for this purpose [133].

Flocking is a common emerging behavior exhibited in nature by large groups moving cohesively [171]. Studies on collective motion led to the definition of the well-known *Reynolds' rules*: cohesion, alignment, and separation, to replicate the behavior observed in nature [172]. Multi-robot systems appear as a direct field of application for flocking; in fact, its principles have been largely applied to enable decentralized coordination, collective motion, and adaptive behavior in dynamic environments [173, 174]. 2D sce-

narios are usually considered [175], thus neglecting altitude alignment. A solution for 3D cases is proposed in [176] using a potential function for separation, which can not guarantee a minimum distance. Compared to this work, we make use of Control Barrier Functions (CBFs) to provide formal guarantees on collision avoidance, combined with Control Lyapunov Functions (CLFs) to provide alignment in both heading and altitude. A similar optimization-based approach with CBFs has been followed in [177], considering rigid bodies moving in a 3D environment. Instead, Deep Reinforcement Learning is used in [178] to solve the flocking problem, while a solution for both leader-follower and leaderless approaches is proposed in [179]. A common strategy among the mentioned works and among the vast majority in the literature is to keep robots connected to each other to ensure information exchange for optimal coordination.

In this section, following the approach we adopted in our previous work in Section 4.2, we do not force robots to stay close to each other to facilitate information exchange. Instead, we accept to lose contact with some neighbors when they are distant, and evaluate the increasing uncertainty on their states when taking decisions. In this way, our solution is also robust to occlusions or missed detections, thanks to an online pose and uncertainty estimator.

The contributions of this work are two-fold and can be summarized as follows:

- a learning-based state and uncertainty estimation for parallel processing of multiple targets; and
- an uncertainty-aware decentralized flocking strategy exploiting CBFs and CLFs for optimization-based control.

5.3.2 Problem Description

Consider N robots moving in a 3D environment possibly populated with obstacles. We denote the state of each robot by $\boldsymbol{\chi} = [\mathbf{p}; \theta] \in \mathbb{R}^4$, where $\mathbf{p} = [p_x, p_y, p_z]^T \in \mathbb{R}^3$ is the 3D position, and $\theta \in [0, 2\pi)$ is the heading in the $x-y$ plane. The agents move obeying single integrator dynamics $\dot{\boldsymbol{\chi}} = \mathbf{u}$, where $\mathbf{u} = [\mathbf{v}; \omega] \in \mathbb{R}^4$ is the control input, comprising linear velocity $\mathbf{v}$ and angular velocity ω. We assume robots know their starting position and orientation relative to a common global frame, and they can estimate the pose of their local reference frame through odometry information. Instead, information from another robot may be obtained via communication or detection according to a probability function ϕ defined as follows:

$$\phi(d_{ij}) = \begin{cases} -\frac{\phi_{\max}}{R_s} d_{ij} + \phi_{\max} & \text{if } d_{ij} \leq R_s \\ 0 & \text{otherwise} \end{cases}, \tag{5.9}$$

where $d_{ij} = \|\mathbf{p}_j - \mathbf{p}_i\|$ is the euclidean distance between the two robots, $R_s > 0$ is the sensing range, and $\phi_{\max} \in [0, 1]$ is the maximum probability of receiving information. In other words, the probability of receiving information from another robot linearly decreases as the distance increases, becoming zero when robot j is outside of the range R_s of robot i. Note that the use of $\phi_{\max}$ makes information gathering not certain even in the extreme case $d_{ij} = 0$, meaning that we take into account dropped communication,

as well as missed detections or occlusions. We further define the received information as $\mathbf{r}_j = \boldsymbol{\chi}_j + \boldsymbol{\rho}$, where $\boldsymbol{\rho} \sim \mathcal{N}(0, \Sigma_M)$ is a zero-mean Gaussian noise defined by the covariance matrix $\Sigma_M \in \mathbb{R}^{4\times4}$. The addition of a noise term makes the gathered information not fully reliable, and models non-idealities usually present in the real world such as communication delays, sensors inaccuracy, and odometry drifting. Finally, we assume robots are capable of detecting and accurately localizing N_{obs} obstacles, which we model as vertical poles and can be seen as circular obstacles in the $x - y$ plane.

Problem Formulation

The goal of this work is the definition of a decentralized controller reflecting Reynolds' flocking rules for cohesion, alignment, and separation, while dealing with uncertainties on the state of other robots. For alignment, we mean robots are required to synchronize both their altitude p_z and their orientation θ.

5.3.3 Learned Estimation

As already mentioned, relative state estimation is of particular importance for a team of robots to achieve optimal coordination, but limited in accuracy and scalability when dealing with large robotics teams. Our solution to overcome those limitations is the development of a learning-based state and uncertainty estimator, trained to reproduce the action of a particle filter.

Data Collection

We collected data for training running a traditional particle filtering state estimator tracking the state $\boldsymbol{\chi}$ of a point moving in the environment. Following the traditional approach, we want to combine the current belief $\hat{\boldsymbol{\chi}}(t-1) \in \mathbb{R}^4$ and uncertainty on $\boldsymbol{\chi}$, namely $\Sigma_\chi(t-1) \in \mathbb{R}^{4\times4}$, with possible measurements $\mathbf{r}(t)$, taking the measurement uncertainty matrix Σ_M into account. From this data, we want to calculate the updated belief $\hat{\boldsymbol{\chi}}(t)$ and covariance matrix $\Sigma_\chi(t)$.
For this purpose, we ran $N_{ep} = 1000$ episodes employing the expert particle filter estimator. We set the number of particles to $N_p = 1000$, and the number of time-steps for each episode to $N_s = 1000$. The measurement covariance was calculated as $\Sigma_M = \sigma_M I$, where $I \in \mathbb{R}^{4\times4}$ is the identity matrix and σ_M is randomly generated for each episode. The likelihood of detecting the target was set to 25% for each time-step. We collected all the mentioned data for each time-step in the form of a feature vector $\mathbf{x}$ and a targets vector $\mathbf{y}$. Additionally, Cholesky decomposition has been applied to the matrices to ensure the neural network will produce a consistent covariance matrix, which must be positive definite. Hence, lower-triangular matrices $\Sigma_\chi^L, \Sigma_M^L \in \mathbb{R}^{4\times4}$ were calculated, such that $\Sigma_\chi = \Sigma_\chi^L {\Sigma_\chi^L}^T$ and $\Sigma_M = \Sigma_M^L {\Sigma_M^L}^T$. Finally, features data is stacked to generate the feature vector $\mathbf{x} = [\hat{\boldsymbol{\chi}}(t-1); \mathbf{L}_\chi(t-1); \mathbf{r}(t); \mathbf{L}_M] \in \mathbb{R}^{28}$, where $\mathbf{L} \in \mathbb{R}^{10}$ is the vector of the non-zero elements of Σ^L. Similarly, the targets vector becomes $\mathbf{y} = [\boldsymbol{\chi}(t); \mathbf{L}_\chi(t)] \in \mathbb{R}^{14}$.

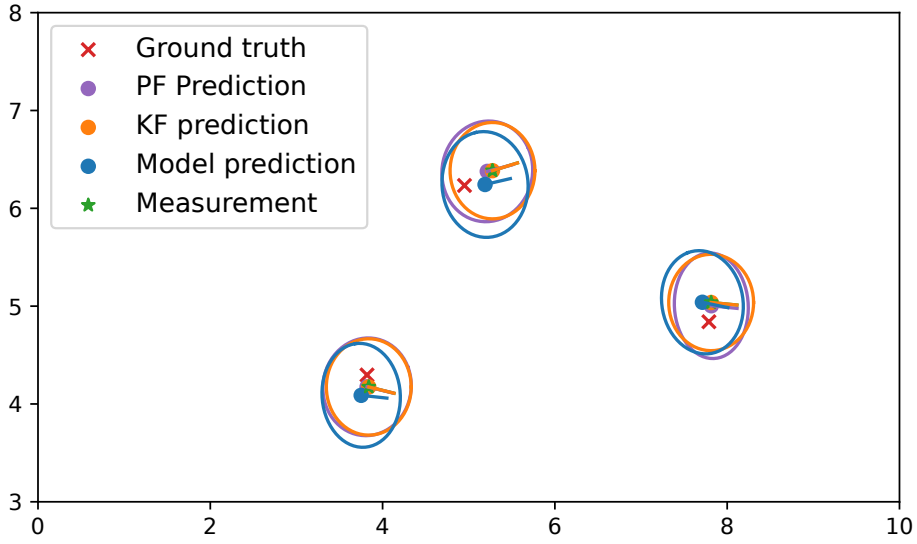


Figure 5.16: Comparison among predictions produced by a Kalman filter (KF, in orange), a particle filter (PF, in purple), and our model (in blue). Only $x - y$ coordinates and orientation are shown for clarity.

The full dataset can be organized into batched features $\mathbf{X} \in \mathbb{R}^{(N_{ep} \cdot N_s) \times 28}$ and batched targets $\mathbf{Y} \in \mathbb{R}^{(N_{ep} \cdot N_s) \times 14}$.

Model Training

After data collection, we moved to the training phase to develop a neural network capable of estimating the state and uncertainty of a tracked target. The chosen architecture was a simple feed-forward network with one hidden layer composed of 512 neurons. Instead, the input and output layers were made consistent with the collected data, assigning them 28 and 14 neurons, respectively. We used the Adam optimizer to minimize the Mean Squared Error (MSE) between the output by the expert estimator and the model's prediction, and we split the dataset into training and testing sets, running the training loop for 100 epochs. The train loss reached a final value $l_T = 2.357 \cdot 10^{-3}$, similar to the test loss $l_t = 2.170 \cdot 10^{-3}$.

Model Evaluation

The evaluation of the trained model was carried out investigating the performance in accuracy and time required for the computation, comparing the results with traditional estimators such as Kalman filters and particle filters. For this purpose, we simulated $N = \{10, 20, 30, 50, 100\}$ robots moving in the environment, whose detection probability was fixed to $\phi = 50\%$. At each time-step, the trained network processed the batched data related to the N robots, while N instances of Kalman filters and particle filters were needed to track the targets. The estimated positions and 95% confidence ellipses calculated from the covariance matrices are depicted in Fig. 5.16 for the 3 methods,

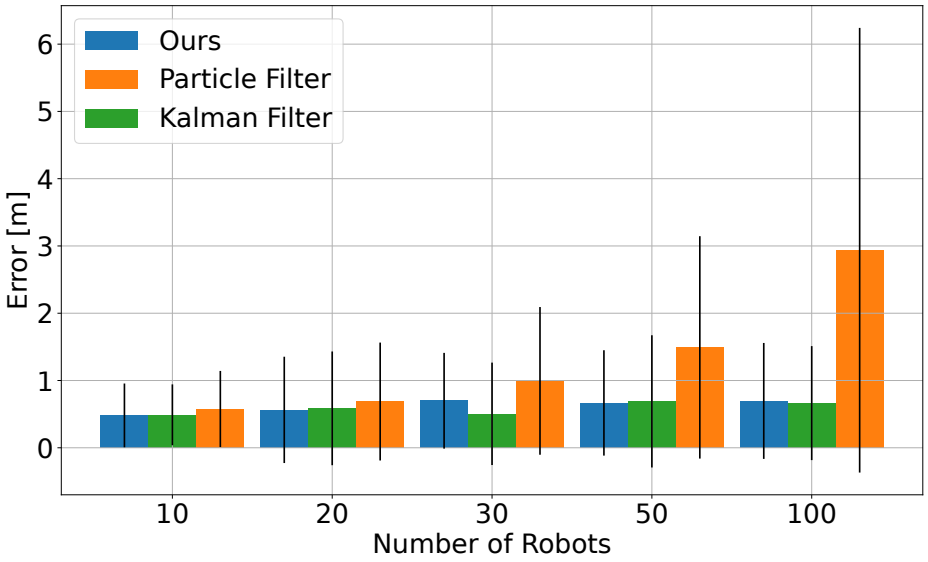


Figure 5.17: Average error of the proposed method (in blue) compared with Kalman filter (in green) and particle filter (in orange). The accuracy of our solution makes it comparable to traditional state estimators.

Table 5.3: Time comparison with different state estimators.

N	Time [ms]		
	Ours	PF	KF
10	0.109	14	**0.107**
20	**0.097**	32	0.216
30	**0.11**	42	0.28
50	**0.115**	66	0.437
100	**0.088**	129	0.842

showing how the neural network gives almost identical results to traditional solutions. In addition, Fig. 5.17 provides numerical details on the accuracy of the analyzed methods, comparing the average tracking errors during the simulation. In our implementation, we used a fixed total number of particles for the particle filter, which were evenly distributed among all robots. Consequently, as the number of robots increases, each robot receives fewer particles, leading to a degradation in estimation accuracy. Although our model has good accuracy, it is important to note that the covariance matrix indicating the uncertainty is what we are mainly interested in (i.e., the elliptical regions in Fig. 5.16), rather than the estimated pose itself. In addition to accuracy, we evaluated the time required to process all the targets, which is the main reason we developed a neural network. We performed inference using our model on an NVIDIA GeForce RTX 4060 Laptop GPU, while the traditional estimators were executed on a 13th Gen Intel Core i7-13700H CPU. As shown in Table 5.3, our model outperforms the other estimators and, most importantly, it shows great scalability properties thanks to parallel computation.

5.3.4 Flocking Control Design

We design a set of CBFs and CLFs to embody Reynolds' flocking rules for cohesion, separation, and alignment among team members. The use of CBFs and CLFs not only provides formal safety guarantees within the team, preventing robots to collide with each other, but also allows the integration of other safety requirements such as avoiding external obstacles. In addition, they allow prioritization of control objectives, e.g., making the team break the formation in order to avoid an obstacle. First, we define the set of indices indicating the team-mates of robot i as $\mathcal{N}_i$, and we formulate each CBF and CLF $\forall j \in \mathcal{N}_i$. We refer to Section 2.2 for all the necessary definitions regarding CBFs and CLFs.

CBFs and CLFs for Reynolds' Flocking Rules

1. **Separation CBFs**: The first and most important behavior that robots should have is separation to prevent collisions. According to Reynolds' rules, a repulsive force steers a robot away from a flock-mate with increasing magnitude as the distance between them decreases. We formulate a CBF $b_{\text{sep}} : \mathbb{R}^3 \to \mathbb{R}$ replicating this behavior as:

$$b_{\text{sep}_j}({}^i\mathbf{p}_j) = \|{}^i\mathbf{p}_j\|^2 - D_s^2, \quad \forall j \in \mathcal{N}_i \tag{5.10}$$

where ${}^i\mathbf{p}_j$ is the position of robot j relative to robot i's local frame, and D_s is a safety distance. Taking $\alpha_{\text{sep}} = \gamma_{\text{sep}} b_{\text{sep}}^3$, with $\gamma_{\text{sep}} \in \mathbb{R}_{>0}$, the condition (2.12) can be approximated to:

$$2\,{}^i\mathbf{p}_j^T \mathbf{v} + \gamma_{\text{sep}} b_{\text{sep}}^3 \geq 0 \tag{5.11}$$

Similarly, given the 2D relative position ${}^i\mathbf{p}_{\text{obs}} \in \mathbb{R}^2$ of an obstacle, we can integrate obstacle avoidance as:

$$b_{\text{obs}}({}^i\mathbf{p}_{\text{obs}}) = \|{}^i\mathbf{p}_{\text{obs}}\|^2 - D_s^2. \tag{5.12}$$

As we already mentioned, we model obstacles as poles in the environment, so we only consider their 2D coordinates, meaning that the team will be forced to move on the $x - y$ plane to avoid an obstacle. Again, the condition (2.12) becomes:

$$2\,{}^i\mathbf{p}_{\text{obs}}^T [v_x; v_y] + \gamma_{\text{obs}} b_{\text{obs}}^3 \geq 0. \tag{5.13}$$

2. **Cohesion CBFs**: Cohesion has the goal of keeping robots within the flock close to each other. It is usually achieved by designing an attractive force driving a robot towards the local centroid of the formation, namely $\boldsymbol{\chi}_{C_i} = [p_{C_x}, p_{C_y}, p_{C_z}, \theta_C]^T \in \mathbb{R}^4$:

$$\boldsymbol{\chi}_{C_i} = \frac{1}{N-1} \sum_{j \neq i} \boldsymbol{\chi}_j. \tag{5.14}$$

Since our goal is to reach a planar configuration for the swarm, we only want the 2D position to be constrained by cohesion, while altitude and orientation will be treated in the following. As a matter of fact, a planar configuration is usually optimal for many tasks like monitoring and search and rescue, and also prevents undesirable

effects such as downwash when robots fly over one another. For this reason, we define $\overline{\boldsymbol{\chi}}_{C_i} = [p_{C_x}; p_{C_y}] \in \mathbb{R}^2$ as the vector of 2D coordinates of the centroid, and we design the cohesion CBFs to keep robots within a distance $R_f \in \mathbb{R}_{>0}$ from $\overline{\boldsymbol{\chi}}_{C_i}$:

$$b_{\text{coh}}({}^i\overline{\boldsymbol{\chi}}_{C_i}) = -\|{}^i\overline{\boldsymbol{\chi}}_{C_i}\|^2 + R_f^2. \tag{5.15}$$

Cohesion must be balanced with separation, which must always be guaranteed for safety reasons. Furthermore, we want to sacrifice cohesion to overcome deadlocks, when robots may need to split up to avoid an obstacle or line up to pass through a narrow corridor. Therefore, we write (2.12) as:

$$-2{}^i\overline{\boldsymbol{\chi}}_{C_i}^T[v_x; v_y] + \gamma_{\text{coh}} b_{\text{coh}}^3 + \delta_{\text{coh}} \geq 0 \tag{5.16}$$

where $\delta_{\text{coh}} \in \mathbb{R}_{\geq 0}$ is a slack variable allowing the system to go outside the safe set defined by (5.15). By minimizing δ_{coh}, robots will try to reach cohesion as much as possible while satisfying safety requirements defined by separation CBFs. Finally, it is important to note that the choice of $\alpha_{\text{coh}} = \gamma_{\text{coh}} b_{\text{coh}}^3$ makes it an extended class $\mathcal{K}$ function according to Def. 2.2.1. As we have already shown in Section 4.2, this makes b_{coh} also a CLF for the system (see Theorem 1). As a result, if the initial distance is greater than R_f, a robot will be moved closer to $\overline{\boldsymbol{\chi}}_{C_i}$ until reaching the safe set defined by (5.15).

3. **Alignment CLFs**: Finally, robots must reach an agreement on altitude and orientation in order to reach a planar configuration and face the same direction. We make use of CLFs to obtain optimization-based control inputs for vertical and angular velocity. A CLF $V_z : \mathbb{R} \to \mathbb{R}$ can be defined for altitude consensus, taking the distance to the z-coordinate of the centroid p_{C_z} to zero:

$$V_z(p_{C_z}) = (p_{C_z} - p_z)^2. \tag{5.17}$$

Condition (2.14), with the addition of a slack variable δ_z and taking $\zeta_z = \gamma_z V_z$, then becomes:

$$-2(p_{C_z} - p_z)v_z + \gamma_z V_z + \delta_z \leq 0. \tag{5.18}$$

In the same way, we can define a CLF for consensus on orientation as:

$$V_\theta(\theta_C) = (\theta_C - \theta)^2 \tag{5.19}$$

and the derived CLF condition (2.14) as:

$$-2(\theta_C - \theta)\omega + \gamma_\theta V_\theta + \delta_\theta \leq 0. \tag{5.20}$$

Uncertainty-Aware Controller Design

The CBFs and CLFs we designed so far make use of the ground truth position of neighbors, $\mathbf{p}_j$. As we already mentioned, in real applications with large teams, this information can be inaccurate and usually not always available because of occlusions, communication

issues, or great distances. As described in Sec. 5.3.3, we aim to always keep an estimate on all other robots within the team, refining the belief when measurements are available. Therefore, the controller can only employ the available information gathered from the learning-based estimator, consisting of the estimated pose $\hat{\boldsymbol{\chi}}_j(t)$ and the covariance matrix $\Sigma_{\chi_j}(t)$, $\forall j \in \mathcal{N}_i$. For this purpose, we replace the ground truth data $\boldsymbol{\chi}_j$ with the estimate $\hat{\boldsymbol{\chi}}_j$ in all the CBFs and CLFs constraints described above. In addition, we make use of the uncertainty on the current estimate by defining a weight $\lambda_j \in \mathbb{R}_{>0}$ as:

$$\lambda_j = \frac{1}{tr(\Sigma_j)}. \tag{5.21}$$

Note that $\lambda_j > 0$ because Σ_j is positive definite, since it is constructed by the Cholesky elements processed by the neural network. This weight indicates how reliable the estimate is, assuming small values for unrealiable information. Thanks to this weighting factor, we can formulate a new weighted centroid definition:

$$\hat{\boldsymbol{\chi}}_{C_i} = \frac{\sum_{j \neq i} \lambda_j \hat{\boldsymbol{\chi}}_j}{\sum_{j \neq i} \lambda_j}. \tag{5.22}$$

Given this new definition of the centroid, the controller will be able to consider all other agents for cohesion and alignment, assigning greater importance to more reliable data.

Following the usual approach in CBF-CLF-based control [39], the optimal control input $\mathbf{u}$ can be calculated as the result of a Quadratic Programming (QP) optimization problem minimally modifying a desired input $\mathbf{u}^* = [\mathbf{v}^*; \omega^*]$ while satisfying CBFs and CLFs conditions. First, we define the linear component $\mathbf{v}^*$ of the desired input to follow a possibly moving rendez-vous point $\mathbf{p}^*$:

$$\mathbf{v}_i^* = k_p(\mathbf{p}^* - \mathbf{p}_i). \tag{5.23}$$

Then, the desired orientation ω^* is chosen to face the motion direction, thus resulting as:

$$\omega_i^* = \text{atan2}(v_{y_i}, v_{x_i}) - \theta_i. \tag{5.24}$$

Combining all the information, we can write all the CBFs and CLFs inequalities explicitly showing the control input $\mathbf{u}$ using the form $A\mathbf{u} \leq b$, which is common for most solvers. As an example, (5.11) becomes $A_{\text{sep}}\mathbf{u} \leq b_{\text{sep}}$, taking $A_{\text{sep}} = -2[{}^i\mathbf{p}_j; 0]^T$ and $b_{\text{sep}} = \gamma_{\text{sep}} b_{\text{sep}}^3$. In the same way, we can get A_{obs}, b_{obs}, A_{coh}, b_{coh}, A_z, b_z, and A_θ, b_θ. In addition, we can define a vector of slack variables $\boldsymbol{\delta}$ as the collection of slack variables appearing in (5.16), (5.18), and (5.20). Each slack variable will contribute to the total cost of the optimization problem weighted by a scalar $\mu \in \mathbb{R}_{>0}$, and we denote as $\boldsymbol{\mu}$ the vector

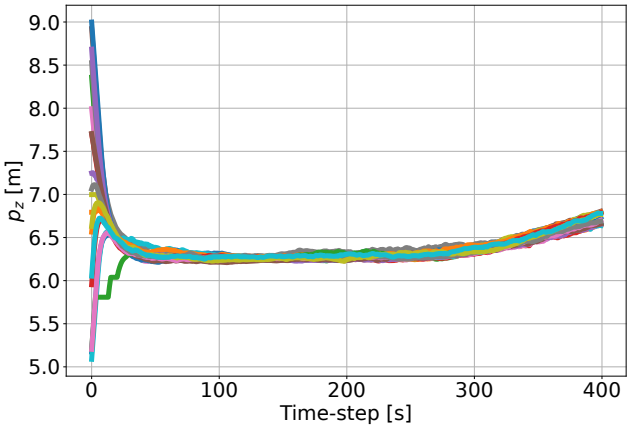


Figure 5.18: Altitude of the robots during the execution of the task: starting from different vertical coordinates, they rapidly align their altitude creating a planar configuration.

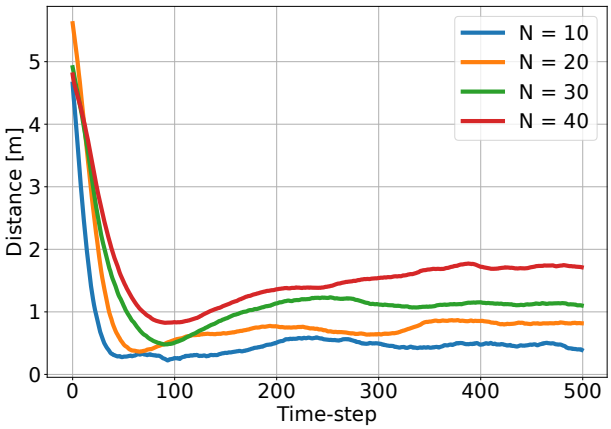


Figure 5.19: Distance between the centroid of the team and the moving target with varying numbers of robots.

containing all the weighting factors. The QP problem can be formulated as follows:

$$\begin{aligned} \mathbf{u} = \underset{\mathbf{u}\in U}{\operatorname{argmin}} & \frac{1}{2}\|\mathbf{u}-\mathbf{u}^*\|^2 + \boldsymbol{\mu}^T\boldsymbol{\delta} \\ \text{s.t. } & A_{\text{sep}}\mathbf{u} \leq b_{\text{sep}} \\ & A_{\text{obs}}\mathbf{u} \leq b_{\text{obs}} \\ & A_{\text{coh}}\mathbf{u} \leq b_{\text{coh}} \\ & A_z\mathbf{u} \leq b_z \\ & A_\theta\mathbf{u} \leq b_\theta \\ & \mathbf{u}_{\min} \leq \mathbf{u} \leq \mathbf{u}_{\max} \end{aligned} \tag{5.25}$$

where the actuation limits $\mathbf{u}_{\min}, \mathbf{u}_{\max} \in \mathbb{R}^4$ have been included to generate admissible control inputs.

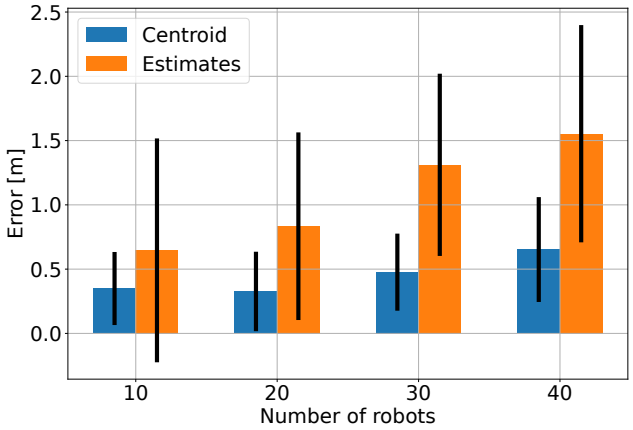


Figure 5.20: Average error between the estimated positions of neighbors and the calculated centroid, relative to the ground truth values. Taking uncertainty into account improves the accuracy of centroid calculation.

5.3.5 Experimental Evaluation

The developed solution was evaluated running extensive simulations. We modeled robots as points moving in a 3D cubic environment with a side length $W = 20$ m where obstacles were also placed. Robots were required to flock in order to cohesively track a moving rendez-vous point, keeping the centroid of the configuration as close as possible to the target. Information gathering from other robots was modeled as described in Sec. 5.3.2, according to the detection probability $\phi(d_{ij})$ calculated from (5.9) with $\phi_{\max} = 0.9$. The communication range was taken as $R_s = 5.0$ m, while the measurement covariance defining the measurement error was calculated as $\Sigma_M = 0.05 \cdot I$, with $I \in \mathbb{R}^4$ being the identity matrix. We defined a safety distance $D_s = 2.0$ m to be kept from obstacles and other robots. In the CLF-CBF optimization problem, we chose $\gamma_{\text{sep}} = \gamma_{\text{obs}} = \gamma_{\text{coh}} = 0.1$, and $\gamma_z = \gamma_\theta = 0.5$. Teams with increasing numbers of robots were tested, running simulations with $N = \{10, 20, 30, 40\}$ agents within the team. In Fig. 5.21 we can see the execution of the task by a team of $N = 30$ robots. Starting in a random state, they start moving towards the target, which is initially static. Once they have reached heading alignment, they flock and follow the target, but they are also able to separate to maintain a safety distance from the obstacle. Finally, cohesion actions take over again, moving the robots to the central area. Fig. 5.18 instead shows how robots start from different heights, but align at the same altitude, creating a planar configuration. Some representative simulation runs are also shown in the accompanying video.

From a quantitative point of view, we evaluated how the team was able to stay close to the rendez-vous point and how the size of the team influences the performance. Fig. 5.19 shows the evolution of the distance between the centroid of the configuration and the target. As it is easy to see, robots rapidly create a compact group close to the target, and keep the monitored distance bounded and stable. Finally, we collected data about the error on estimated neighbors' positions from the neural network predictor and their ground truth locations, and we made a comparison on the error on the estimated centroid $\hat{\boldsymbol{\chi}}$. In Fig. 5.20, it is shown how (5.14) is effective in calculating $\hat{\boldsymbol{\chi}}$ taking uncertainty into account, improving accuracy when estimates are highly unreliable.

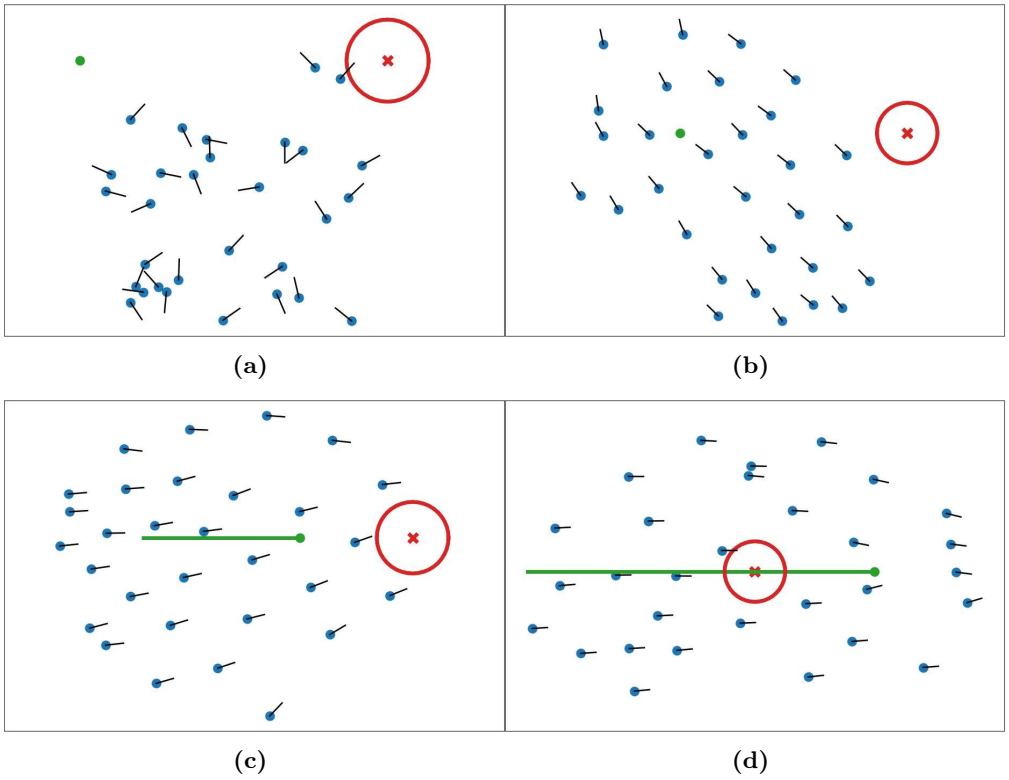


Figure 5.21: Top view snapshots showing a team of 30 robots following a moving target (in green). (a) Robots are randomly placed in the environment; (b) the target is initially static, and robots align their heading; (c) the target starts moving and the team moves accordingly; (d) robots spread to maintain a safety distance from the obstacle (in red), then reunite in the center.

5.3.6 Conclusion

In this Section, we presented a decentralized solution for flocking control. Unlike the common approach, our solution does not force robots to stay close to each other to facilitate information exchange. Instead, we developed a learning-based state and uncertainty estimation using possibly available detections to evaluate the state of neighbors. The produced uncertainty is then exploited to define a CBF-CLF-based optimization problem, providing optimal control inputs for a robot to reproduce Reynolds' flocking rules for collective motion. Future works will include improvement of the learned estimator, possibly including a prediction module exploiting data from an expert controller to forecast how other robots are likely to move based on the task they are accomplishing. We also plan to run real-world experiments and comparisons with other decentralized methods.

5.4 Distributed Multi-Robot Ergodic Coverage Control for Estimating Time-Varying Spatial Processes

While the previous section relied on neural networks for estimating agent states, this work employs a different machine learning technique, Gaussian Processes, to model environmental dynamics and uncertainty. We present a fully distributed framework for multi-robot exploration and coverage of time-varying spatial processes in complex, non-convex environments. Building on heat-equation-driven adaptive coverage (HEDAC) and system ergodicity, the proposed approach enables robots to autonomously navigate arbitrary domains, reconstruct unknown spatial fields, and continuously balance exploration and coverage without centralized coordination. A temporal decay mechanism promotes adaptive monitoring by regulating the relevance of past observations. Simulation and real-world experiments demonstrate the effectiveness and robustness of the method.

5.4.1 Introduction

Multi-robot systems are increasingly used to explore and monitor environments where human operation is impractical. Central to this is coverage control, which allows robot teams to focus sensing resources on spatial regions of interest, typically described by a density distribution.

Traditional approaches, such as Voronoi-based coverage control [37, 180], position robots at the centroids of their Voronoi cells to optimize static coverage. These approaches, however, generally assume convex environments and strong connectivity, constraining their performance in cluttered or resource-limited scenarios. More recent solutions employing model predictive control (MPC) or graph neural networks (GNN) architectures have attempted to overcome these limitations, but they often incur high computational costs or require extensive ad hoc training [181, 182, 183].

Ergodic coverage control offers a dynamic alternative, aligning a robot's time-averaged trajectory with a spatial distribution of interest. The Spectral Multiscale Coverage (SMC) framework [184] introduced a metric-based formulation, later extended with time-optimal [185, 186] and receding-horizon methods [187]. Still, SMC assumes convexity, lacks inter-robot collision avoidance, and tends to emphasize global coverage at the expense of distant high-priority regions [188, 189].

The Heat Equation Driven Area Coverage (HEDAC) framework [190] addresses some of these issues by embedding the ergodic metric in a stationary heat equation. This formulation naturally handles non-convex domains and encourages local coverage through heat diffusion. Obstacle modeling was further generalized via the Finite Element Method in [191]. However, HEDAC does not incorporate inter-robot safety and fails to guarantee a minimum distance between robots.

A further limitation of prior work is the assumption of full knowledge of a static spatial process. Real-world applications often involve unknown, time-varying phenomena requiring online exploration, learning, and monitoring. While Gaussian Processes (GPs) are

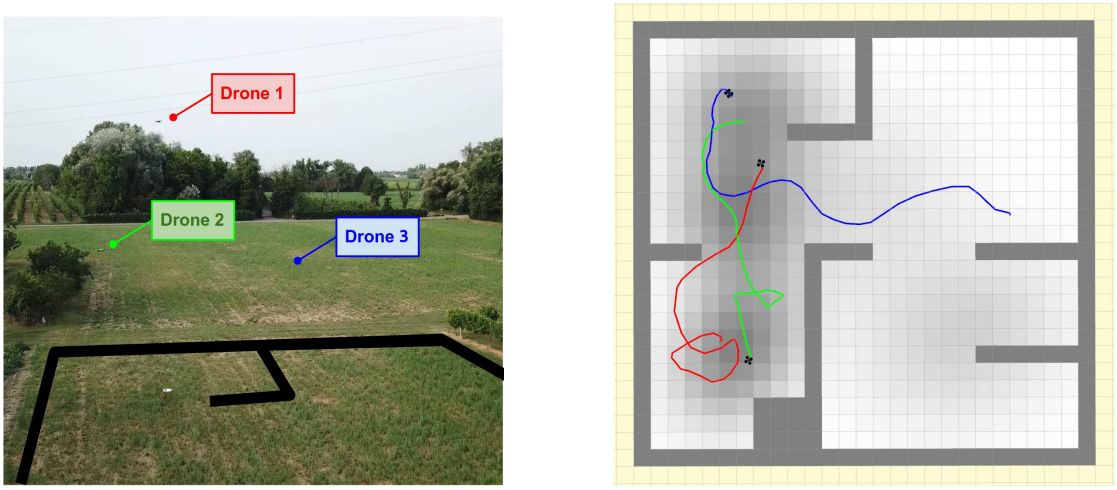


Figure 5.22: Snapshot from a real-world experiment demonstrating the proposed distributed strategy using a team of three Uvify IFO-S drones. Each drone runs the algorithm fully onboard and collaborates with neighboring agents via minimal communication. The image shows the experiment in progress, accompanied by an RViz visualization depicting the environment, robot trajectories, and spatial process estimation.

widely used for spatial modeling [192, 193, 194], their computational cost and sensitivity to redundant data limit scalability [195]. Recent sampling strategies have tackled this by retaining only informative data points [195], or decaying older samples to promote adaptiveness [196]. Yet these methods face trade-offs between memory efficiency and process tracking fidelity.

Despite recent advances, based on an extensive review of the literature, no existing method simultaneously integrates distributed ergodic control, responsiveness to time-varying spatial processes, and collision-aware motion in complex, non-convex environments. The only notable exception is the decentralized ergodic control framework of [197]. However, their approach relies on a pre-trained GP to model the spatial process, without explicitly leveraging GP uncertainty, and is tailored to target localization rather than coverage control. As such, it does not address coverage and estimation of time-varying processes, and therefore cannot serve as a meaningful baseline for our problem. Other works have investigated multi-agent ergodic trajectory optimization in the context of exploration with time-varying sensor visibility [198], but these approaches likewise do not provide a solution to the persistent coverage problem. To overcome these limitations, we propose a fully distributed framework for exploration and monitoring of unknown, time-varying spatial processes in complex environments using multi-robot teams with limited sensing and communication. Building on HEDAC [190] and recent time-varying GP models [196], our approach enables robots to collaboratively reconstruct and track dynamic spatial fields while continuously adapting to environmental changes.

Our key contributions are:

- A fully distributed ergodic coverage framework for multi-robot exploration and adaptive monitoring of unknown, time-varying spatial processes;
- A novel ergodic metric based on a dynamic goal density function that accounts for both exploration and coverage of key areas in the environment.
- A novel temporal decay mechanism for continuous adaptation to evolving spatial

phenomena;

- Extensive simulations and real-world experiments demonstrating robustness, scalability, and effectiveness in complex, non-convex environments.

5.4.2 Background and Notation

Let us define with $\|\cdot\|_n$ the n-dimensional norm. We define with $\mathbf{R}(\theta)$ the rotation matrix that aligns the local coordinate system according to a direction angle θ. Although real-world coverage problems are typically two-dimensional, the proposed method can be extended to n-dimensional problems. For clarity, we consider a two-dimensional domain $\Omega \subset \mathbb{R}^2$ with a Lipschitz continuous boundary. An arbitrary point in the domain is described with $\mathbf{x} \in \Omega$. An integrable probability density function $\phi : \Omega \to \mathbb{R}_{\geq 0}$ is defined to represent the areas of the environment Ω of highest relevance, such that $\int_\Omega \phi(\mathbf{x})d\mathbf{x} = 1$. Let us define the robots' positions with $\mathbf{z}_i \in \Omega$ and their trajectories with $\mathbf{\Gamma}_i : [0, t] \to \mathbb{R}^2$, for $i = 1, \ldots, R$, where R is the total number of robots. The neighbor set of robot i within radius r is denoted by $N_{i,r}$.

Heat-Driven Area Coverage Framework (HEDAC)

In ergodic control formulations, the ergodic metric is a fundamental concept, typically defined as the difference between the empirical distribution of the robots' trajectories $\mathbf{\Gamma}_i$, and the probability distribution defined onto the environment $\phi(\mathbf{x})$. In the original HEDAC formulation [190], the empirical distribution of the robots' trajectories is approximated using Radial Basis Functions (RBFs). This approach smooths the trajectory representation, replacing the standard SMC formulation, which traditionally represents the empirical distribution $\mathbf{\Gamma}_i$ using Dirac delta functions. The use of RBFs provides a more continuous and analytically manageable representation, enhancing the stability and interpretability of the method.

Therefore, the scalar field representing the spatial distribution of the error is defined as

$$e(\mathbf{x}, t) = c_\phi(\mathbf{x}) - c_\gamma(\mathbf{x}, t), \tag{5.26}$$

where $c_\phi(\mathbf{x})$ represents the coverage density from the robots' trajectories and $c_\gamma(\mathbf{x}, t)$ is the smoothed goal density. Finally, the ergodic metric is given by the L_2 norm of the error $E(t) = \|e(\mathbf{x}, t)\|_2$, which we aim to minimize over time, so that the system is ergodic, that is, $\lim_{t\to\infty} E(t) = 0$. In the HEDAC framework, a stationary heat equation is used to form a potential field to guide robots toward target areas, supporting ergodic coverage in non-convex domains with obstacles [191]. The solution of the heat equation $u(\mathbf{x}, t)$ is used to compute the robot input. We refer the reader to [190] for full details.

Robot Model and Control

We model each robot as a point mass with second-order planar dynamics and a limited field of view (FoV) defined by depth $\mathrm{d_{fov}}$ and angle θ_{fov}, capturing typical sensing

constraints such as those from onboard cameras [199, 200].

The i-th robot state at time t is

$$\boldsymbol{\chi}_i(t) = \begin{bmatrix} \mathbf{z}_i(t) \\ \theta_i(t) \end{bmatrix}, \tag{5.27}$$

where $\mathbf{z}_i(t)$ is the planar position and $\theta_i(t) \in [0, 2\pi)$ is the heading angle. Given a known field $u(\mathbf{x}, t)$ from the heat equation, we define the desired velocity as the normalized gradient of the field $u(\mathbf{x}, t)$ evaluated for $\mathbf{x} = \mathbf{z}_i$, scaled by a proportional gain $k_v > 0$,

$$\mathbf{v}_d(\mathbf{z}_i, t) = k_v \frac{\nabla u(\mathbf{z}_i, t)}{\|\nabla u(\mathbf{z}_i, t)\|_2}. \tag{5.28}$$

Normalizing the field gradient enforces constant-speed motion and avoids stalls from eventual vanishing accelerations, provided $\nabla u \neq 0$. The acceleration dynamics of the i-th robot are

$$\ddot{\boldsymbol{\chi}}_i(t) = \begin{bmatrix} \ddot{\mathbf{z}}_i(t) \\ \ddot{\theta}_i(t) \end{bmatrix} = \begin{bmatrix} \min(k_p(\mathbf{v}_d(\mathbf{z}_i, t) - \dot{\mathbf{z}}_i(t)), a_{\max}) \\ \alpha_i(\theta_i, t) \end{bmatrix}, \tag{5.29}$$

where $k_p > 0$ controls acceleration responsiveness. The angular acceleration $\alpha_i(\theta, t)$ is a PD controller on heading error

$$\alpha_i(\theta_i, t) = \mathrm{sat}_{\alpha_{\max}} \left(k_\theta e_{\theta_i} + d_\theta \frac{de_{\theta_i}}{dt} \right), \tag{5.30}$$

with heading error $e_{\theta_i} = \theta_{d_i} - \theta_i$, where the desired heading θ_{d_i} aligns with the potential field gradient. Gains $k_\theta > 0$, $d_\theta \geq 0$ regulate convergence, and $\mathrm{sat}_{\alpha_{\max}}(\cdot)$ bounds angular acceleration to $\pm\alpha_{\max}$. The proposed strategy is model-agnostic, as its vector field can be tracked by many systems using suitable low-level controllers.

Gaussian Processes

GPs provide a principled framework for inferring latent functions from sparse data while quantifying predictive uncertainty [201, 202]. In this work, we consider an underlying task-dependent spatial field $\phi(\mathbf{x})$ representing a quantity of interest distributed over the environment and sensed by the robots (e.g., temperature, gas concentration, or a vision-based detection likelihood). We model this field as $\phi(\mathbf{x}) \sim \mathrm{GP}(\mu(\mathbf{x}), \Sigma(\mathbf{x}, \mathbf{x}'))$, where $\mu(\mathbf{x})$ is the mean and $\Sigma(\mathbf{x}, \mathbf{x}')$ is the covariance function. Assuming spatial smoothness, we use the squared exponential kernel

$$\kappa_{\mathrm{SE}}(\mathbf{x}, \mathbf{x}') = \sigma_c^2 \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|_2^2}{2l^2} \right), \tag{5.31}$$

with σ_c and l controlling variance and correlation length, respectively. Observations are noisy measurements $y(\mathbf{x}) = \phi(\mathbf{x}) + \mathcal{N}(0, \sigma_n^2)$, where σ_n^2 is the sensor noise. This observation model abstracts the sensing process by assuming pointwise access to the field; more complex or indirect sensor models can be incorporated through an appropriate

measurement mapping. Let $\mathfrak{D}_t = \{\mathbf{X}_t, \mathbf{y}_t, \mathbf{t}_t\}$ as the dataset of N samples collected up to time t, with inputs $\mathbf{X}_t = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, outputs $\mathbf{y}_t = \{y_1, \ldots, y_N\}$, and timestamps $\mathbf{t}_t = \{t_1, \ldots, t_N\}$.

5.4.3 Problem Statement

We consider the problem of enabling a team of mobile robots to autonomously explore, learn, and monitor unknown time-varying spatial processes in complex, non-convex environments. The primary challenges are:

1. Autonomous exploration and reconstruction of a dynamic spatial process.
2. Persistent monitoring of high-interest regions, balancing exploration and coverage in real time.
3. Fully distributed operation, with local coordination and collision avoidance guarantees to improve coverage efficiency.

The problem is grounded in the following constraints:

- *Limited sensing*: each robot perceives the environment only through its onboard sensors (e.g., camera, LiDAR), resulting in a narrow, camera-like field of view and sparse observations.
- *Limited communication*: communication is modeled as range-limited, and only robots within range may share poses and local dataset samples, while those outside the range cannot exchange information.
- *Sparse data acquisition*: due to the limited FoV, each robot collects only partial, spatially sparse observations at each timestep.

Problem. *Design a fully distributed strategy that enables a multi-robot team to explore and estimate an unknown, time-varying spatial process in a complex environment, while continuously balancing exploration and perpetual coverage of key areas.*

5.4.4 Distributed Ergodic Coverage Control

Here, we introduce our proposed fully distributed ergodic coverage control approach within the HEDAC framework from [190]. An overview of the proposed closed-loop system architecture is represented in Fig. 5.23.

Distributed Gaussian Process

Unlike prior ergodic coverage methods that assume full knowledge of a static spatial process, we employ time-varying GPs to enable distributed exploration and persistent monitoring of dynamic environments. Each robot maintains its own local GP and collaborates with neighbors through limited dataset sharing.

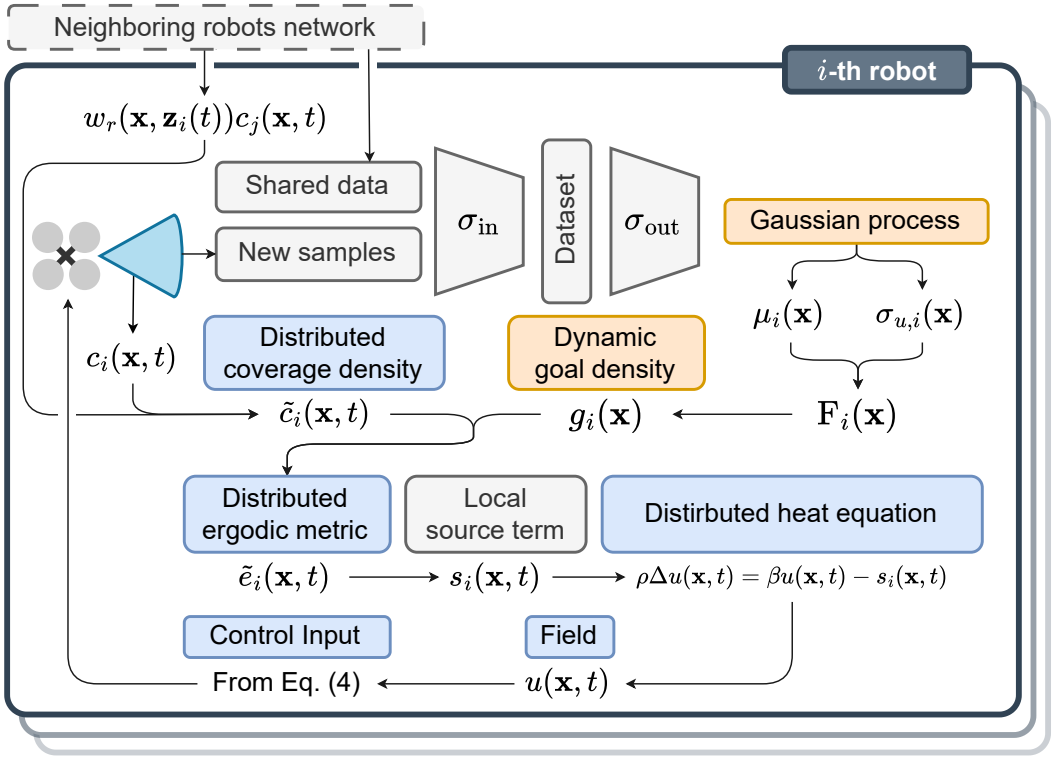


Figure 5.23: System architecture. Blue blocks denote control and dynamical components, orange blocks denote learning-based modules, and gray blocks denote data exchange and filtering.

Following [196] and the formulation in Sec. 5.4.2, predictions are computed at target locations $\mathbf{X}^* = \{\mathbf{x}_1^*, \ldots, \mathbf{x}_M^*\}$ using spatio-temporal decay, such that older or spatially distant samples contribute less to the posterior $(\phi^* \mid \mathfrak{D}_t) \sim \mathcal{N}(\mu(\mathbf{X}^*), \Sigma(\mathbf{X}^*))$, with

$$\begin{aligned} \mu(\boldsymbol{X}^*) &= \tilde{\kappa}(\mathbf{X}^*, \mathbf{X}_t)^\intercal [\tilde{\kappa}(\mathbf{X}_t, \mathbf{X}_t) + \sigma_n^2 \mathbf{I}]^{-1} \mathbf{y}_t, \\ \Sigma(\mathbf{X}^*) &= \kappa(\mathbf{X}^*, \mathbf{X}^*) - \tilde{\kappa}(\mathbf{X}^*, \mathbf{X}_t)^\intercal \\ &\quad \cdot [\tilde{\kappa}(\mathbf{X}_t, \mathbf{X}_t) + \sigma_n^2 \mathbf{I}]^{-1} \tilde{\kappa}(\mathbf{X}^*, \mathbf{X}_t). \end{aligned} \tag{5.32}$$

The decayed covariance matrices are

$$\begin{aligned} \tilde{\kappa}(\mathbf{X}_t, \mathbf{X}_t) &= \kappa_{\text{SE}}(\mathbf{X}_t, \mathbf{X}_t) \odot \mathbf{D}_t \odot \mathbf{T}^D, \\ \tilde{\kappa}(\mathbf{X}^*, \mathbf{X}_t) &= \kappa_{\text{SE}}(\mathbf{X}^*, \mathbf{X}_t) \odot \mathbf{d}_t^\intercal \odot \mathbf{T}^d, \end{aligned} \tag{5.33}$$

with $\odot$ denoting element-wise multiplication. The temporal decay components are

$$\begin{aligned} [\mathbf{T}^d]_i &= T_\tau(\Delta t_i), \quad \Delta t_i = t - t_i, \\ [\mathbf{T}^D]_{ij} &= \begin{cases} [\mathbf{T}^d]_i \cdot [\mathbf{T}^d]_j & \text{if } i \neq j, \\ 1 & \text{otherwise.} \end{cases} \end{aligned} \tag{5.34}$$

As in [196], an exponential decay is used, defined as $T_\tau(\Delta t) = \exp(-\Delta t / \lambda_t)$, where λ_t controls how quickly information loses relevance. However, the spatial decay in [196] is based on sample index proximity rather than true spatial distance, causing inconsistencies, i.e., samples that are far in space but close in index may be incorrectly treated as nearby. Alternative approaches implement decay by augmenting the noise variance model

with a time-dependent term that grows over time [203]. However, this mechanism does not account for the spatial correlations between samples and relies solely on a heuristic aging scheme. To address this, we define spatial decay directly in terms of Euclidean distance

$$\begin{aligned}[\mathbf{D}_t]_{ij} &= \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2/\lambda_s), \\ [\mathbf{d}_t]_i &= \exp(-\|\mathbf{x}_i - \mathbf{z}_i\|_2^2/\lambda_s),\end{aligned} \tag{5.35}$$

where $\mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}_t$ are data samples, $\mathbf{z}_i$ is the current robot position, and λ_s sets the spatial decay rate. By embedding spatial and temporal decay in the kernel, measurement influence decreases as data become older or farther from the robot's current location. The GP therefore assigns higher uncertainty to predictions based on outdated or distant data. The decay parameters λ_t and λ_s control how quickly information loses relevance over time and space, providing a principled way to downweight past measurements while preserving spatial correlations.

To manage the $\mathcal{O}(N^3)$ computational cost of GPs, we adopt threshold-based data filtering inspired by [195, 196]. New samples are added when their predictive standard deviation satisfies $\sigma_{u,i}(\mathbf{x}') \geq \sigma_{\text{in}}$, with $\sigma_{u,i}(\mathbf{x}) = \sqrt{\Sigma_i(\mathbf{x})}$, while existing samples are removed once $\sigma(\mathbf{x}_t) \geq \sigma_{\text{out}}$, with thresholds defined as $\sigma_{\text{in}} = e_{\text{in}}/z$ and $\sigma_{\text{out}} = e_{\text{out}}/z$, and $e_{\text{in}} > e_{\text{out}}$ to prevent premature data removal. This selective data management bounds computation and communication costs while preserving adaptability in dynamic environments. For additional information, readers are referred to [195, 196].

Dynamic Goal Density

We introduce a novel dynamic goal density that adaptively integrates information from the GP, facilitating both exploration and targeted monitoring of the environment. From each robot's GP_i, the estimated spatial process $\mu_i(\mathbf{x})$ and the corresponding uncertainty $\sigma_{u,i}(\mathbf{x}) = \sqrt{\Sigma_i(\mathbf{x})}$ are computed, $\forall \mathbf{x} \in \Omega$, from (5.32). This uncertainty representation effectively distinguishes between explored and unexplored regions, ensuring an informed and adaptive coverage strategy. For every robot, we then combine these two components into a straightforward unified spatial distribution, $\tilde{\text{F}}_i(\mathbf{x})$, which highlights both areas that require monitoring and those that need further exploration. This distribution is defined as

$$\tilde{\text{F}}_i(\mathbf{x}) = (\exp(\mu_i(\mathbf{x})) - 1) + (\exp(\sigma_{u,i}(\mathbf{x})) - 1). \tag{5.36}$$

This formulation directs robots to prioritize exploration when the predictive uncertainty $\sigma_{u,i}(\mathbf{x})$ exceeds the mean estimate $\mu_i(\mathbf{x})$, while favoring the monitoring of high-priority regions otherwise. To eliminate contributions when $\mu_i(\mathbf{x}) \approx 0$ or $\sigma_{u,i}(\mathbf{x}) \approx 0$, we offset both terms by subtracting 1 from $\exp(\mu_i(\mathbf{x}))$ and $\exp(\sigma_{u,i}(\mathbf{x}))$. The resulting normalized unified spatial distribution is then defined as

$$\text{F}_i(\mathbf{x}) = \frac{\tilde{\text{F}}_i(\mathbf{x})}{\int_\Omega \tilde{\text{F}}_i(\mathbf{x}) d\mathbf{x}}. \tag{5.37}$$

Here, $\mu_i(\mathbf{x})$ and $\sigma_{u,i}(\mathbf{x})$ are normalized to ensure that both mean and uncertainty contribute proportionally to the final distribution. This proposed combination effectively

highlights areas that require both monitoring and further exploration, even in complex environments. At each time step, the goal density is updated dynamically as the robot refines its knowledge of the evolving spatial process through the time-varying GP, either via exploration or by incorporating data shared by its neighbors. This update follows the formulation provided in [190] yielding a novel local dynamic goal density computed by the i-th robot as

$$g_i(\mathbf{x}) = \langle \varphi_\gamma, \mathrm{F}_i \rangle_\Omega = \int_\Omega \varphi_\gamma(\mathbf{x}) \mathrm{F}_i(\mathbf{x}) d\mathbf{x}. \tag{5.38}$$

Here, $\varphi_\gamma(\mathbf{x})$ is an RBF acting as a smoothing with shape γ defined as in [190]. As the robot explores, it continually refines its understanding of the environment through the time-varying GPs that facilitate adaptiveness to the dynamic nature of the underlying monitored process.

Distributed Ergodic Control

To enable scalable, fully distributed collaboration, robots must avoid globally sharing their entire coverage histories, which is both unnecessary and computationally inefficient. Since a robot's decisions primarily depend on avoiding redundant coverage with nearby robots, only local information exchange is required. Each robot therefore maintains a local coverage density that includes its own past observations and those of nearby neighbors within a communication range r. This localized strategy significantly reduces communication overhead and acts as a spatiotemporal memory horizon, promoting coordination while relaxing bandwidth demands.

We define the local coverage density of robot i as

$$c_i(\mathbf{x}, t) = \frac{1}{t} \int_0^t p_i(\mathbf{r}_i^\tau) d\tau, \tag{5.39}$$

where $p_i(\mathbf{r}_i^\tau)$ is the instantaneous sensing footprint centered at $\mathbf{r}_i^t = \mathbf{R}(\theta_i(t))(\mathbf{z}_i(t) - \mathbf{x}) \in \mathbb{R}^2$, where $\mathbf{x}$ is the point of interest. For simplicity, we assume all robots are equipped with identical camera-like sensors, leaving the consideration of heterogeneous capabilities for future work. Furthermore, at time t, we define the sensor detection function as an RBF centered within the sensor's FoV, such that

$$p_i(\mathbf{r}_i^t) = \exp(-(\epsilon \mathbf{r}_i^t)^2), \tag{5.40}$$

here, $\epsilon > 0$ controls the shape of the RBF and is a tunable parameter. The explicit inclusion of time in the notation accounts for dynamic changes in the detection function. In complex environments, external factors such as obstacles can partially occlude the sensor's FoV, altering its effective coverage.

To incorporate neighbor information, each robot aggregates only the spatially relevant contributions using a weighting function $w_r(\mathbf{x}, \mathbf{z}_i(t))$ centered at its current location $\mathbf{z}_i(t)$. This yields the distributed coverage density

$$\tilde{c}_i(\mathbf{x}, t) = c_i(\mathbf{x}, t) + \sum_{j \in N_{i,r}} w_r(\mathbf{x}, \mathbf{z}_i(t)) c_j(\mathbf{x}, t), \tag{5.41}$$

where $N_{i,r}$ denotes the set of neighbors within range r, and w_r may be chosen as a hard indicator or a smooth decay kernel (e.g., Gaussian) to enforce locality with a support radius r that is the communication range. Building on this, similar to (5.26), we propose a new distributed error field that is used by each robot to guide its trajectory, defined as

$$\tilde{e}_i(\mathbf{x}, t) = g_i(\mathbf{x}) - \tilde{c}_i(\mathbf{x}, t), \tag{5.42}$$

where $g(\mathbf{x})$ is the dynamic goal density computed as in (5.38). As in [190], the distributed ergodic metric minimized by each robot according to (5.42) is $\mathcal{E}_i = \|\tilde{e}_i(\mathbf{x}, t)\|_2$. This formulation enables distributed operation through neighbor-limited information sharing.

This newly defined ergodic metric serves as the basis for the local source term that each robot independently computes to solve its own heat equation. Following [190], we define the local source for each robot as

$$s_i(\mathbf{x}, t) = \begin{cases} \tilde{e}_i(\mathbf{x}, t)^2 & \text{if } \tilde{e}_i(\mathbf{x}, t) \geq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{5.43}$$

In our formulation, each robot autonomously solves a local heat equation using only its own data and information shared by nearby neighbors. Building on the framework from [190], the distributed heat equation is defined as

$$\begin{aligned} \rho \Delta u(\mathbf{x}, t) &= \beta u(\mathbf{x}, t) - s_i(\mathbf{x}, t), \\ \frac{\partial u}{\partial \mathbf{n}} &= 0, \quad \text{on } \delta\Omega, \end{aligned} \tag{5.44}$$

where $\mathbf{n}$ is the outward unit normal on the boundary $\delta\Omega$. The term $\rho \Delta u(\mathbf{x}, t)$ governs spatial diffusion, with larger ρ producing wider influence. Domain-wide decay is controlled by $\beta u(\mathbf{x}, t)$, where $\beta = \beta_0/|\Omega|$ ensures scale invariance.

Unlike [190], which relies on localized cooling to reduce robot overlap without enforcing guarantees, we explicitly integrate a potential-field-based collision avoidance mechanism, as introduced in [204], to maintain a minimum safety distance r_s between robots. The method in [204] provides theoretical guarantees of maintaining the minimum safety distance, under the mild assumption that the initial deployment does not violate it. This ensures safe, distributed coverage while preserving the ergodic exploration behavior.

Stability Analysis

We study the stability of the distributed scheme induced by the dynamic goal density (5.38). For smoothly time-varying goal densities with bounded temporal derivative, the ergodic error remains uniformly bounded, while in the static case asymptotic convergence is recovered as in classical HEDAC. Instantaneous changes in the spatial process induce discontinuities in the Lyapunov function; however, between such events the closed-loop system either exhibits exponential convergence (in static intervals) or remains uniformly bounded (under smooth variations).

Following classic Lyapunov stability theory, let

$$V_i(t) = \frac{1}{2} \left\| g_i(x,t) - \tilde{c}_i(x,t) \right\|_2^2 = \frac{1}{2} \int \tilde{e}_i(x,t)^2 \, dx \tag{5.45}$$

be the Lyapunov functional associated with the distributed ergodic error of robot i. In general, we note that $V_i(t) \geq 0$ for all t, and $V_i(t) = 0$ if and only if $\tilde{c}_i(x,t) = g_i(x,t)$.

We consider the more general case where the spatial process evolves smoothly in time. Here, asymptotic convergence cannot be guaranteed because the target density is changing; instead, we derive a uniform ultimate bound that depends on the process's rate of change.

Theorem 3 (Bounded tracking for smoothly varying processes)**.** *Consider a multi-robot system composed of R robots, where each robot $i \in \{1, \ldots, R\}$ evolves according to the dynamics in* (5.29) *and applies the control input defined in* (5.28)*. Each robot maintains its own local dynamic goal density g_i, which it updates based on its individual GP as well as information received from neighboring robots. Then, the local ergodic error remains bounded with an asymptotic tight bound proportional to $D_{\max}$.*

Proof. To prove the statement, we show that the local Lyapunov functional $V_i(t)$ introduced in (5.45) is uniformly ultimately bounded, and, in particular, $\lim_{t\to\infty} V_i(t) \leq D_{\max}^2 / 2\alpha^2$. When $\partial g_i / \partial t \neq 0$, the Lyapunov derivative includes the nominal HEDAC term and a disturbance term:

$$\dot{V}_i(t) = -2\alpha \, V_i(t) + \int_\Omega \tilde{e}_i(x,t) \, \frac{\partial g_i}{\partial t}(x,t) \, dx.$$

Applying the Cauchy–Schwarz inequality gives

$$\int_\Omega \tilde{e}_i(x,t) \, \frac{\partial g_i}{\partial t}(x,t) \, dx \leq \left\| \tilde{e}_i(\cdot,t) \right\|_2 \left\| \frac{\partial g_i}{\partial t}(\cdot,t) \right\|_2,$$

and since $\|\tilde{e}_i(\cdot,t)\|_2 = \sqrt{2V_i(t)}$, define

$$D_i(t) := \left\| \frac{\partial g_i}{\partial t}(\cdot,t) \right\|_2.$$

To simplify the inequality, set $W_i(t) = \sqrt{V_i(t)}$, so $V_i = W_i^2$ and

$$\dot{W}_i(t) = \frac{\dot{V}_i(t)}{2\sqrt{V_i(t)}} \leq -\alpha \, W_i(t) + \frac{1}{\sqrt{2}} D_i(t). \tag{i}$$

Assume the temporal derivative of the spatial process is uniformly bounded, i.e., $D_i(t) \leq D_{\max}$. Consider the linear comparison system

$$\dot{y}(t) = -\alpha y(t) + \frac{1}{\sqrt{2}} D_{\max}, \qquad y(0) = W_i(0),$$

whose solution is

$$y(t) = e^{-\alpha t} W_i(0) + \frac{1}{\alpha\sqrt{2}} D_{\max}(1 - e^{-\alpha t}).$$

Since $W_i(t)$ satisfies the inequality (i) and $y(t)$ satisfies the corresponding equality, the comparison lemma ensures $W_i(t) \leq y(t)$ for all $t \geq 0$. Hence,

$$W_i(t) \leq e^{-\alpha t} W_i(0) + \frac{1}{\alpha\sqrt{2}} D_{\max}(1 - e^{-\alpha t}).$$

Taking the limit as $t \to \infty$ yields

$$\lim_{t\to\infty} W_i(t) \leq \frac{D_{\max}}{\alpha\sqrt{2}}.$$

Since $V_i(t) = W_i(t)^2$, the Lyapunov function, and thus the ergodic error, is bounded as follows

$$\lim_{t\to\infty} V_i(t) \leq \frac{D_{\max}^2}{2\alpha^2},$$

and this result completes the proof. □

This result establishes a formal tracking guarantee: slowly varying environments lead to small steady-state errors, while faster variations result in larger, but still finite, errors. The static case arises as a special instance of the theorem with $D_{\max} = 0$, for which the bound collapses to zero, thereby recovering the asymptotic convergence guarantees of the classical HEDAC framework.

Overall, the analysis shows that the proposed distributed controller preserves classical HEDAC stability in static environments, provides bounded tracking performance under smooth temporal variations, and remains well behaved between instantaneous changes, while a full global characterization under arbitrary jump sequences is left for future work.

5.4.5 Experimental Validation

In this section, we conduct a comprehensive set of experiments to evaluate the proposed approach. Simulations were conducted on a PC equipped with an *Intel Core i7* CPU, an *NVIDIA GeForce RTX 4080* GPU, and 16 GB of RAM.

The experiments consider key parameters, including those governing the heat equation, the time-varying GPs used by the robots, and their intrinsic capabilities. Each robot's GP parameters are updated online by fitting the streaming data collected as the robots explore and exchange data, rather than being fixed in advance. We fix all robot-level and heat-equation parameters as reported in Table 5.4, to ensure a consistent comparison across experiments. The parameters in Table 5.4 combine standard practice with task-specific intuition. Robot-level parameters (sensing and communication ranges, controller gains, dynamic limits) are set to reflect typical platforms and ensure smooth, stable coverage. HEDAC constants follow common choices in ergodic-control work, yielding a well-conditioned balance between coverage and trajectory smoothness. GP model parameters follow prior literature [195, 196]: spatial and temporal decay rates encode assumptions on process evolution, while the pruning thresholds e_{in} and e_{out} regulate online updates to retain only informative measurements. These settings were robust across all experiments, but reducing manual tuning and deriving principled guidelines, especially for GP filtering

Table 5.4: Summary of parameters used in the experiments.

Parameter	Description	Value
Robot sensing and communication		
d_{fov}	Sensor field-of-view depth	5 m
θ_{fov}	Sensor field-of-view angle	90°
r	Communication range	5 m
r_s	Minimum safety separation	1 m
Robot dynamics and control		
k_p	Linear velocity proportional gain	3
k_θ	Heading proportional gain	1
d_θ	Heading derivative gain	$2\sqrt{k_\theta}$
$a_{\max}$	Max linear acceleration	2.5 m/s
$\alpha_{\max}$	Max angular acceleration	2.5 m/s^2
Heat equation and coverage parameters		
γ	RBF shape parameter (goal density smoothing)	$1/d_{\text{fov}}$
ϵ	RBF shape parameter (sensor footprint)	$1/d_{\text{fov}}$
ρ	Heat diffusion coefficient	1
ξ	Heat equation decay coefficient	1
Gaussian Process parameters		
$\lambda_{s,t}$	Decay length-scale	100
σ_n	Measurement noise std. deviation	0.01
e_{in}	GP inclusion threshold	0.90
e_{out}	GP removal threshold	0.70

and HEDAC weights, remains future work, including establishing theoretical guarantees on boundedness, convergence, and stability of the GP estimates.

We evaluate the proposed method in two scenarios. First, we consider a static spatial process, where baseline approaches typically converge to key regions and cease exploration. In contrast, our strategy is designed to sustain an explore-cover-repeat behavior, reflected by oscillations in the ergodic metric. Second, we assess adaptiveness by comparing against the oracle HEDAC framework [190] in a scenario where the spatial process undergoes an abrupt change in location, requiring renewed exploration. In both cases, the region of interest is modeled as a single Gaussian peak in a complex environment.

This setup highlights the distinguishing behavior of the proposed distributed strategy, which explicitly accounts for sensing constraints such as limited range and restricted field of view, unlike the oracle baseline. Finally, real-world experiments validate the applicability of the approach under practical sensing and operational conditions.

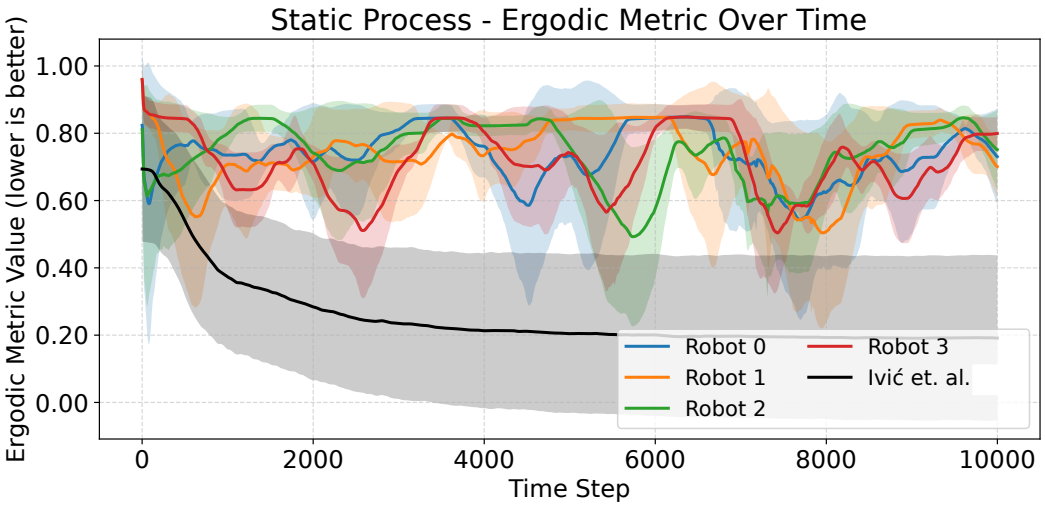


Figure 5.24: Evolution of the ergodic metric over time in a static spatial process. Four robots, each running the proposed distributed strategy, monitor the environment with randomized initial positions and key area locations. The black line shows the oracle baseline from [190], where all robots share a centralized ergodic metric and immediately converge to the key area, without exploration or sustained monitoring. The proposed method, using a time-varying GP and dynamic goal density, balances exploration (higher metric) and focused coverage (lower metric) for effective long-term monitoring.

Simulations: static process

We first evaluate the proposed approach in a baseline scenario with a static spatial process. A team of four robots applies the distributed strategy using time-varying GPs and dynamic goal densities to monitor a static Gaussian field. Figure 5.24 reports the mean and standard deviation of the ergodic metric for each robot over multiple simulation runs, along with the HEDAC baseline [190]. Initial robot positions and the Gaussian peak location are randomized across runs.

Although each robot optimizes a modified ergodic metric with respect to its own dynamic goal density (5.42), performance is evaluated against the ground-truth spatial distribution to enable consistent comparison across methods. This metric quantifies how effectively robots cover the key regions of interest.

As shown in Fig. 5.24, the method exhibits the expected explore-cover-repeat behavior: lower ergodic values correspond to focused coverage of the key region, while higher values indicate exploratory motion. Figure 5.25 illustrates a representative run, showing the environment, the spatial process, and the trajectory of a single robot color-coded by its instantaneous ergodic metric.

The visualization highlights the adaptive nature of the approach: the ergodic metric increases during exploration of less-visited areas and decreases as the robot returns to the key region. The GP decay mechanism enables gradual forgetting of past observations, promoting re-exploration even in this static scenario.

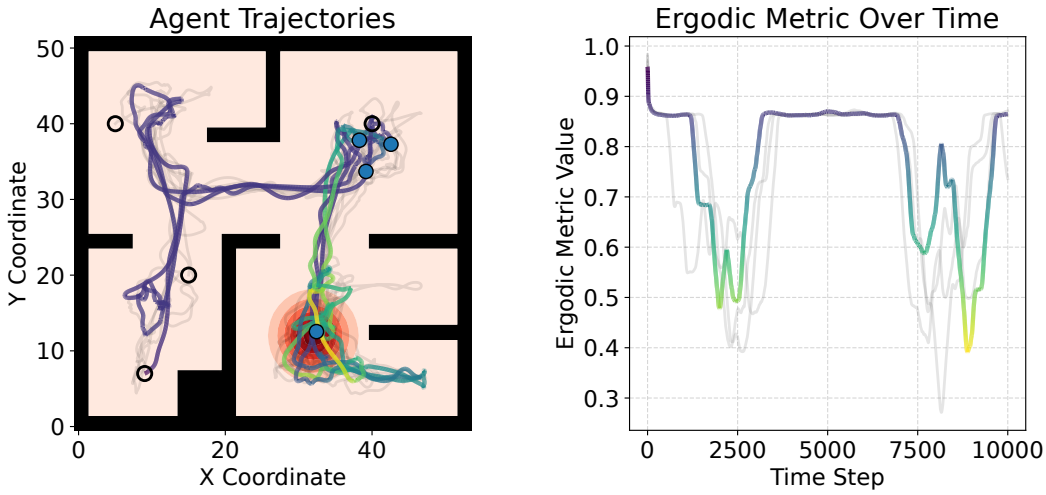


Figure 5.25: A simulation run showing one robot's behavior while monitoring a spatial process represented as a heatmap (color denotes the value of the true underlying process, with red indicating higher intensity). The robot's trajectory is color-coded by its instantaneous ergodic metric; brighter colors indicate lower metric values (i.e., proximity to key regions), while darker colors denote exploratory phases. Black circles denote initial positions and blue dots final positions; other robots' trajectories are shown in gray. This illustrates the robot's ability to dynamically alternate between exploration and focused coverage.

Simulations: time-varying process

To evaluate our distributed strategy in a dynamic setting, we consider a challenging scenario where the spatial process abruptly shifts location within a complex environment. A team of four robots must explore, learn, and monitor the process using our method. To increase difficulty, the shift occurs instantaneously at a predefined time step.

Figure 5.26 shows the ergodic metric (mean and standard deviation) over time across multiple runs, with robot trajectories color-coded. Since distributed ergodic control in time-varying scenarios remains largely unexplored, we compare our approach against the HEDAC framework [190] (black line). At time step 5000, marked by the dashed black line, the spatial process changes location. Our distributed strategy, leveraging time-varying GPs and dynamic goal densities, maintains stable ergodic values and rapidly adapts to the new target. The decay mechanism fosters continuous exploration, enabling fast detection, coverage redirection, and efficient learning of the updated process. By contrast, while the oracle achieves lower initial ergodic values through a globally shared metric, it lacks adaptability: after the process shifts, coverage remains biased toward outdated regions, requiring substantial re-exploration to adapt.

Figure 5.27 illustrates a representative run, focusing on one robot's trajectory and the corresponding ergodic metric before and after the process change. Initially, the robot balances exploration with monitoring of the first key area (red). After the shift at time step 5000, it detects the change, explores, and reallocates coverage to the new key region. The ergodic metric highlights periods when the robot effectively monitors the relevant area, both before and after the shift.

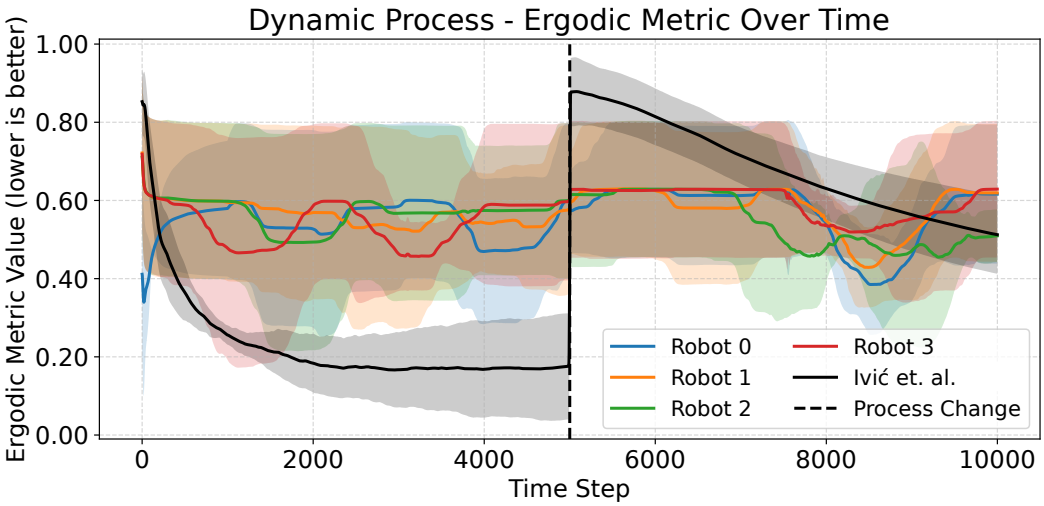


Figure 5.26: Ergodic metric over time during multiple simulation runs, where a team of four robots monitors a time-varying spatial process in a complex environment using the proposed distributed strategy. Each robot's metric is shown in a different color, while the black line represents the baseline HEDAC method [190]. At time step 5000 (dashed black line), the spatial process abruptly changes location. Our method enables adaptive exploration and coverage through time-varying GPs and dynamic goal densities, while the centralized baseline exhibits lower initial metrics but lacks responsiveness to process changes.

Real-world experiments

We validate the proposed distributed strategy in real-world experiments using three *Uvify IFO-S* drones, configured as in simulation. Each drone runs the full algorithm onboard on an *Nvidia Jetson Nano (4 GB)* and interfaces with a *PX4* flight controller. The ergodic controller computes acceleration commands that are internally integrated into velocity setpoints and sent to PX4 for low-level stabilization and constraint handling. Robots communicate over WiFi as independent ROS nodes, exchanging only minimal information (positions and selected dataset samples) with nearby neighbors. A ground station hosts the *ROS core* and *QGroundControl* for supervision and data logging, but performs no inference, control, or coordination.

Each robot generates onboard synthetic scalar measurements to validate the distributed inference-control loop independently of perception. Despite real-world effects such as GPS inaccuracies, IMU drift, estimator latency, and WiFi packet loss, the system remains robust due to its low-bandwidth, neighbor-only communication. Integrating real perception with realistic FoV and sensing-noise models is left for future work. Figure 5.22 shows a representative snapshot.

5.4.6 Conclusion and Future Works

In conclusion, we presented a distributed multi-robot framework for autonomous exploration and adaptive monitoring of time-varying spatial processes in complex environments. Each robot maintains an online time-varying Gaussian Process model from local measurements and selectively shares information with nearby teammates to improve inference without centralized coordination. A dynamic goal density and distributed cov-

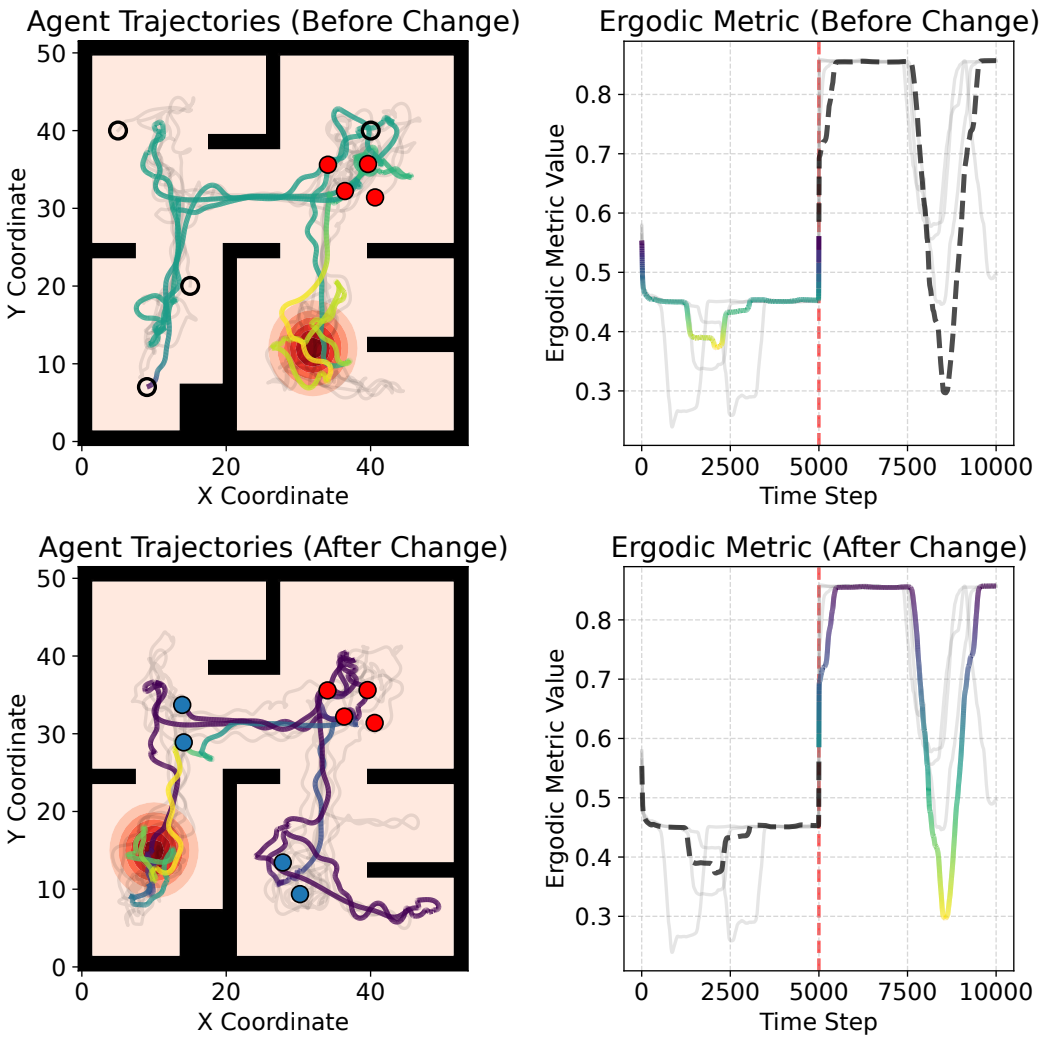


Figure 5.27: Simulation run showing one robot's behavior before and after a sudden change in the spatial process (visualized as a heatmap; color indicates the process intensity, with red denoting higher values). Each subplot displays 5000 time steps. The robot's trajectory is color-coded by its instantaneous ergodic metric (bright colors correspond to lower metric values). Black circles mark initial positions, red dots indicate robot positions at the moment of the process change, and blue dots denote final positions; other robots' trajectories appear in gray.

erage mechanism balance exploration and focused monitoring, while GP temporal decay ensures persistent revisiting of critical regions. Safe and coordinated motion is achieved through a heat-equation–based controller augmented with minimum-distance constraints.

Several limitations motivate future work. First, both the HEDAC-based coverage and time-varying GP filtering rely on parameters that could benefit from more principled, less parametric designs to reduce tuning and improve robustness. Second, while local stability guarantees are provided for instantaneous changes in the spatial process, a full global analysis under rapid or repeated jumps requires hybrid-systems tools such as dwell-time conditions. Third, explicitly accounting for bandwidth constraints through communication-aware strategies is essential for large-scale deployments. Finally, extending the hardware validation to larger robot teams and integrating real perception with realistic FoV and sensing-noise models are key steps toward practical real-world monitoring applications.

5.5 Online Learning-Enhanced High Order Adaptive Safety Control

While the previous sections utilized Control Barrier Functions (CBFs) for high-level coordination and various learning techniques for perception and mapping, this work addresses the fundamental reliance of CBFs on accurate system dynamics. Although CBFs are an effective model-based tool to formally certify the safety of a system, their success in transferring guarantees to real-world systems is critically tied to model accuracy. For example, payloads or wind disturbances can significantly influence the dynamics of an aerial vehicle and invalidate the safety guarantee. In this work, we propose an efficient yet flexible online learning-enhanced high-order adaptive control barrier function using Neural ODEs. Our approach improves the safety of a CBF-certified system on the fly, even under complex time-varying model perturbations. In particular, we deploy our hybrid adaptive CBF controller on a 38g nano quadrotor, keeping a safe distance from the obstacle, against 18km/h wind.

5.5.1 Introduction

As the complexity of both robotic mechanical systems and tasks increases, such as high-dimensional, nonlinear systems and agile control in quadrotors or quadruped robots, it leads to a broader adoption of learning-based approaches, which often come without safety guarantees. This necessitates rigorous safety guarantees built into the controller by definition. Control barrier functions (CBFs) have become a popular model-based tool for safety-critical control owing to their guarantees on set invariance (i.e., safety) without compromising performance [38, 39]. That is, a safe control is computed by solving a quadratic program (QP) in a closed-loop system. In practice, however, CBF-certified systems can sometimes fail to meet safety requirements due to an inaccurate model. This inaccuracy is often caused by a model mismatch or by external forces, like changes in payload or wind disturbances. The adaptive CBFs (aCBFs) compensate for complex dynamic residuals and improve safety in real-world systems, serving as a viable solution for the aforementioned safety violations.

Inspired by the success of data-driven approaches that learn the model of complex dynamical systems [205, 206, 207, 208], we propose a learning-enhanced high-order adaptive CBF (HO-aCBF) that improves safety performance online, handling time-varying model perturbations. Our hybrid approach combines a nominal model with a neural network residual model to efficiently estimate the true dynamics using Knowledge-based Neural ODEs (KNODE) [206]. The model trains on state-control data and allows data sampling with different time intervals (typical for a real-time system). Our hybrid model approach offers high data efficiency compared to a pure learning-based approach by utilizing a nominal model, yet has the rich expressiveness of a neural network to capture complex residuals online.

Neural-Fly [208] learns an adaptation term in its control law directly and achieves online adaptation to severe wind disturbances, but requires a significant offline training phase

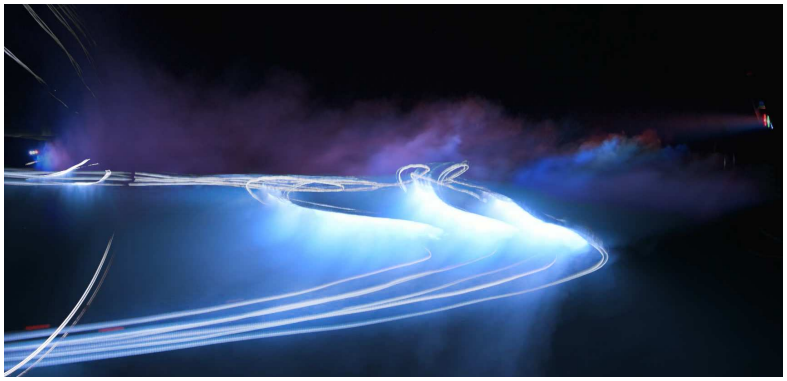


Figure 5.28: A 38g nano quadrotor tracks a circular trajectory while keeping a safe distance from the obstacle, against an 18km/h wind. The safety is improved on-the-fly, i.e., the quadrotor moves further away from the obstacle after experiencing the wind once.

of approximately 12-minute flight data under diverse wind conditions. In contrast, our method learns the model online and improves performance via a model-based controller, and does not require offline training, while supporting it when desired. Gaussian Processes (GPs) approaches, such as PILCO [209], learns dynamics for data-efficient policy search in reinforcement learning, and online GP residual learning [210] has been applied for model predictive control. While the above methods provide alternative strategies for learning dynamics, they differ from our approach, which embeds learned dynamics directly into CBF constraints to improve safety online.

Recent studies have shown that aCBFs (specifically, residual learning approaches) are effective at improving safety under model perturbations [211, 212]. However, these methods require offline training, which leads to safety violations when the residuals are out of the training distribution. Our learning-enhanced aCBF, owing to its hybrid model approach, can adapt to unseen residuals online, making it well-suited for real-world systems. In addition, in [211, 212], supervision of time derivatives of a control barrier function is required, which is obtained from noisy numerical differentiations. Instead, we use the Neural ODEs to approximate the residuals directly from observable state-control data. In practice, most CBF constraints for a mechanical system have relative degrees larger than one, thus requiring a high-order CBF (HOCBF). Our HO-aCBF algorithm generalizes the standard aCBF to handle high-order safety-critical constraints for a larger category of systems.

Robust CBFs guarantee safety against model perturbations by adding a robustifying term in CBF constraints [213]. Robust CBFs, however, require knowing the bounds of the model uncertainty and result in overly conservative controllers. Recently, robust adaptive CBFs (RaCBFs) have been proposed [214, 215, 216]. In their formulation, an aCBF is used to mitigate over-conservativeness by estimating parametric uncertainty. The authors in [217] extend the concurrent-learning aCBFs in [218] to HO-aCBFs. In these works, a strong assumption is made that the residual is a linear model combining a known knowledge matrix and a learnable constant vector. In practice, this matrix may not be available, and the uncertainty is not constant, thus breaking their safety guarantees. Extensive tuning is also required for the concurrent-learning approach.

Unlike aCBF frameworks in [214, 215, 216, 217, 218], our approach does not require prior

knowledge or assumption of residuals, captures time-varying dynamics, and is robust to tuning. In experiments, we demonstrate that HO-aCBF controllers are insufficient to capture time-varying residuals even with increased state-control data, resulting in inadequate performance. Additionally, our hybrid approach has high data efficiency compared to residual learning approaches in [211, 212], making it suitable for online adaptation. These learning frameworks [211, 212] require estimating time derivatives of a control barrier function using numerical differentiation. Instead, our approach learns the residual dynamics directly from state-control data.

The contributions of this work are twofold:

1. a learning-enhanced high-order adaptive CBF that improves safety under complex time-varying model perturbations; and

2. an online hybrid approach using Knowledge-based Neural ODEs that solves CBF optimization with learnable parameters at 100Hz and trains online.

We demonstrate our controller's efficacy in simulation with different nonstationary residuals, benchmarked with HOCBFs [42], HO-aCBFs [217, 218] (with different hyperparameters). In physical experiments, we deploy our algorithm on a nano quadrotor weighing only 38g against 18km/h turbulent wind and efficiently improve safety performance on the fly.

5.5.2 Background: Knowledge-based Neural ODEs (KNODE)

KNODE [206] is an efficient data-driven approach that approximates a dynamical system relying on state-control pairs, i.e., $\mathbf{z} := [\mathbf{x}; \mathbf{u}]$. Its efficiency comes from combining a nominal model (typical for a mechanical system) with a neural network, learning merely the residual dynamics. The hybrid model is given by

$$\dot{\mathbf{x}} = \tilde{f}(\mathbf{z}) + \hat{f}_{\boldsymbol{\theta}}(\mathbf{z}) = f_h(\mathbf{z}, \hat{f}_{\boldsymbol{\theta}}(\mathbf{z})), \tag{5.46}$$

where $\tilde{f}$ denotes a nominal model and $\hat{f}_{\boldsymbol{\theta}}$ denotes the residual model estimated by a neural network parameterized with $\boldsymbol{\theta}$. The neural ODE-based approach learns the model of a dynamical system. Instead of learning in the state space, it learns its dynamical model directly and predicts the state using ODE solvers, thus allowing observations sampled with different time intervals (typical in a physical real-time system). An efficient learning approach preserves the physics of a dynamic model, making it well-suited for our online HO-aCBF controller.

5.5.3 Problem Formulation

Consider a robot that follows a reference trajectory with a desired control $\mathbf{u}_{\text{des}}$ and avoids obstacles by certifying the desired control using a CBF with unknown dynamic perturbations. The guarantee provided by CBF can be compromised due to the inaccurate model.

We adopt the formulation in [214], and assume the true dynamics in the following form

$$\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u} + d(\mathbf{x}), \tag{5.47}$$

where $f(\mathbf{x}) + g(\mathbf{x})\mathbf{u}$ forms a nominal model and $d(\mathbf{x})$ includes all unmodeled model residuals and is allowed to change over time. The problem we want to solve is to recover the safety guarantee of the CBF-based controller by identifying the complex model perturbation $d(\mathbf{x})$ using online state and control data.

We propose an efficient learning-enhanced high-order aCBF that compensates for unknown complex time-varying and nonlinear dynamic perturbations $d(\mathbf{x})$ and improves the system safety performance online. Our hybrid model combines the nominal model with a neural network residual model $\hat{d}_{\boldsymbol{\theta}}$ parameterized with $\boldsymbol{\theta}$, using Knowledge-based Neural ODEs [206] that obey physical laws. The residual model is trained on observable state-control data, i.e., $\mathbf{z} := [\mathbf{x}; \mathbf{u}]$. Furthermore, we generalize the proposed aCBFs to high-order aCBFs to handle a larger category of systems.

5.5.4 Learning-enhanced High-order Adaptive Control Barrier Functions

CBF-based controllers are safe by definition for a known dynamical system. In practice, such safety guarantees can be violated due to model inaccuracies, such as not fully capturing the true dynamics of the robot. In this work, we propose an efficient learning-enhanced high-order aCBF using knowledge-based neural ODEs, or NODE-HO-aCBF for short. Our approach compensates for complex model perturbations from observable state-control data, improving safety on the fly. In Sec. 5.5.4, we introduce our learning-enhanced NODE-HO-aCBF optimization framework. In Sec. 10, we introduce KNODE for learning residual dynamics online. In Sec. 5.5.5, we present an example of a double integrator that maintains a safety distance from obstacles using our NODE-HO-aCBF controller. The overall framework for NODE-HO-aCBF is depicted in Fig. 5.29.

Learning-enhanced HO-aCBFs

Simplified models are often chosen for their efficiency and analytical advantages in robotics applications. These models, however, do not capture the true dynamics. For example, a quadrotor cannot follow a double integrator trajectory precisely due to its kinematic constraints. Time-varying external residual dynamics, such as wind turbulence or downwash effect between quadrotors flying in close proximity, also introduce an unmodeled nonlinear term to the model. Our approach enables a high-order adaptive CBF to learn such model mismatches and external dynamics from data, therefore, improving safety on the fly.

We assume the true dynamics as in (5.47), where we consider a time-varying model perturbation term $d(\mathbf{x})$, which depends on the robot state and is allowed to change over time. We formulate our hybrid model, combining the known nominal dynamics with a

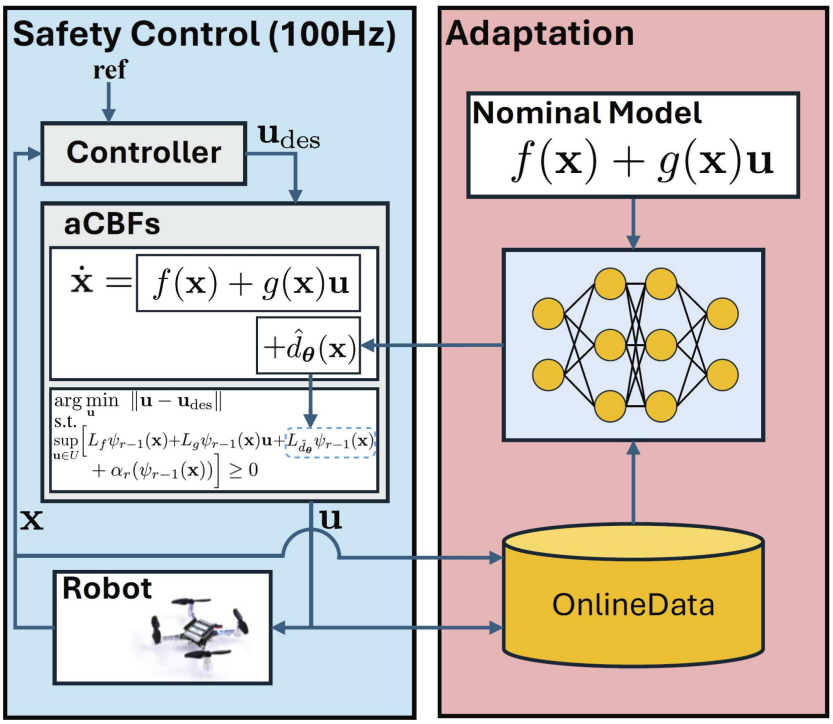


Figure 5.29: The picture depicts the system overview of our NODE-HO-aCBF framework. The safety control solves the NODE-HO-aCBF QP, which uses the most recently trained model, at 100Hz to certify the desired control. The state and control input data are stored in an online queue for training. The hybrid adaptation module combines a nominal model with a learning-based term that approximates the unknown residual dynamics online.

neural ODE approximated residual dynamics, as follows

$$f_h(\mathbf{z}, \hat{d}_{\boldsymbol{\theta}}(\mathbf{x})) = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u} + \hat{d}_{\boldsymbol{\theta}}(\mathbf{x}), \tag{5.48}$$

where $\hat{d}_{\boldsymbol{\theta}}$ denotes the residual dynamics estimated by a neural network parameterized with $\boldsymbol{\theta}$.

As a model-based approach, successfully transferring its guarantee to real-world systems is inherently tied to the fidelity of the system model. To handle model perturbation, we consider the hybrid model from (5.48) and propose a Neural ODE enhanced high-order adaptive control barrier function, namely, NODE-HO-aCBF. Similar to a typical HOCBF QP formulation, the objective is to minimize the deviation between the safe control and the desired control input. To adapt to the unknown residual dynamics, we modify the HOCBF constraint in (2.18) to (5.49b). The Lie-derivative of residual term $\hat{d}_\theta$, i.e., $L_{\hat{d}_\theta}$, is added to the HOCBF constraint to certify the hybrid system model. We formulate the QP for NODE-HO-aCBF as follows:

$$\underset{\mathbf{u}}{\text{argmin}} \; ||\mathbf{u} - \mathbf{u}_{\text{des}}|| \tag{5.49a}$$

$$\text{s.t.} \; \sup_{\mathbf{u}\in U}\Big[L_f\psi_{r-1}(\mathbf{x}) + L_g\psi_{r-1}(\mathbf{x})\mathbf{u} + L_{\hat{d}_{\boldsymbol{\theta}}}\psi_{r-1}(\mathbf{x}) + \alpha_r(\psi_{r-1}(\mathbf{x}))\Big] \geq 0, \tag{5.49b}$$

Algorithm 3: Learning-enhanced HO-aCBF Control Loop

Initialize the current time, latest model update time, total duration and training data as t_i, t_m, t_T, and $\mathcal{Q}_{\text{train}}$
$t_i \leftarrow 0$; $\mathcal{Q}_{\text{train}} \leftarrow [\,]$
while $t_i < t_T$ **do**
    if *New hybrid model is available* **then**
        NODE-HO-aCBF loads the new model
    PID generate a desired control $\mathbf{u}_{\text{des}}$ given robot state $\mathbf{x}$ and a reference $\mathbf{x}_{\text{ref}}$
    Solve QP in (5.49) and generate a safe control $\mathbf{u}$
    Robot updates the state using $\mathbf{u}$
    Append $\mathbf{z}_{t_i}$ to the queue $\mathcal{Q}_{\text{train}}$ (pop out the oldest data if the queue is full).
    $t_i \leftarrow$ current time

where,

$$\psi_r = \dot{\psi}_{r-1} + \alpha_r(\psi_{r-1}) \tag{5.50}$$

$$= \left(\frac{\partial \psi_{r-1}}{\partial \mathbf{x}}\right)^\top (f + g\mathbf{u} + \hat{d}_{\boldsymbol{\theta}}) + \alpha_r(\psi_{r-1}) \tag{5.51}$$

$$\psi_0 = h(\mathbf{x}). \tag{5.52}$$

To compute ψ_r, the algorithm requires taking the $(r-1)$-th derivative of the residual dynamics $\hat{d}_{\boldsymbol{\theta}}$ estimated by the neural network w.r.t. the state, i.e. $\nabla_{\mathbf{x}}^{r-1}\hat{d}_{\boldsymbol{\theta}}$, which can be computed by automatic differentiation from machine learning frameworks or numerical methods. Our learning-enhanced controller runs at 100Hz to provide high-frequency safety control for the real-time system. At each iteration step (see Algo. 3), the NODE-HO-aCBF loads the hybrid model if a new learned model is ready. A PID controller takes the current state $\mathbf{x}$ and reference $\mathbf{x}_{\text{ref}}$ to generate a desired control $\mathbf{u}_{\text{des}}$. By solving the QP in (5.49), a safe control $\mathbf{u}$ is produced and executed by the robot. The state-control pair $\mathbf{z} := [\mathbf{x}, \mathbf{u}]$ is added to the online queue for training.

KNODE Online Learning

An unknown nonstationary residual, as prior models quickly become obsolete, requires continuous online adaptation to identify the current true dynamics on the fly.We employ a training queue $\mathcal{Q}_{\text{train}}$ to store state-control pairs in a first-in, first-out (FIFO) manner, ensuring that the hybrid model f_h is trained online with the most recent data. The neural ODE learns a dynamical system directly. Consider an objective function $\mathcal{L}$ that penalizes the error between the predictions and observations by constraining the prediction state to obey a learning-based ODE, defined as follows,

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{m-1}\sum_{i=1}^{m-1}\int_{t_i}^{t_{i+1}} \delta\left(t_s - \tau\right) \|\hat{\mathbf{x}}_\tau - \mathbf{x}_\tau\|^2 d\tau + \lambda||\boldsymbol{\theta}||_2^2 \tag{5.53}$$

where m is the number of samples in $\mathcal{Q}_{\text{train}}$, $\delta(\cdot)$ is the Dirac delta function, t_s is any sampling time in $\mathcal{Q}_{\text{train}}$, $\mathbf{x}_\tau$ is a shorthand for $\mathbf{x}(\tau)$. For the learned model to generalize, we use the norm-2 regularization with a weight λ [219]. The prediction state is computed

Algorithm 4: Online Dynamic Learning

```
Initialize the current time, total duration, training data as t_i, t_T, Q_train, and the
  hybrid model in (5.48)
while t_i < t_T do
    if Q_train ≠ ∅ then
        Train the hybrid model by optimizing (5.55) using Q_train
        Save the new model
    t_i ← current time
```

as follows:

$$\hat{\mathbf{x}}_\tau = \mathbf{x}_{t_i} + \int_{t_i}^{\tau} f_h(\mathbf{z}_w, \hat{d}_{\boldsymbol{\theta}}(\mathbf{x}_w))dw. \tag{5.54}$$

The learnable parameters are obtained by solving the following optimization problem:

$$\underset{\boldsymbol{\theta}}{\text{argmin}} \ \mathcal{L}(\boldsymbol{\theta}) \tag{5.55a}$$

$$\text{s.t. } \dot{\mathbf{x}} = f_h(\mathbf{z}, \hat{d}_{\boldsymbol{\theta}}(\mathbf{x})). \tag{5.55b}$$

In practice, a gradient-based optimization tool, such as PyTorch, can be employed to solve the above optimization problem efficiently.

Our algorithm can also be used in offline mode. In offline mode, we collect state-control data by operating the vehicle using the HOCBF safety controller for a certain duration. We pre-train the neural network to approximate the residual dynamics using all the data collected and deploy the NODE-HO-aCBF controller with the learned $\hat{d}_{\boldsymbol{\theta}}(\mathbf{x})$.

5.5.5 Simulations

To demonstrate the efficacy of the proposed algorithm, we apply it in a simulation with a double integrator model with different types of model residuals.

NODE-HO-CBFs with Double Integrator Nominal Model

We consider an example where a robot is required to follow a reference trajectory in a workspace containing obstacles $\mathcal{O}_1, \ldots, \mathcal{O}_{N_{\text{obs}}}$, from which the robot is required to keep a safety distance D_s. The robot state $\mathbf{x} = [\mathbf{r}; \dot{\mathbf{r}}] \in \mathbb{R}^6$ represents its position and velocity. The control input $\mathbf{u} \in \mathbb{R}^3$ is the acceleration. We consider the nominal model as a double integrator, and the hybrid model can be written

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} + \hat{d}_{\boldsymbol{\theta}}(\mathbf{x}), \tag{5.56}$$

where $A = [\mathbf{0}, \mathbf{I}; \mathbf{0}, \mathbf{0}] \in \mathbb{R}^{6\times6}$, $B = [\mathbf{0}; \mathbf{I}] \in \mathbb{R}^{6\times3}$. Here $\mathbf{0} \in \mathbb{R}^{3\times3}$, $\mathbf{I} \in \mathbb{R}^{3\times3}$ are the zero and identity matrices, respectively.

At each iteration step, we obtain a reference of position, velocity, and acceleration, i.e.,

$[\mathbf{r}_{\text{des}}; \mathbf{v}_{\text{des}}; \mathbf{a}_{\text{des}}]$. The desired control $\mathbf{u}_{\text{des}}$ is generated using a PID controller. We investigate the CBF that ensures safety by maintaining a minimum distance from obstacles. Specifically, we define the CBF corresponding to the i-th obstacle as

$$h_i(\mathbf{x}) = ||\mathbf{r} - \mathbf{r}_{\text{obs},i}||_2^2 - D_s^2, \forall i \in \{1, \dots N_{obs}\}, \tag{5.57}$$

where $\mathbf{r}_{\text{obs},i}$ denotes the position of i-th obstacle. The relative degree of h_i is two with respect to the model in (5.56), according to Def. 2.2.4. We consider the extended class $\mathcal{K}$ functions as $\alpha_i(h) = \gamma_i h$. Thus the optimization is formulated as follows:

$$\begin{aligned} \underset{\mathbf{u}}{\text{argmin}} \; & ||\mathbf{u} - \mathbf{u}_{\text{des}}|| && \text{(5.58a)} \\ \text{s.t.} \; & \sup_{\mathbf{u} \in U} \Big[(L_f^2 h + L_{\hat{d}_{\boldsymbol{\theta}}}^2 h) + L_f L_{\hat{d}_{\boldsymbol{\theta}}} h + L_{\hat{d}_{\boldsymbol{\theta}}} L_f h \\ & \quad + [L_g (L_f h + L_{\hat{d}_{\boldsymbol{\theta}}} h)]\mathbf{u} + (\gamma_1 + \gamma_2)(L_f h + L_{\hat{d}_{\boldsymbol{\theta}}} h) \\ & \quad + \gamma_2 \gamma_1 h \Big] \geq 0. && \text{(5.58b)} \end{aligned}$$

The solution of the optimization (5.58) is the control $\mathbf{u}$ for our hybrid learning-enhanced high-order aCBF controller.

Simulation Setups

A perturbed double integrator is implemented as $\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} + d(\mathbf{x})$, where we assume the residual dynamics $d(\mathbf{x})$ is unknown to the robot in the simulation. Three types of residuals are considered, namely, "Attractive", "Repulsive", and "Time-varying". In "Attractive" and "Repulsive" experiments, residuals attract and repulse the robot towards and away from the center of a forbidden region where the robot is not allowed to enter, i.e., $d_{\text{att}}(\mathbf{x}) = [\mathbf{0}; -k\mathbf{r}]$, $d_{\text{rep}}(\mathbf{x}) = [\mathbf{0}; k\mathbf{r}]$, where $k = 0.4$. In "Time-varying" experiment, residuals change between repulsive and attractive over time, i.e., $d_{\text{var}}(\mathbf{x}) = [\mathbf{0}, k \sin(t)\mathbf{r}]$, where $k = 1.0$. A Runge–Kutta (RK4) numerical solver with sampling time of $\Delta t = 0.01$s is used to simulate the system response given the control. A closed-loop PID controller synthesizes the desired closed-loop response $\mathbf{u}_{\text{des}}$. We assume the robot state-control pair $\mathbf{z} := [\mathbf{x}; \mathbf{u}]$ is observable and can be measured at 100Hz. The closed-loop control is applied at a rate of 100Hz. We use a similar neural network architecture to [220]. The value of $\hat{d}_{\boldsymbol{\theta}}$ is computed as the addition of outputs of three parallel neural networks with the same architecture and input $\mathbf{x}$. Each neural network has two hidden layers, each with only 16 neurons, where the hyperbolic tangent activation function is applied after the first hidden layer.

The robot is commanded to follow a circular reference trajectory of radius 2m at height 1m, centered at the x-y plane origin, where a spherical forbidden region of radius $D_s = 3$m is also located. Thus, the CBF must be effective to avoid safety violations. Our NODE-HO-aCBF controller supports both offline and online modes as described in Sec. 10. For our online NODE-HO-aCBF controller, we maintain an FIFO queue of 10 seconds of online data for training. For the offline model, we collect 40 seconds of state-control data for training.

Benchmarking. We compare the safety performance between: 1) HOCBFs baseline [42], where no external residual exists, 2) HOCBFs baseline, 3) HO-aCBFs baseline [217, 218], 4) our online NODE-HO-aCBFs, and 5) our offline NODE-HO-aCBFs. The HOCBF controller without the external residual sets the baseline safety performance of the system. The comparison of online and offline NODE-HO-aCBFs with HOCBFs baseline demonstrates that our algorithm improves safety performance by adapting to the residual term. The comparison of online and offline NODE-HO-aCBFs with the adaptive HOCBFs highlights that our algorithm can handle time-varying residuals, and our algorithm does not require knowledge of the residual dynamics nor extensive tuning. For all the approaches, we use extended class $\mathcal{K}$ functions $\alpha_i(h) = \gamma_i h$, where $\gamma_i = 2$ for $i = 1, 2$, in CBF constraints. For the HO-aCBFs baseline [217, 218], it is assumed that $d(\mathbf{x}) = Y(\mathbf{x})\hat{\boldsymbol{\vartheta}}$, where $Y(\mathbf{x}) \in \mathbb{R}^{p\times s}$ is a known matrix and $\hat{\boldsymbol{\vartheta}} \in \mathbb{R}^s$ is an unknown constant vector to estimate. To update the parameter $\hat{\boldsymbol{\vartheta}}$, HO-aCBFs require computing its update law $\dot{\hat{\boldsymbol{\vartheta}}}$ using an adaptation gain $\kappa \in \mathbb{R}_{>0}$ and a positive definite gain matrix $\Gamma \in \mathbb{R}^{s\times s}$ (see [217, eq. (14)]). We extensively tune the gains κ and Γ in their formulation of $\dot{\hat{\boldsymbol{\vartheta}}}$ and assume a known $Y(\mathbf{x}) = [\mathbf{0}_{3\times 3}; -r_1, 0, 0; 0, -r_2, 0; 0, 0, -(r_3 - 1)]$. Note that this is a strong assumption and we usually do not have prior knowledge about $Y(\mathbf{x})$ in practice. To compute the HO-aCBF controller online at 100Hz, we can only use a history stack of 0.5s. As we will point out in the following section, increasing the HO-aCBF's history stack does not improve its performance significantly, yet largely increases its runtime, prohibiting its performance. For HO-aCBF baseline [217], we set the bounds of the estimate $\hat{\boldsymbol{\vartheta}}$ between $[-4, -4, -4]$ to $[4, 4, 4]$, which sufficiently cover the range of the underlying dynamic residual term given our choice of $Y(\mathbf{x})$ matrix.

Simulation Qualitative Results

We demonstrate the typical trajectories of baseline and our NODE-HO-aCBF controllers with three residual types in Fig. 5.30. In the "Attractive" experiment, the HOCBF baseline fails to adapt to external residuals, resulting in a significant safety violation and deviating from the safety boundary, as depicted by the green trajectory in the top-left panel. The HO-aCBFs, when well-tuned and given a high-fidelity knowledge matrix $Y(\mathbf{x})$, are able to compensate for the residuals. When a constant matrix $Y(\mathbf{x})$ is applied, the HO-aCBF controller significantly violates safety and settles within the safety boundary, showcasing reduced adaptability, as indicated by the brown trajectory. Our approach, NODE-HO-aCBF, both its online and offline modes, are able to adapt to external residuals and settle safely outside of the forbidden region without tuning. In the "Repulsive" experiments, instead of being attracted towards the center and violating the safety, the baseline HOCBF controller settles farther from the center. The HO-aCBF baseline, similar to the "Attractive" experiment, is only able to adapt when a high-fidelity knowledge matrix is available and the controller is well-tuned. NODE-HO-aCBFs, again, are able to adapt and settle at the boundary of the forbidden region. In both experiments, the residual term convergence rate of HO-aCBFs is higher than that of NODE-HO-aCBFs. However, due to the limitation on expressiveness of HO-aCBF, it is not practical for real-world systems under complex residual dynamics. We high-

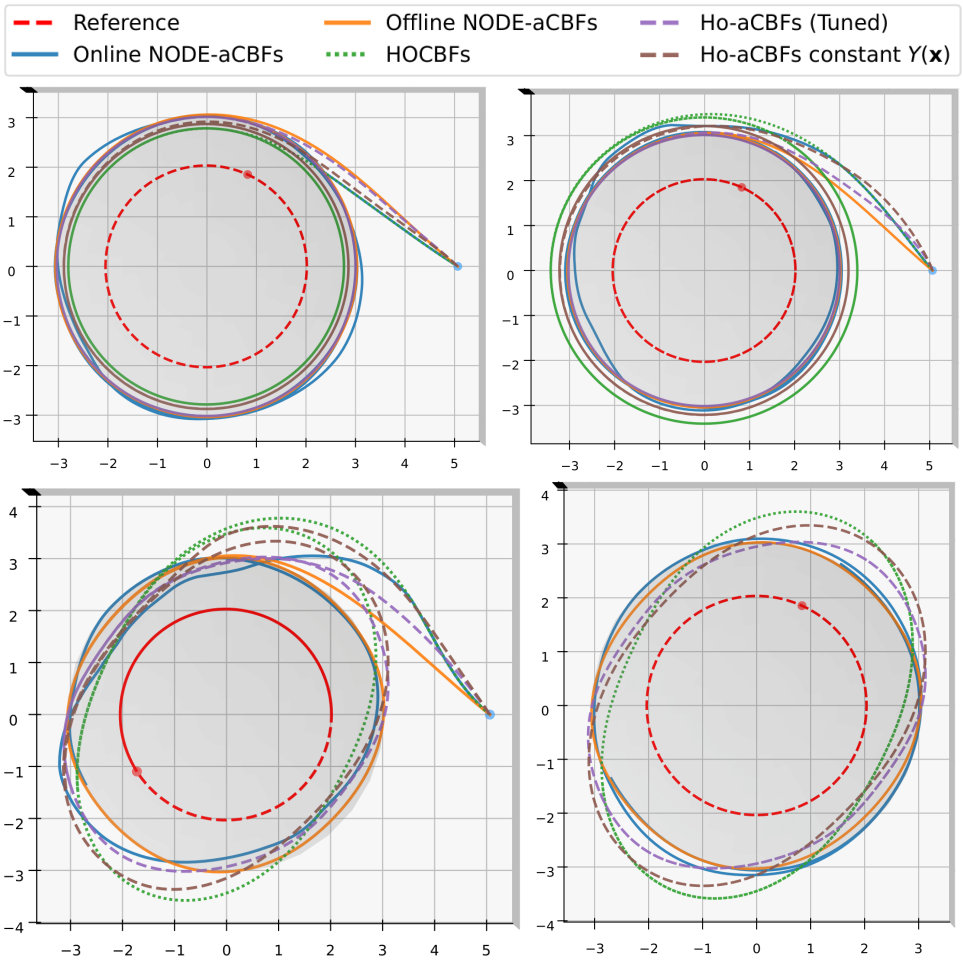


Figure 5.30: Qualitative result of our algorithms compared to baseline controllers with the presence of external residual dynamics. The blue dot represents the initial state, and the red dot represents the current reference. The Gray sphere is the forbidden region. Top left: "Attractive" residual, Top right: "Repulsive" residual, Bottom left: "Time-varying" residual from 0-20s, Bottom right: "'Time-varying" residual from 20-40s.

light NODE-HO-aCBFs' adaptive performance in the "Time-varying" experiment, as all baselines fail to adapt, even a well-tuned HO-aCBF controller. In the following analysis, we note that, even with an increased history stack for the HO-aCBF baseline, it cannot adapt to such a time-varying residual, but invalidates the desired control frequency and its real-time performance as its runtime grows with the history stack. Both our online and offline controllers can adapt and have significant advantages in settling closer to the safety boundary, i.e., improve the safety.

Next, we discuss tuning fragility, strong dependence on assumptions, and limited expressiveness of the HO-aCBF baseline. We tune the gains κ and Γ in the HO-aCBF controller, and provide a strong assumption of the knowledge matrix $Y(\mathbf{x})$. In Fig. 5.31 (left), the controller is executed in the "Attractive" experiment. Note that a small $\Gamma = (1e-5)\mathbf{I}$, a large $\Gamma = \mathbf{I}$, or a constant matrix $Y(\mathbf{x}) = [\mathbf{0}_{3\times3}; -\mathbf{I}_{3\times3}]$ (no knowledge) all lead to a quickly degenerated performance of the HO-aCBF controller and result in significant safety violations. In Fig. 5.31 (right), we use the tuned HO-aCBF controller and increase its history stack $\mathcal{Q}_{\text{train}}$ to 5s and 15s (a long history stack invalidates the desired control frequency, so we freeze the simulation when computing the control). However, the controller cannot utilize the long history and still presents significant safety violations.

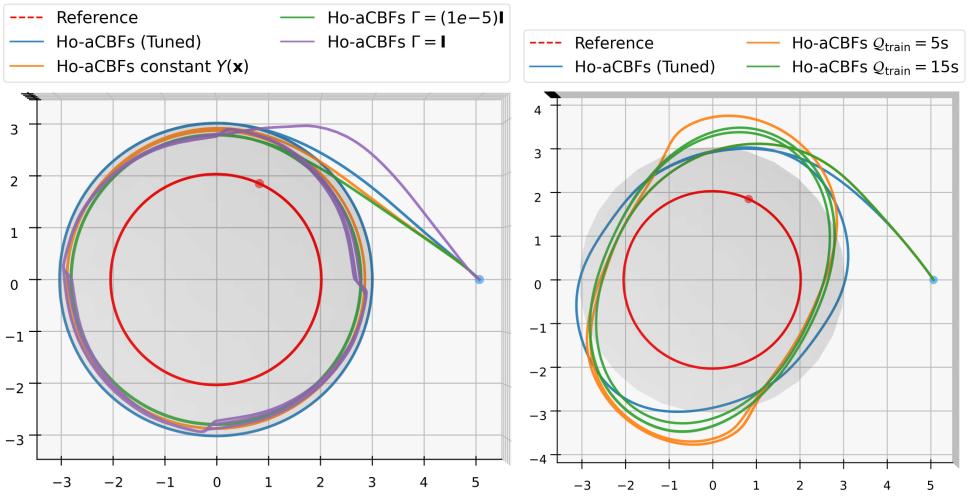
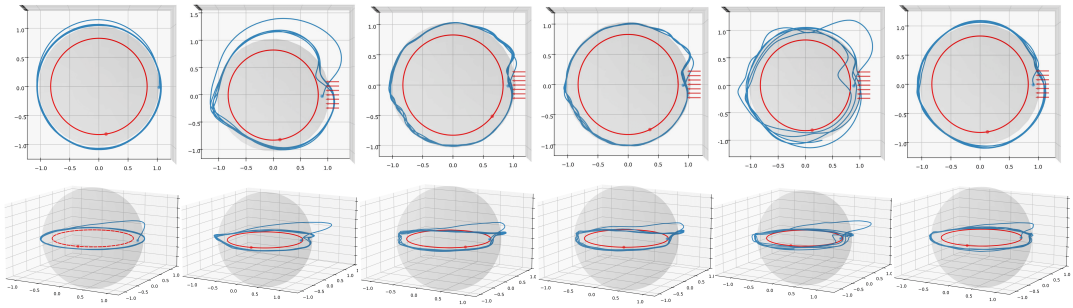


Figure 5.31: Qualitative comparison between baseline HO-aCBF controller with different settings. "Attractive" residual (on the left), and "Time-varying" residual (on the right) are applied in experiments.



(a) **(b)** **(c)** **(d)** **(e)** **(f)**

Figure 5.32: The top views (first row) and the side views (second row) of 60s trajectories using different safety controllers in the physical experiments: (a) HOCBFs with no wind, (b) HOCBFs, (c) HO-aCBFs with state-dependent $Y(\mathbf{x})$, (d) HO-aCBFs with constant $Y(\mathbf{x})$, (e) online NODE-HO-aCBFs, and (f) offline NODE-HO-aCBFs. The gray circle indicates the forbidden region, the red circle indicates the desired trajectory, and the red arrows indicate the location and direction of the wind.

These prominent drawbacks—fragility to gains tuning, reliance on strong knowledge, and limited expressiveness in recognizing the pattern of a long horizon history stack—make it secondary compared to our approach when exposed to time-varying model perturbations.

Simulation Quantitative Results

We use the following quantitative evaluation criteria:

- h_{min}: the minimum value of $h(\mathbf{x})$ during execution.
- h_{neg}: the fractional time for which $h(\mathbf{x}) < 0$ over execution time.
- *Average Distance (Avg.Dist.)*: the average distance to the forbidden region surface over execution time.
- *Average settleDistance (Avg.sDist.)*: same as *Avg.Dist.*, excluding the first 20s in the simulation execution.
- *Signed settleDistance Variance (S.sDist.Var.)*: the variance of the signed distance

Attractive					
Method	$h_{\min}$	$h_{\neg}$	Avg.Dist.	Avg.sDist.	Var.
HOCBF - No residual	0.09	0.0%	0.07	0.01	2.02e-10
HOCBFs	-1.42	95.8%	0.27	0.25	8.20e-11
HO-aCBFs (Tuned)	-0.12	93.0%	**0.06**	**0.01**	9.89e-6
HO-aCBFs $\Gamma = 1$	-1.50	92.6%	0.19	0.15	1.63e-3
HO-aCBFs const $Y(\mathbf{x})$	-0.93	95.1%	0.19	0.16	1.44e-6
Online NODE-HO-aCBFs	-0.87	46.8%	**0.07**	**0.02**	3.93e-4
Offline NODE-HO-aCBFs	**0.01**	**0.0%**	**0.06**	**0.01**	1.45e-5

Repulsive					
Method	$h_{\min}$	$h_{\neg}$	Avg.Dist.	Avg.sDist.	Var.
HOCBF - No residual	0.09	0.0%	0.07	0.01	2.02e-10
HOCBFs	2.34	0.0%	0.42	0.37	5.64e-10
HO-aCBFs (Tuned)	-0.10	89.5%	**0.07**	**0.01**	9.89e-6
HO-aCBFs $\Gamma = 1$	13.31	0.0%	1.49e+9	2.97e+9	4.69e+19
HO-aCBFs const $Y(\mathbf{x})$	0.84	0.0%	0.24	0.18	5.29e-6
Online NODE-HO-aCBFs	-1.61	37.7%	0.13	**0.03**	9.74e-4
Offline NODE-HO-aCBFs	**0.01**	**0.0%**	**0.06**	**0.01**	1.62e-5

Time-varying					
Method	$h_{\min}$	$h_{\neg}$	Avg.Dist.	Avg.sDist.	Var.
HOCBF - No residual	0.09	0.0%	0.07	0.01	2.02e-10
HOCBFs	-2.78	44.7%	0.45	0.39	0.18
HO-aCBFs (Tuned)	-1.98	49.7%	0.27	0.23	0.06
HO-aCBFs $\Gamma = 1$	-6.98	0.0%	1.47e+6	2.94e+6	3.74e+13
HO-aCBFs const $Y(\mathbf{x})$	-2.65	45.4%	0.41	0.36	0.16
Online NODE-HO-aCBFs	-2.98	52.3%	0.20	**0.04**	**2.29e-3**
Offline NODE-HO-aCBFs	**-0.91**	68.5%	**0.10**	**0.05**	3.11e-3

Table 5.5: Quantitative comparison of baseline and proposed controllers under attractive, repulsive, and time-varying residual dynamics. Statistics averaged over 10 trials.

to the forbidden region surface (negative value indicates the robot is inside the forbidden region), excluding the first 20s in the simulation execution.

In Table 5.5, we record the averaged quantitative results from 10 simulation trials. In "Attractive" and "Repulsive" experiments, we observe that HO-aCBFs (Tuned), online NODE-HO-aCBFs, and offline NODE-HO-aCBFs are able to maintain a small *Avg.Dist.* and *Avg.sDist.*, indicating their success in residual adaptation and safety improvement. However, the HO-aCBF has high h_{neg}, indicating that the HO-aCBF controller failed to estimate a high-fidelity residual term to fully recover the safety. Among them, the offline NODE-HO-aCBF has the $h_{\min}$ closest to that of HOCBF with no external residuals and 0% of h_{neg}, indicating the highest fidelity of residual estimations. These three controllers all demonstrate small *S.sDist.Var.*, indicating their settled trajectory maintains a stable distance to the forbidden region surface. All other baselines fail to adapt in the "Attractive" and "Repulsive" experiments. In the "Time-varying" experiments, only our NODE-HO-aCBFs can adapt. According to *Avg.Dist.* and *Avg.sDist.* metrics, both online and offline NODE-HO-aCBFs are significantly closer to the safety boundary and have similar results with HOCBF with no external residuals. According to *S.sDist.Var.*, our online and offline NODE-HO-aCBFs settle on trajectories that variate least from the safety boundary, indicating a stable adaptation. Note that even a well-tuned HO-aCBF

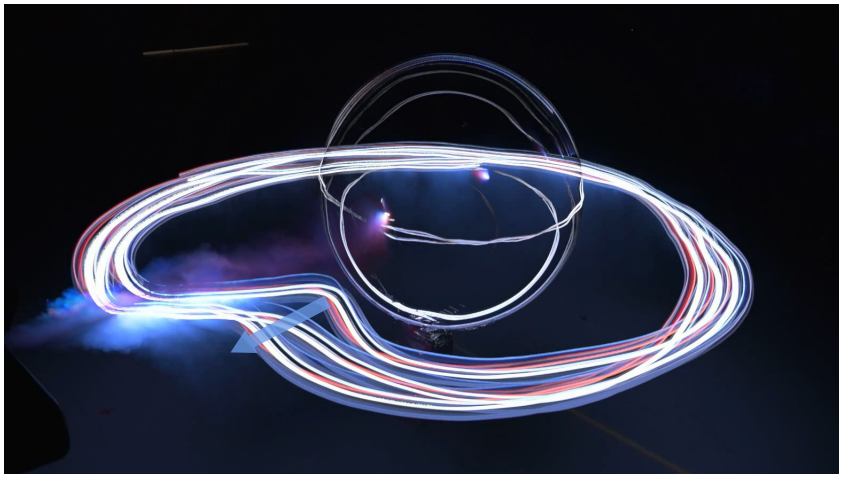


Figure 5.33: Long exposure trajectory of the nano quadrotor, maintain the safety under wind turbulence. The blue arrow indicates the robot recovers safety over time.

controller results in large *Avg.Dist.*, *Avg.sDist.* and *S.sDist.Var.*, as depicted in Fig. 5.30 (bottom right). Offline NODE-HO-aCBFs learn offline and thus adapt the residual without delay, resulting in minimal $h_{\min}$. Online NODE-HO-aCBFs have a learning delay, resulting in a slightly larger $h_{\min}$ and *Avg.Dist.* than the offline method, but as it learns online, it adapts better and settles stabler with incoming data and results in a smaller *Avg.sDist.* and *S.sDist.Var.*.

5.5.6 Physical Experiments

Physical Experiment Setup

The Crazyflie 2.1 nano quadrotor is used as the physical platform to verify the efficacy of our hybrid controller. The open-source library Crazyswarm [221] is used as an interface to send position-velocity-acceleration controls. The nano quadrotor weighs 38g with a 300mAh battery. A tower fan is placed horizontally at a height of 0.7m and 1.16m in the positive direction of the x-axis from the origin. Its horizontal airflow produces an effective turbulent region of approximately 0.55m wide, which travels 18km/h (its location and direction are depicted in Fig. 5.32). The quadrotor is required to keep a 1m distance from the origin. A laptop with Intel i7 is used as the base station, and communication with the Crazyflie is established using Crazyradio PA. We obtain 100Hz position data from Vicon motion capture, and estimate velocity from filtering. Control inputs are computed and sent to the quadrotor at 100Hz. We increase the number of neurons in each layer to 64 in our NODE-HO-aCBF controller to capture the wind turbulence. We experiment with both online and offline modes of our NODE-HO-aCBF controller. For the offline mode, we trained on 350s state-control data collected by executing the HOCBF baseline controller. A representative physical experiment using our online NODE-HO-aCBF controller is shown in Fig. 5.33. Note that safety is improved against the wind on the fly. For the HO-aCBF baseline controller, we used the state-informed matrix $Y(\mathbf{x}) = [\mathbf{0}_{3\times3}; r_1, 0, 0; 0, r_2, 0; 0, 0, r_3]$, namely HO-aCBF State $Y(\mathbf{x})$, and a constant matrix $Y(\mathbf{x}) = [\mathbf{0}_{3\times3}; \mathbf{I}_{3\times3}]$, namely HO-aCBF Constant $Y(\mathbf{x})$.

Physical Experiment Results

We evaluate the efficacy of our controller both qualitatively and quantitatively. In Fig. 5.32, the first 60s of a representative trajectory for each controller is depicted as the blue curve; the gray area is the forbidden region; the red circle is the reference trajectory; and the red arrows indicate the wind location and direction. Notice that all the baseline methods (Fig. 5.32(b)-(d)) cannot handle the wind turbulence, and converged to an unsafe trajectory. Our online controller is able to improve safety after a few experiences with wind turbulence, while our offline controller, having learned the residuals, is able to significantly improve safety at the beginning and maintain it throughout execution.

Table 5.6 and Fig. 5.34 show the quantitative results in the physical experiments. For the metrics computed on settled trajectory, i.e., *Avg.sDist.* and *S.sDist.Var.*, we exclude the first 60s of the trajectory. The learning model takes longer to converge due to the sparsity of turbulence data. To highlight the adaptation results, we evaluate the trajectory segments where the wind causes significant effects, that is, from $[0, \pi/4]$, and $[\pi, 5\pi/4]$ radian on the x-y plane. Our offline approach results in fewer safety violations, as indicated by a smaller h_{neg}, adapts better to turbulence, according to smaller *Avg.Dist.* and *Avg.sDist.*, and settled on a more stable trajectory, according to a smaller *S.sDist.Var.*, compared to HOCBF and HO-aCBF baseline controllers. Note that our offline approach results in smaller *Avg.Dist.* and *Avg.sDist.* even compared to the HOCBF without wind turbulence, as model mismatches exist between a double integrator and a quadrotor model in the HOCBF controller, learning the residuals adapt to such model mismatch. Our online approach has a smaller h_{neg} than the HO-aCBF controller and is slightly larger than that of HOCBF, as the online approach requires sufficient data to learn the residual. As the online approach sees more data, it adapts turbulence better and settles on a more stable trajectory, as indicated by smaller *Avg.sDist.* and *S.sDist.Var.*, compared to HOCBF and HO-aCBF controllers.

Method	h_{min}	h_{neg}	Avg.Dist.	Avg.sDist.	S.sDist.Var.
HOCBFs - No Wind	0.03	0.0%	6.46e-2	6.39e-2	1.02e-4
HOCBFs	-0.34	31.1%	9.03e-2	8.30e-2	7.61e-3
HO-aCBFs position $Y(\mathbf{x})$	-0.23	47.2%	4.68e-2	4.76e-2	3.26e-3
HO-aCBFs const $Y(\mathbf{x})$	**-0.22**	45.4%	4.49e-2	4.53e-2	2.90e-3
Online NODE-HO-aCBFs	-0.55	35.3%	7.08e-2	4.29e-2	2.49e-3
Offline NODE-HO-aCBFs	-0.24	**15.9%**	**4.44e-2**	**4.17e-2**	**1.34e-3**

Table 5.6: h_{min}, h_{neg}, *Avg.Dist.*, *Avg.sDist.* and *S.sDist.Var.* metrics of the baseline controllers and our controllers in the physical experiments. The statistics are averaged across 10 trials.

5.5.7 Conclusion

In this work, we present a learning-enhanced high-order adaptive control barrier function that improves safety online under time-varying complex model perturbations. We use a hybrid structure that combines a nominal model with a neural network residual model, using neural ODEs. Due to its hybrid design, our controller is data-efficient and expressive,

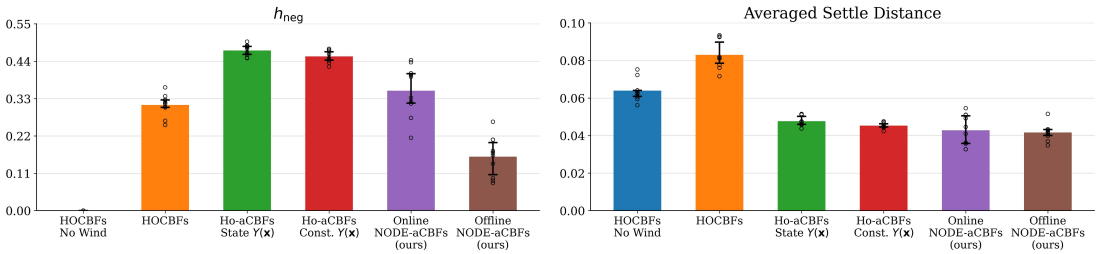


Figure 5.34: h_{neg} and *Avg.sDist.* of baselines and our controllers. The top of the bars denotes the median, while the ends of the error bars represent the 25^{th} and 75^{th} percentiles. The statistics are obtained from 10 trials.

making it suitable for adapting complex, time-varying residuals online. We demonstrate its efficacy across three types of residuals in simulations and a representative physical experiment, benchmarked with baseline HOCBFs [42] and HO-aCBFs [217, 218] (with different hyperparameters). A 38g nano quadrotor, deployed with our online NODE-HO-aCBF controller, can quickly improve its safety performance in 18km/h turbulent wind. Current large-scale swarm trajectory planning algorithms, such as [130, 222], lack an adaptive controller that improves safety and stability in close-proximity flight. For future work, we aim to provide probabilistic safety guarantees for the controller proposed in this work and to improve the safety and stability of close-proximity trajectory planning for large-scale swarms.

Chapter 6

Main Findings and Future Directions

6.1 Main Findings

In this thesis, we addressed the critical gap between theoretical control formulations and the practical deployment of multi-robot systems in real-world scenarios. The central objective was to develop decentralized control strategies capable of handling the inherent non-idealities of physical implementations, including limited sensing and communication capabilities, anisotropic sensor characteristics, and environmental uncertainties. Throughout the dissertation, we emphasized the importance of robust, local coordination mechanisms that function effectively without reliance on centralized infrastructure or perfect global knowledge.

We began by tackling the challenges posed by limited sensing ranges in exploration and monitoring tasks in Chapter 3. In Section 3.1, we introduced bio-inspired mechanisms based on virtual pheromones to drive the exploration of unknown, non-convex environments, ensuring continuous discovery through decay dynamics. Furthermore, we investigated the integration of human operators into the robotic loop, exploring a dual perspective: first, in Sections 3.2 and 3.3 as supervisors who guide the swarm through graphical and Mixed Reality interfaces by defining regions of interest; and second, as dynamic obstacles whose motion must be predicted and avoided to ensure safe coexistence during coverage operations in Section 3.4.

Subsequently, we addressed the specific constraints of anisotropic sensing in Chapter 4, focusing on robots equipped with sensors having a limited angular field of view. Initially, we developed control architectures that prioritize the maintenance of visual connectivity in Section 4.1, ensuring that neighboring robots remain within sight to preserve formation topology. In Section 4.2, we extended these concepts to scenarios characterized by the complete absence of communication, proposing probabilistic frameworks combining particle filters with barrier functions. This allowed agents to estimate the states of their neighbors based on intermittent visual data, balancing the trade-off between exploiting current knowledge for task execution and exploring to reduce uncertainty. A further ex-

tension was presented in Section 4.3, addressing the problem of Control Barrier Functions short-sightedness by combining them with Model Predictive Control. This integration enabled robots to plan trajectories that account for future states, thereby enhancing safety and performance. Finally, Section 4.4 presented a different approach to the limited field of view problem by clustering robots into groups to maximize the overall area coverage, demonstrating the effectiveness of cooperative strategies in overcoming individual sensor limitations.

Finally, we explored the integration of machine learning techniques to enhance the adaptability and performance of traditional control methods. We demonstrated how Convolutional Neural Networks can be leveraged to extract semantic information from the environment, such as road networks for surveillance in Section 5.1, and to learn effective coverage policies from expert demonstrations in complex settings in Section 5.2. Beyond perception, we utilized learning-based approaches to improve state estimation and handling dynamic uncertainties. This included the use of Neural Networks in Section 5.3 to approximate particle filters for computationally efficient flocking, Gaussian Processes for ergodic control of time-varying spatial phenomena in Section 5.4, and Neural Ordinary Differential Equations in Section 5.5 to compensate for unmodeled dynamics and external disturbances like wind.

Collectively, the methodologies presented in this thesis provide a comprehensive framework for deploying robust, autonomous multi-robot systems. By combining rigorous control theoretic tools such as Control Barrier Functions and Model Predictive Control with data-driven learning and probabilistic estimation, we have paved the way for robotic teams that are not only theoretically sound but also capable of operating safely and effectively in the complex, uncertain environments found in the real world.

6.2 Future Works

Several improvements and studies have yet to be done in order to fully take advantage of multi-robot systems in our every-day life and in all the possible real scenarios. A first important aspect that has not been addressed in this thesis is the heterogeneity of the robotic agents. In all the presented works, we assumed that all robots were identical in terms of sensing, communication and dynamics capabilities. For instance, the strategies proposed in Chapters 3 and 4 could be adapted for mixed teams of aerial and ground robots, where the former could provide high-level global coverage with limited resolution, while ground agents perform detailed local inspections. Investigating how the limited field of view constraints and communication-denied protocols differ between these platforms, and how they can cooperatively compensate for each other's blind spots, represents a promising direction for future research.

Another avenue for future work involves extending the proposed control strategies to fully three-dimensional environments. While the current thesis primarily focuses on planar or quasi-planar scenarios, many real-world applications require navigation in complex 3D spaces. Future works should aim to extend these methodologies to fully three-dimensional

environments, such as the interior of collapsed buildings or infrastructure inspection under bridges. Furthermore, the handling of occlusion in the limited-FOV scenarios analyzed in Chapter 4 becomes significantly more complex in 3D cluttered environments, requiring more sophisticated visibility-maintenance controllers.

In Section 3.2 and Section 3.3, we demonstrated the efficacy of Graphical User Interfaces (GUIs) and Mixed Reality (MR) for supervising robot swarms. A natural evolution of this work is the integration of Large Language Models (LLMs) to facilitate more intuitive, semantic interaction. Future research could investigate an interface where the operator issues high-level verbal commands (e.g., "Inspect the red building and avoid the wet terrain"), which are automatically parsed into the mathematical density functions (GMMs) or constraint sets used by our control framework. This would further lower the cognitive load on the operator and allow for more rapid deployment in emergency situations.

Finally, the work presented in Section 5.5 utilizes Neural Ordinary Differential Equations (NODEs) to learn residual dynamics for a single agent. A significant step forward would be extending this framework to multi-robot systems, for instance in cooperative transportation scenarios. Usually in such cases, the exact weight, center of mass, and inertia tensor of the load act as significant unmodeled disturbances on the robotic team. NODEs could be effectively employed to learn these coupled dynamics online, adapting the internal model of the formation as the transport begins. By capturing these unknown inertial parameters, the CBF constraints could be dynamically adjusted to ensure stability and prevent dangerous oscillations during transport.

Bibliography

[1] M. Catellani and L. Sabattini, "Distributed control of a limited angular field-of-view multi-robot system in communication-denied scenarios: A probabilistic approach," *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 739–746, 2023.

[2] F. Bertoncelli, V. Radhakrishnan, M. Catellani, G. Loianno, and L. Sabattini, "Directed graph topology preservation in multi-robot systems with limited field of view using control barrier functions," *IEEE Access*, vol. 12, pp. 9682–9690, 2024.

[3] L. Pan, M. Catellani, L. Sabattini, and N. Ayanian, "Robust trajectory generation and control for quadrotor motion planning with field-of-view control barrier certification," *IEEE Robotics and Automation Letters*, vol. 11, no. 2, pp. 1682–1689, 2025.

[4] M. Mantovani, M. Catellani, and L. Sabattini, "Distributed multi-robot ergodic coverage control for estimating time-varying spatial processes," *IEEE Robotics and Automation Letters*, 2026.

[5] L. Pan, M. Catellani, L. Sabattini, and N. Ayanian, "Online learning-enhanced high order adaptive control barrier functions using neural odes," *submitted to IEEE Robotics and Automation Letters*, 2026.

[6] M. Catellani, F. Pratissoli, F. Bertoncelli, and L. Sabattini, "Coverage control for exploration of unknown non-convex environments with limited range multi-robot systems," in *International Symposium on Distributed Autonomous Robotic Systems*. Springer, 2022, pp. 550–562.

[7] M. Catellani, E. Mazzocco, F. Bertoncelli, and L. Sabattini, "Distributed control for human-swarm interaction in non-convex environments using gaussian mixture models," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 3314–3319, 2023.

[8] M. Catellani, F. Nironi, and L. Sabattini, "A mixed reality interface for human-swarm interaction," in *European Robotics Forum*. Springer, 2024, pp. 112–117.

[9] M. Catellani, M. Belal, and L. Sabattini, "Dual-layer control architecture for cooperative environmental monitoring in multi-robot systems," in *International Symposium on Distributed Autonomous Robotic Systems*. Springer, 2024, pp. 15–27.

[10] M. Catellani, A. Alboni, and L. Sabattini, "Distributed multi-robot control for streets surveillance from aerial images with neural networks," *IFAC-PapersOnLine*, vol. 59, no. 18, pp. 175–180, 2025.

[11] M. Catellani and L. Sabattini, "Uncertainty-aware multi-robot flocking via learned state estimation and control barrier functions," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025.

[12] M. Catellani, M. Gabbi, and L. Sabattini, "Hmpcc: Human-aware model predictive coverage control," in *IEEE International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, 2025.

[13] M. Catellani, M. Mantovani, M. Montanari, and L. Sabattini, "Decentralized learning-based coverage control for multi-robot systems with obstacle awareness: A cnn-driven approach," *submitted to IFAC-PapersOnLine*, 2026.

[14] D. Nath, *Smart Farming: Automation and Robotics in Agriculture*, 09 2023.

[15] C. Ju, J. Kim, J. Seol, and H. I. Son, "A review on multirobot systems in agriculture," *Computers and Electronics in Agriculture*, vol. 202, p. 107336, 2022.

[16] K. Bazargani and T. Deemyad, "Automation's impact on agriculture: opportunities, challenges, and economic effects," *Robotics*, vol. 13, no. 2, p. 33, 2024.

[17] J. Koetsier, "Starship's 2,700 robots have made 9m deliveries. now they're coming to your city," October 2025, Forbes [Online; posted 15-October-2025].

[18] S. AZIMI, "Last-mile delivery using drones in the healthcare supply chain management."

[19] P. Corke, R. Peterson, and D. Rus, "Networked robots: Flying robot navigation using a sensor net," in *Robotics Research. The Eleventh International Symposium: With 303 Figures*. Springer, 2005, pp. 234–243.

[20] "Robotics: Latest trends & growth areas in the post pandemic world," https://analyticsindiamag.com/robotics-latest-trends-growth-areas-in-the-post-pandemic-world/.

[21] "Robotics outlook 2030: How intelligence and mobility will shape the future," https://www.bcg.com/publications/2021/how-intelligence-and-mobility-will-shape-the-future-of-the-robotics-industry.

[22] B. Siciliano, O. Khatib, and T. Kröger, *Springer handbook of robotics*. Springer, 2008, vol. 200.

[23] J. P. Queralta, J. Taipalmaa, B. C. Pullinen, V. K. Sarker, T. N. Gia, H. Tenhunen, M. Gabbouj, J. Raitoharju, and T. Westerlund, "Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision," *Ieee Access*, vol. 8, pp. 191 617–191 643, 2020.

[24] I. Martinez-Alpiste, G. Golcarenarenji, Q. Wang, and J. M. Alcaraz-Calero, "Search and rescue operation using uavs: A case study," *Expert Systems with Applications*, vol. 178, p. 114937, 2021.

[25] S. Li, Y. Guo, and B. Bingham, "Multi-robot cooperative control for monitoring and tracking dynamic plumes," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 67–73.

[26] A. I. De Castro, Y. Shi, J. M. Maja, and J. M. Peña, "Uavs for vegetation monitoring: Overview and recent scientific contributions," *Remote Sensing*, vol. 13, no. 11, p. 2139, 2021.

[27] H. Qin, Z. Meng, W. Meng, X. Chen, H. Sun, F. Lin, and M. H. Ang, "Autonomous exploration and mapping system using heterogeneous uavs and ugvs in gps-denied environments," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1339–1350, 2019.

[28] J. Kim, S. Kim, C. Ju, and H. I. Son, "Unmanned aerial vehicles in agriculture: A review of perspective of platform, control, and applications," *Ieee Access*, vol. 7, pp. 105 100–105 115, 2019.

[29] A. Gupta, T. Afrin, E. Scully, and N. Yodo, "Advances of uavs toward future transportation: The state-of-the-art, challenges, and opportunities," *Future transportation*, vol. 1, no. 2, pp. 326–350, 2021.

[30] E. Tuci, M. H. Alkilabi, and O. Akanyeti, "Cooperative object transport in multi-robot systems: A review of the state-of-the-art," *Frontiers in Robotics and AI*, vol. 5, p. 59, 2018.

[31] J. Gielis, A. Shankar, and A. Prorok, "A critical review of communications in multi-robot systems," *Current robotics reports*, vol. 3, no. 4, pp. 213–225, 2022.

[32] E. Klavins, "Communication complexity of multi-robot systems," in *Algorithmic Foundations of Robotics V*. Springer, 2004, pp. 275–291.

[33] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[34] M. Schwager, J. McLurkin, and D. Rus, "Distributed coverage control with sensory feedback for networked robots." in *robotics: science and systems*, 2006, pp. 49–56.

[35] Q. Du, M. Emelianenko, and L. Ju, "Convergence of the lloyd algorithm for computing centroidal voronoi tessellations," *SIAM journal on numerical analysis*, vol. 44, no. 1, pp. 102–119, 2006.

[36] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[37] F. Pratissoli, B. Capelli, and L. Sabattini, "On coverage control for limited range multi-robot systems," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 9957–9963.

[38] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *53rd IEEE Conf. on Decis. and Control*. IEEE, 2014, pp. 6271–6278.

[39] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *18th Eur. Control Conf. (ECC)*. IEEE, 2019, pp. 3420–3431.

[40] K. Yano, *The theory of Lie derivatives and its applications*. Courier Dover Publications, 2020.

[41] A. D. Ames, K. Galloway, and J. W. Grizzle, "Control lyapunov functions and hybrid zero dynamics," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. IEEE, 2012, pp. 6837–6842.

[42] W. Xiao and C. Belta, "High-order control barrier functions," *IEEE Trans. on Autom. Control*, vol. 67, no. 7, pp. 3655–3662, 2021.

[43] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles*. Springer, 1986, pp. 396–404.

[44] M. Kegeleirs, D. Garzón Ramos, and M. Birattari, "Random walk exploration for swarm mapping," in *Towards Autonomous Robotic Systems*, K. Althoefer, J. Konstantinova, and K. Zhang, Eds. Cham: Springer International Publishing, 2019, pp. 211–222.

[45] K. Senthilkumar and K. K. Bharadwaj, "Multi-robot exploration and terrain coverage in an unknown environment," *Robotics and Autonomous Systems*, vol. 60, no. 1, pp. 123–132, 2012.

[46] J. De Hoog, S. Cameron, and A. Visser, "Dynamic team hierarchies in communication-limited multi-robot exploration," in *2010 IEEE Safety Security and Rescue Robotics*. IEEE, 2010, pp. 1–7.

[47] M. A. Batalin and G. S. Sukhatme, "Coverage, exploration and deployment by a mobile robot and communication network," *Telecommunication Systems*, vol. 26, no. 2, pp. 181–196, 2004.

[48] ——, "The analysis of an efficient algorithm for robot coverage and exploration based on sensor network deployment," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 3478–3485.

[49] C. Stachniss, "Coordinated multi-robot exploration," in *Robotic mapping and exploration*. Springer, 2009, pp. 43–71.

[50] A. Franchi, L. Freda, G. Oriolo, and M. Vendittelli, "The sensor-based random graph method for cooperative robot exploration," *IEEE/ASME transactions on mechatronics*, vol. 14, no. 2, pp. 163–175, 2009.

[51] O. Arslan and D. E. Koditschek, "Sensor-based reactive navigation in unknown convex sphere worlds," *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 196–223, 2019.

[52] J. Hu, H. Niu, J. Carrasco, B. Lennox, and F. Arvin, "Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14 413–14 423, 2020.

[53] H. Ryu, "Graph search-based exploration method using a frontier-graph structure for mobile robots," *Sensors*, vol. 20, no. 21, p. 6270, 2020.

[54] J. Kim, "Cooperative exploration and networking while preserving collision avoidance," *IEEE transactions on cybernetics*, vol. 47, no. 12, pp. 4038–4048, 2016.

[55] K. Masaba and A. Quattrini Li, "Gvgexp: Communication-constrained multi-robot exploration system based on generalized voronoi graphs," in *2021 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, 2021.

[56] M. Corah, C. O'Meadhra, K. Goel, and N. Michael, "Communication-efficient planning and mapping for multi-robot exploration in large environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1715–1721, 2019.

[57] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science robotics*, vol. 7, no. 66, p. eabm6074, 2022.

[58] G. Francesca, M. Brambilla, A. Brutschy, V. Trianni, and M. Birattari, "Automode: A novel approach to the automatic design of control software for robot swarms," *Swarm Intell*, vol. 8, pp. 1–24, 06 2014.

[59] C. Yan, B. Shao, H. Zhao, R. Ning, Y. Zhang, and F. Xu, "3d room layout estimation from a single rgb image," *IEEE Transactions on Multimedia*, vol. 22, no. 11, pp. 3014–3024, 2020.

[60] A. Hussein and H. Abbass, "Mixed initiative systems for human-swarm interaction: Opportunities and challenges," in *2018 2nd Annual Systems Modelling Conference (SMC)*, 2018.

[61] C. C. Cheah, S. P. Hou, and J. J. E. Slotine, "Region-based shape control for a swarm of robots," *Automatica*, vol. 45, no. 10, 2009.

[62] W.-T. Li and Y.-C. Liu, "Human-swarm collaboration with coverage control under nonidentical and limited sensory ranges," *Journal of the Franklin Institute*, vol. 356, no. 16, pp. 9122–9151, 2019.

[63] Y. Diaz-Mercado, S. G. Lee, and M. Egerstedt, "Distributed dynamic density coverage for human-swarm interactions," in *2015 American control conference (ACC)*, 2015, pp. 353–358.

[64] S. Kotz, N. Balakrishnan, and N. L. Johnson, *Continuous multivariate distributions, Volume 1: Models and applications*. John Wiley & Sons, 2004, vol. 1.

[65] G. J. McLachlan, S. X. Lee, and S. I. Rathnayake, "Finite mixture models," *Annual review of statistics and its application*, vol. 6, pp. 355–378, 2019.

[66] F. Bertoncelli, M. Belal, D. Albani, F. Pratissoli, and L. Sabattini, "On limited-range coverage control for large-scale teams of aerial drones: Deployment and study," in *International Symposium on Distributed Autonomous Robotic Systems*. Springer, 2022, pp. 333–346.

[67] G. Schwarz, "Estimating the dimension of a model," *The annals of statistics*, pp. 461–464, 1978.

[68] F. Ferraguti, C. T. Landi, A. Singletary, H.-C. Lin, A. Ames, C. Secchi, and M. Bonfe, "Safety and efficiency in robotics: The control barrier functions approach," *IEEE Robotics & Automation Magazine*, vol. 29, no. 3, pp. 139–151, 2022.

[69] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras, "Human motion trajectory prediction: A survey," *The International Journal of Robotics Research*, vol. 39, no. 8, pp. 895–935, 2020.

[70] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[71] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE international conference on robotics and automation*. Ieee, 2008, pp. 1928–1935.

[72] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research: The 14th International Symposium ISRR*. Springer, 2011, pp. 3–19.

[73] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 961–971.

[74] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2255–2264.

[75] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*. Springer, 2020, pp. 683–700.

[76] T. Maeda and N. Ukita, "Fast inference and update of probabilistic density estimation on trajectory prediction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 9795–9805.

[77] A. Breitenmoser, M. Schwager, J.-C. Metzger, R. Siegwart, and D. Rus, "Voronoi coverage of non-convex environments with a group of networked robots," in *2010 IEEE international conference on robotics and automation*. IEEE, 2010, pp. 4982–4989.

[78] I. Kamon, E. Rimon, and E. Rivlin, "Tangentbug: A range-sensor-based navigation algorithm," *The International Journal of Robotics Research*, vol. 17, no. 9, pp. 934–953, 1998.

[79] S. Bhattacharya, N. Michael, and V. Kumar, "Distributed coverage and exploration in unknown non-convex environments," in *Distributed autonomous robotic systems*. Springer, 2013, pp. 61–75.

[80] C. Franco, D. M. Stipanović, G. López-Nicolás, C. Sagüés, and S. Llorente, "Persistent coverage control for a team of agents with collision avoidance," *European Journal of Control*, vol. 22, pp. 30–45, 2015.

[81] Q. Liu, Z. Zhang, N. K. Le, J. Qin, F. Liu, and S. Hirche, "Distributed coverage control of constrained constant-speed unicycle multi-agent systems," *IEEE Transactions on Automation Science and Engineering*, 2024.

[82] A. Carron and M. N. Zeilinger, "Model predictive coverage control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6107–6112, 2020.

[83] R. Rickenbach, J. Köhler, A. Scampicchio, M. N. Zeilinger, and A. Carron, "Active learning-based model predictive coverage control," *IEEE Transactions on Automatic Control*, vol. 69, no. 9, pp. 5931–5946, 2024.

[84] A. Breitenmoser and A. Martinoli, "On combining multi-robot coverage and reciprocal collision avoidance," in *Distributed Autonomous Robotic Systems: The 12th International Symposium*. Springer, 2016, pp. 49–64.

[85] D. Claes and K. Tuyls, "Multi robot collision avoidance in a shared workspace," *Autonomous Robots*, vol. 42, no. 8, pp. 1749–1770, 2018.

[86] Z. Talebpour and A. Martinoli, "Multi-robot coordination in dynamic environments shared with humans," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 4593–4600.

[87] I. Bae, Y.-J. Park, and H.-G. Jeon, "Singulartrajectory: Universal trajectory predictor using diffusion model," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 17 890–17 901.

[88] N. E. Du Toit and J. W. Burdick, "Probabilistic collision checking with chance constraints," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 809–815, 2011.

[89] L. Sabattini, C. Secchi, N. Chopra, and A. Gasparri, "Distributed control of multirobot systems with global connectivity maintenance," *IEEE Transactions on Robotics*, vol. 29, no. 5, pp. 1326–1332, 2013.

[90] M. Fiacchini and I.-C. Morărescu, "Convex conditions on decentralized control for graph topology preservation," *IEEE Transactions on Automatic Control*, vol. 59, no. 6, pp. 1640–1645, 2013.

[91] K. Khateri, M. Pourgholi, M. Montazeri, and L. Sabattini, "A comparison between decentralized local and global methods for connectivity maintenance of multi-robot networks," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 633–640, 2019.

[92] S. Yi, W. Luo, and K. Sycara, "Distributed topology correction for flexible connectivity maintenance in multi-robot systems," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 8874–8880.

[93] P. Ong, B. Capelli, L. Sabattini, and J. Cortés, "Network connectivity maintenance via nonsmooth control barrier functions," in *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 2021, pp. 4786–4791.

[94] H.-A. Hung, H.-H. Hsu, and T.-H. Cheng, "Image-based multi-uav tracking system in a cluttered environment," *IEEE Transactions on Control of Network Systems*, vol. 9, no. 4, pp. 1863–1874, 2022.

[95] I. Salehi, G. Rotithor, R. Saltus, and A. P. Dani, "Constrained image-based visual servoing using barrier functions," in *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2021, pp. 14 254–14 260.

[96] G. Loianno, C. Brunner, G. McGrath, and V. Kumar, "Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and imu," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 404–411, 2016.

[97] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "Osqp: An operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.

[98] J. Chen, P. Han, Y. Zhang, T. You, and P. Zheng, "Scheduling energy consumption-constrained workflows in heterogeneous multi-processor embedded systems," *Journal of Systems Architecture*, vol. 142, p. 102938, 2023.

[99] Z. Wang, Y. Hirata, and K. Kosuge, "Control a rigid caging formation for cooperative object transportation by multiple mobile robots," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 2. IEEE, 2004, pp. 1580–1585.

[100] C. Bhowmick, L. Behera, A. Shukla, and H. Karki, "Flocking control of multi-agent system with leader-follower architecture using consensus based estimated flocking center," in *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2016, pp. 166–171.

[101] J. Chen, T. Li, Y. Zhang, T. You, Y. Lu, P. Tiwari, and N. Kumar, "Global-and-local attention-based reinforcement learning for cooperative behaviour control of multiple uavs," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 3, pp. 4194–4206, 2023.

[102] D. Panagou, D. M. Stipanović, and P. G. Voulgaris, "Distributed dynamic coverage and avoidance control under anisotropic sensing," *IEEE Transactions on Control of Network Systems*, vol. 4, no. 4, pp. 850–862, 2016.

[103] M. B. Silic, Z. Song, and K. Mohseni, "Anisotropic flocking control of distributed multi-agent systems using fluid abstraction," in *2018 AIAA Information Systems-AIAA Infotech@ Aerospace*, 2018, p. 2262.

[104] A. Wasik, P. U. Lima, and A. Martinoli, "A robust localization system for multi-robot formations based on an extension of a gaussian mixture probability hypothesis density filter," *Autonomous Robots*, vol. 44, no. 3, pp. 395–414, 2020.

[105] W. Luo, W. Sun, and A. Kapoor, "Multi-robot collision avoidance under uncertainty with probabilistic safety barrier certificates," *Advances in Neural Information Processing Systems*, vol. 33, pp. 372–383, 2020.

[106] J. Horyna, V. Walter, and M. Saska, "Uvdar-com: Uv-based relative localization of uavs with integrated optical communication," in *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2022, pp. 1302–1308.

[107] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman filter: Particle filters for tracking applications*. Artech house, 2003.

[108] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, 2002.

[109] K. Pearson, "X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 50, no. 302, pp. 157–175, 1900.

[110] O. Kanoun, F. Lamiraux, and P.-B. Wieber, "Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task," *IEEE Transactions on Robotics*, vol. 27, no. 4, 2011.

[111] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "Rotors—a modular gazebo mav simulator framework," in *Robot Operating System (ROS) The Complete Reference (Volume 1)*. Springer, 2016, pp. 595–625.

[112] D. S. Drew, "Multi-agent systems for search and rescue applications," *Current Robotics Reports*, vol. 2, pp. 189–200, 2021.

[113] J. Liu, P. Li, Y. Wu, G. S. Sukhatme, V. Kumar, and L. Zhou, "Multi-robot target tracking with sensing and communication danger zones," in *Int. Symp. Distrib. Auton. Robot. Syst. (DARS)*. Springer, 2024, pp. 600–615.

[114] G. Li and G. Loianno, "Nonlinear model predictive control for cooperative transportation and manipulation of cable suspended payloads with multiple quadrotors," in *2023 IEEE/RSJ Int. Conf. on Intell. Robots and Syst. (IROS)*. IEEE, 2023, pp. 5034–5041.

[115] H. Huang, Y. Xu, Y. Chen, and S.-K. Yeung, "360vot: A new benchmark dataset for omnidirectional visual object tracking," in *Proceedings of the IEEE/CVF Int. Conf. on Comput. Vis.*, 2023, pp. 20 566–20 576.

[116] W. Xiao, C. A. Belta, and C. G. Cassandras, "High order control lyapunov-barrier functions for temporal logic specifications," in *Amer. Control Conf. (ACC)*. IEEE, 2021, pp. 4886–4891.

[117] H. Abdi, G. Raja, and R. Ghabcheloo, "Safe control using vision-based control barrier function (v-cbf)," in *IEEE Int. Conf. on Robot. and Automat. (ICRA)*. IEEE, 2023, pp. 782–788.

[118] X. Xu, J. W. Grizzle, P. Tabuada, and A. D. Ames, "Correctness guarantees for the composition of lane keeping and adaptive cruise control," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 3, 2017.

[119] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Trans. Robot.*, vol. 33, no. 3, pp. 661–674, 2017.

[120] B. Trimarchi, F. Schiano, and R. Tron, "A control barrier function candidate for quadrotors with limited field of view," in *Amer. Control Conf. (ACC)*. IEEE, 2025, pp. 251–258.

[121] M. Zhou, M. Shaikh, V. Chaubey, P. Haggerty, S. Koga, D. Panagou, and N. Atanasov, "Control strategies for pursuit-evasion under occlusion using visibility and safety barrier functions," in *Proc. Int. Conf. Robot. Automat.* IEEE, 2025, pp. 12 863–12 869.

[122] M. Santilli, P. Mukherjee, R. K. Williams, and A. Gasparri, "Multirobot field of view control with adaptive decentralization," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2131–2150, 2022.

[123] J. Zeng, B. Zhang, and K. Sreenath, "Safety-critical model predictive control with discrete-time control barrier function," in *Amer. Control Conf. (ACC)*. IEEE, 2021, pp. 3882–3889.

[124] L. Sforni, G. Notarstefano, and A. D. Ames, "Receding horizon cbf-based multi-layer controllers for safe trajectory generation," in *Amer. Control Conf. (ACC)*. IEEE, 2024, pp. 4765–4770.

[125] A. Dickson, C. G. Cassandras, and R. Tron, "Spline trajectory tracking and obstacle avoidance for mobile agents via convex optimization," in *IEEE 63rd Conference on Decision and Control (CDC)*. IEEE, 2024, pp. 6572–6577.

[126] K. I. Joy, "Bernstein polynomials," *On-Line Geometric Modeling Notes*, vol. 13, no. 5, 2000.

[127] L. Pan, K. Hsu, and N. Ayanian, "Hierarchical large scale multirobot path (re) planning," in *2024 IEEE/RSJ Int. Conf. on Intell. Robots and Syst. (IROS)*. IEEE, 2024, pp. 5319–5326.

[128] T. Mercy, R. Van Parys, and G. Pipeleers, "Spline-based motion planning for autonomous guided vehicles in a dynamic environment," *IEEE Trans. Control Syst. Technol.*, vol. 26, no. 6, pp. 2182–2189, 2017.

[129] W. Hönig, J. A. Preiss, T. S. Kumar, G. S. Sukhatme, and N. Ayanian, "Trajectory planning for quadrotor swarms," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 856–869, 2018.

[130] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Online trajectory generation with distributed model predictive control for multi-robot motion planning," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, 2020.

[131] A. Maoudj and A. L. Christensen, "Improved decentralized cooperative multi-agent path finding for robots with limited communication," *Swarm Intelligence*, vol. 18, no. 2, pp. 167–185, 2024.

[132] W. Xiao, C. G. Cassandras, and C. Belta, *Safe autonomy with control barrier functions: theory and applications.* Springer, 2023.

[133] R. Ge, M. Lee, V. Radhakrishnan, Y. Zhou, G. Li, and G. Loianno, "Vision-based relative detection and tracking for teams of micro aerial vehicles," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE, 2022, pp. 380–387.

[134] A. Kiran, N. Harutyunyan, N. Ayanian, and K. Breuer, "Downwash dynamics: Impact of separation on forces, moments, and velocities for dense quadrotor flight," in *AIAA AVIATION FORUM AND ASCEND 2024*, 2024, p. 4145.

[135] Z. Zhang, Y. Han, Y. Zhou, and M. Dai, "A novel absolute localization estimation of a target with monocular vision," *Optik*, vol. 124, no. 12, pp. 1218–1223, 2013.

[136] B.-F. Wu, W.-C. Lu, and C.-L. Jen, "Monocular vision-based robot localization and target tracking," *Journal of Robotics*, vol. 2011, 2011.

[137] Y. Ming, X. Meng, C. Fan, and H. Yu, "Deep learning for monocular depth estimation: A review," *Neurocomputing*, vol. 438, pp. 14–33, 2021.

[138] L. Zhou, V. D. Sharma, Q. Li, A. Prorok, A. Ribeiro, P. Tokekar, and V. Kumar, "Graph neural networks for decentralized multi-robot target tracking," in *2022 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR).* IEEE, 2022, pp. 195–202.

[139] J. Alonso-Mora, E. Montijano, T. Nägeli, O. Hilliges, M. Schwager, and D. Rus, "Distributed multi-robot formation control in dynamic environments," *Autonomous Robots*, vol. 43, pp. 1079–1100, 2019.

[140] J. Alonso-Mora, S. Baker, and D. Rus, "Multi-robot formation control and object transport in dynamic environments via constrained optimization," *The International Journal of Robotics Research*, vol. 36, no. 9, pp. 1000–1021, 2017.

[141] L. Sabattini, C. Secchi, and C. Fantuzzi, "Arbitrarily shaped formations of mobile robots: artificial potential fields and coordinate transformation," *Autonomous Robots*, vol. 30, pp. 385–397, 2011.

[142] C. Bai, P. Yan, W. Pan, and J. Guo, "Learning-based multi-robot formation control with obstacle avoidance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 11 811–11 822, 2021.

[143] L. Martins, C. Cardeira, and P. Oliveira, "Inner-outer feedback linearization for quadrotor control: two-step design and validation," *Nonlinear Dynamics*, vol. 110, no. 1, pp. 479–495, 2022.

[144] Y. Bai, K. Asami, M. Svinin, and E. Magid, "Cooperative multi-robot control for monitoring an expanding flood area," in *2020 17th International Conference on Ubiquitous Robots (UR)*. IEEE, 2020, pp. 500–505.

[145] S. Thoduka, J. Gall, and P. G. Plöger, "Using visual anomaly detection for task execution monitoring," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 4604–4610.

[146] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer, 2015, pp. 234–241.

[147] D. Vallejo, P. Remagnino, D. N. Monekosso, L. Jiménez, and C. González, "A multi-agent architecture for multi-robot surveillance," in *Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems: First International Conference, ICCCI 2009, Wrocław, Poland, October 5-7, 2009. Proceedings 1*. Springer, 2009, pp. 266–278.

[148] P. Yao, Q. Zhu, and R. Zhao, "Gaussian mixture model and self-organizing map neural-network-based coverage for target search in curve-shape area," *IEEE Transactions on Cybernetics*, vol. 52, no. 5, pp. 3971–3983, 2020.

[149] L. Lin and M. A. Goodrich, "Hierarchical heuristic search using a gaussian mixture model for uav coverage planning," *IEEE transactions on cybernetics*, vol. 44, no. 12, 2014.

[150] I. Demir, K. Koperski, D. Lindenbaum, G. Pang, J. Huang, S. Basu, F. Hughes, D. Tuia, and R. Raskar, "Deepglobe 2018: A challenge to parse the earth through satellite images," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[151] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[152] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. Jorge Cardoso, "Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations," in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: Third International Workshop, DLMIA 2017, and 7th International Workshop, ML-CDS 2017, Held in Conjunction with MICCAI 2017, Québec City, QC, Canada, September 14, Proceedings 3*. Springer, 2017, pp. 240–248.

[153] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[154] B. Karlsson, "Renderdoc," *URL https://renderdoc. org*, 2019.

[155] C. He, Z. Feng, and Z. Ren, "Distributed algorithm for voronoi partition of wireless sensor networks with a limited sensing range," *Sensors*, vol. 18, no. 2, p. 446, 2018.

[156] W.-T. Li and Y.-C. Liu, "Dynamic coverage control for mobile robot network with limited and nonidentical sensory ranges," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 775–780.

[157] K. Laventall and J. Cortés, "Coverage control by multi-robot networks with limited-range anisotropic sensory," *International Journal of Control*, vol. 82, no. 6, pp. 1113–1121, 2009.

[158] W. Gosrich, S. Mayya, R. Li, J. Paulos, M. Yim, A. Ribeiro, and V. Kumar, "Coverage control in multi-robot systems via graph neural networks," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 8787–8793.

[159] M. Zhu, D. Simon, N. Rajpurohit, S. J. Kalathia, and W. Wu, "Reinforcement learning for multi-robot field coverage based on local observation," in *2020 IEEE 15th International Conference of System of Systems Engineering (SoSE)*. IEEE, 2020, pp. 35–40.

[160] F. Foroughi, Z. Chen, and J. Wang, "A cnn-based system for mobile robot navigation in indoor environments via visual localization with a small dataset," *World Electric Vehicle Journal*, vol. 12, no. 3, p. 134, 2021.

[161] R. Azzam, I. Boiko, and Y. Zweiri, "Swarm cooperative navigation using centralized training and decentralized execution," *Drones*, vol. 7, no. 3, p. 193, 2023.

[162] M. Catellani, M. Mantovani, F. Pratissoli, and L. Sabattini, "Safe trajectories from local information for coverage control in non-convex environments," Zenodo, doi: 10.5281/zenodo.13763588, 2024.

[163] Y. LeCun, Y. Bengio, *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[164] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[165] W. Giernacki, M. Skwierczyński, W. Witwicki, P. Wroński, and P. Kozierski, "Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering," in *2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR)*. IEEE, 2017, pp. 37–42.

[166] K. D. Do, "Flocking for multiple elliptical agents with limited communication ranges," *IEEE transactions on robotics*, vol. 27, no. 5, pp. 931–942, 2011.

[167] X. Shao, J. Zhang, and W. Zhang, "Distributed cooperative surrounding control for mobile robots with uncertainties and aperiodic sampling," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 18 951–18 961, 2022.

[168] K. Hausman, J. Müller, A. Hariharan, N. Ayanian, and G. S. Sukhatme, "Cooperative multi-robot control for target tracking with onboard sensing," *The International Journal of Robotics Research*, vol. 34, no. 13, pp. 1660–1677, 2015.

[169] V. Brunacci, A. De Angelis, G. Costante, and P. Carbone, "Development and analysis of a uwb relative localization system," *IEEE Transactions on Instrumentation and Measurement*, vol. 72, 2023.

[170] V. Walter, N. Staub, A. Franchi, and M. Saska, "Uvdar system for visual relative localization with application to leader–follower formations of multirotor uavs," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2637–2644, 2019.

[171] A. Okubo, "Dynamical aspects of animal grouping: swarms, schools, flocks, and herds," *Advances in biophysics*, vol. 22, pp. 1–94, 1986.

[172] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 1987, pp. 25–34.

[173] Z. A. Ali, E. H. Alkhammash, and R. Hasan, "State-of-the-art flocking strategies for the collective motion of multi-robots," *Machines*, vol. 12, no. 10, p. 739, 2024.

[174] K. Saulnier, D. Saldana, A. Prorok, G. J. Pappas, and V. Kumar, "Resilient flocking for mobile robot teams," *IEEE Robotics and Automation letters*, vol. 2, no. 2, pp. 1039–1046, 2017.

[175] L. A. V. Reyes and H. G. Tanner, "Flocking, formation control, and path following for a group of mobile robots," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 4, pp. 1268–1282, 2014.

[176] T. Hatanaka, N. Chopra, M. Fujita, and M. W. Spong, *Passivity-based control and estimation in networked robotics*. Springer, 2015.

[177] T. Ibuki, S. Wilson, J. Yamauchi, M. Fujita, and M. Egerstedt, "Optimization-based distributed flocking control for multiple rigid bodies," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1891–1898, 2020.

[178] P. Zhu, W. Dai, W. Yao, J. Ma, Z. Zeng, and H. Lu, "Multi-robot flocking control based on deep reinforcement learning," *IEEE Access*, vol. 8, pp. 150 397–150 406, 2020.

[179] E. J. Ávila-Martínez and J. G. Barajas-Ramírez, "Flocking motion in swarms with limited sensing radius and heterogeneous input constraints," *Journal of The Franklin Institute*, vol. 358, no. 4, 2021.

[180] J. Cortes, S. Martinez, and F. Bullo, "Spatially-distributed coverage optimization and control with limited-range interactions," *ESAIM: Control, Optimisation and Calculus of Variations*, vol. 11, no. 4, pp. 691–719, 2005.

[181] A. Carron and M. N. Zeilinger, "Model predictive coverage control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6107–6112, 2020, 21st IFAC World Congress.

[182] M. Catellani, M. Gabbi, and L. Sabattini, "HMPCC: human-aware model predictive coverage control," in *Proceedings of the IEEE International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, Singapore, dec. 2025.

[183] W. Gosrich, S. Mayya, R. Li, J. Paulos, M. Yim, A. Ribeiro, and V. Kumar, "Coverage control in multi-robot systems via graph neural networks," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE Press, 2022, p. 8787–8793.

[184] G. Mathew and I. Mezić, "Metrics for ergodicity and design of ergodic dynamics for multi-agent systems," *Physica D: Nonlinear Phenomena*, vol. 240, no. 4-5, pp. 432–442, 2011.

[185] L. M. Miller and T. D. Murphey, "Trajectory optimization for continuous ergodic exploration," in *2013 American Control Conference*. IEEE, 2013, pp. 4196–4201.

[186] M. Sun, A. Gaggar, P. Trautman, and T. Murphey, "Fast ergodic search with kernel functions," 2024.

[187] D. Dong, H. Berger, and I. Abraham, "Time optimal ergodic search," *arXiv preprint arXiv:2305.11643*, 2023.

[188] E. Ayvali, H. Salman, and H. Choset, "Ergodic coverage in constrained environments using stochastic trajectory optimization," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 5204–5210.

[189] C. Lerch, D. Dong, and I. Abraham, "Safety-critical ergodic exploration in cluttered environments via control barrier functions," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 10 205–10 211.

[190] S. Ivić, B. Crnković, and I. Mezić, "Ergodicity-based cooperative multiagent area coverage via a potential field," *IEEE Transactions on Cybernetics*, vol. 47, no. 8, pp. 1983–1993, 2017.

[191] S. Ivić, A. Sikirica, and B. Crnković, "Constrained multi-agent ergodic area surveying control based on finite element approximation of the potential field," *Engineering Applications of Artificial Intelligence*, vol. 116, p. 105441, 2022.

[192] M. Santos, U. Madhushani, A. Benevento, and N. E. Leonard, "Multi-robot learning and coverage of unknown spatial fields," in *2021 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, 2021, pp. 137–145.

[193] K.-C. Ma, L. Liu, and G. S. Sukhatme, "Informative planning and online learning with sparse gaussian processes," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 4292–4298.

[194] L. Wei, A. McDonald, and V. Srivastava, "Multi-robot gaussian process estimation and coverage: Deterministic sequencing algorithm and regret analysis," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 9080–9085.

[195] M. Mantovani, F. Pratissoli, and L. Sabattini, "Distributed coverage control for spatial processes estimation with noisy observations," *IEEE Robotics and Automation Letters*, vol. 9, no. 5, pp. 4431–4438, 2024.

[196] F. Pratissoli, M. Mantovani, A. Prorok, and L. Sabattini, "Distributed coverage control for time-varying spatial processes," *IEEE Transactions on Robotics*, vol. 41, pp. 1602–1617, 2025.

[197] I. Abraham and T. D. Murphey, "Decentralized ergodic control: distribution-driven sensing and exploration for multiagent systems," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2987–2994, 2018.

[198] E. Wittemyer, A. Rao, I. Abraham, and H. Choset, "Multi-agent ergodic exploration under smoke-based time-varying sensor visibility constraints," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 5452–5458.

[199] S. Ivić, A. Andrejčuk, and S. Družeta, "Autonomous control for multi-agent non-uniform spraying," *Applied Soft Computing*, vol. 80, pp. 742–760, 2019.

[200] S. Ivić, "Motion control for autonomous heterogeneous multiagent area search in uncertain conditions," *IEEE Transactions on Cybernetics*, vol. 52, no. 5, pp. 3123–3135, 2022.

[201] C. E. Rasmussen, *Gaussian Processes in Machine Learning*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 63–71.

[202] E. Schulz, M. Speekenbrink, and A. Krause, "A tutorial on gaussian process regression: Modelling, exploring, and exploiting functions," *Journal of Mathematical Psychology*, vol. 85, pp. 1–16, 2018.

[203] V. Gómez-Verdejo, E. Parrado-Hernández, and M. Martínez-Ramón, "Adaptive sparse gaussian process," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 11, pp. 16 383–16 395, 2024.

[204] I. I. Hussein and D. M. Stipanovic, "Effective coverage control for mobile sensor networks with guaranteed collision avoidance," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 4, pp. 642–657, 2007.

[205] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," *Advances in neural information processing systems*, vol. 31, 2018.

[206] T. Z. Jiahao, M. A. Hsieh, and E. Forgoston, "Knowledge-based learning of nonlinear dynamics and chaos," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 31, no. 11, 2021.

[207] T. Z. Jiahao, L. Pan, and M. A. Hsieh, "Learning to swarm with knowledge-based neural ordinary differential equations," in *2022 International conference on robotics and automation (ICRA)*. IEEE, 2022, pp. 6912–6918.

[208] M. O'Connell, G. Shi, X. Shi, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S.-J. Chung, "Neural-fly enables rapid learning for agile flight in strong winds," *Science Robotics*, vol. 7, no. 66, p. eabm6597, 2022.

[209] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on machine learning (ICML-11)*, 2011, pp. 465–472.

[210] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious model predictive control using gaussian process regression," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2736–2743, 2019.

[211] A. Taylor, A. Singletary, Y. Yue, and A. Ames, "Learning for safety-critical control with control barrier functions," in *Learning for dynamics and control*. PMLR, 2020, pp. 708–717.

[212] J. Choi, F. Castaneda, C. J. Tomlin, and K. Sreenath, "Reinforcement learning for safety-critical control under model uncertainty, using control lyapunov functions and control barrier functions," *arXiv preprint arXiv:2004.07584*, 2020.

[213] M. Jankovic, "Robust control barrier functions for constrained stabilization of nonlinear systems," *Automatica*, vol. 96, pp. 359–367, 2018.

[214] A. J. Taylor and A. D. Ames, "Adaptive safety with control barrier functions," in *2020 American Control Conference (ACC)*. IEEE, 2020, pp. 1399–1405.

[215] B. T. Lopez, J.-J. E. Slotine, and J. P. How, "Robust adaptive control barrier functions: An adaptive and data-driven approach to safety," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 1031–1036, 2020.

[216] M. Black and D. Panagou, "Safe control design for unknown nonlinear systems with koopman-based fixed-time identification," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 11 369–11 376, 2023.

[217] M. H. Cohen and C. Belta, "High order robust adaptive control barrier functions and exponentially stabilizing adaptive control lyapunov functions," in *2022 American Control Conference (ACC)*. IEEE, 2022, pp. 2233–2238.

[218] A. Isaly, O. S. Patil, R. G. Sanfelice, and W. E. Dixon, "Adaptive safety with multiple barrier functions using integral concurrent learning," in *2021 American Control Conference (ACC)*. IEEE, 2021, pp. 3719–3724.

[219] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," *arXiv preprint arXiv:1611.03530*, 2016.

[220] T. Z. Jiahao, K. Y. Chee, and M. A. Hsieh, "Online dynamics learning for predictive control with an application to aerial robots," in *Conference on Robot Learning*. PMLR, 2023, pp. 2251–2261.

[221] J. A. Preiss, W. Honig, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3299–3304.

[222] L. Pan, Y. Wang, and N. Ayanian, "Hierarchical trajectory (re) planning for a large scale swarm," *arXiv preprint arXiv:2501.16743*, 2025.