HUMAN-IN-THE-LOOP MODEL PREDICTIVE TRAJECTORY GENERATION FOR FLOCKS OF DRONES

HUMAN-IN-THE-LOOP MODEL PREDICTIVE TRAJECTORY GENERATION FOR FLOCKS OF DRONES

BY

ALI GRIVANI, B.Sc.

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

Master of Applied Science (2023) McMaster University

(Electrical & Computer Engineering) Hamilton, Ontario, Canada

TITLE:	Human-in-the-Loop Model Predictive Trajectory Generation for Flocks of Drones
AUTHOR:	Ali Grivani B.Sc. (Mechanical Engineering), Sharif University, Tehran, Iran
SUPERVISOR:	Shahin Sirouspour
NUMBER OF PAGES:	xv, 100

Lay Abstract

The rise of unmanned aerial vehicle technology and the increase in their accessibility have made them viable solutions for serious missions such as search and rescue operations. Complex cooperative tasks can be conducted via a collection of drones which can show higher levels of robustness and agility as a system. Although repetitive and simple actions can be easily automated, real-world problems are unpredictable in which complex decision-making is involved. Such scenarios can be tackled by the presence of a human supervisor to empower the system with strong cognitive capabilities. This thesis presents a multi-layer control framework for human-in-the-loop operation of a flock of unmanned aerial vehicles. This method continuously optimizes the drones trajectories to adhere as closely as possible to operator's motion commands while avoiding collisions among them and with obstacles in their task environment. This new control framework is successfully validated in both simulations and experiments in a laboratory environment.

Abstract

This thesis presents a novel architecture for human-in-the-loop control of multiple drones. The design of such systems must address several challenges at the same time. The drones must avoid collisions with each other and with obstacles in their task environment while following operator's command as closely as possible to navigate their environment. To this end, they should be able to adjust their pre-defined desired formation and, if needed, transition to alternative formations to ensure collision-free operation in their task environment while following the operator's commands.

The proposed control strategy is a central algorithm with multiple stages and relies on formulating and solving convex optimization problems in real time to achieve the control objectives. The operator provides reference velocity commands for the flock of drones to move them in the task environment. The algorithm creates linear collision avoidance constraints and distributes the operator's commands among the drones through a number of intermediate steps. It generates reference trajectories for the drones motion by solving a model-based optimization problem over a receding horizon. Conventional trajectory controllers generate the control inputs for individual drones.

Prospective formation shapes are obtained for the drones by formulating and solving parallel convex optimizations, considering the operator's reference command and

the obstacle-free space. While keeping the convexity of the optimization problem, the proposed algorithm allows for the presence of obstacles in the middle of the formation. This is achieved by properly assigning obstacle-free regions to each agent separately in the formation. In addition, safe convex regions in the form of linear inequality constraints are generated in the direction of the operator's commanded velocity. Moreover, constraints are introduced to avoid inter-drone collisions at each step. Trajectory optimization is formulated as a quadratic programming problem similar to model predictive control schemes to minimize deviation from human operator's command.

The effectiveness of the proposed control algorithm is initially verified by simulating two different operational scenarios. Furthermore, the algorithm is implemented on actual hardware to operate a flock of three drones in a laboratory setting. The implementation of the algorithm in C++ utilizes high-performance computation techniques to achieve sufficiently high real-time control update rates for smooth and stable operation of the drones.

Acknowledgements

This research could not have progressed without direct moral support and technical advice from my supervisor, Dr. Shahin Sirouspour. I would like to express my gratitude for the opportunity to work under his supervision.

I would also like to express my appreciation toward my family, friends, and Cheryl Gies for their support and encouragement throughout my graduate education at McMaster University.

Contents

List of Figures

List of Tables

Notation, Definitions, and Abbreviations

Notation

$s \in S$	s is an element of the set denoted by S.
d	Dimension of the workspace, $d = 2$ or $d = 3$.
$\mathbb{N}$	The set of Natural Numbers.
$\mathbb{R}$	The set of Real Numbers.
$\mathbb{R}^a$	The set of all vectors with $a \in \mathbb{N}$ real elements.
$\mathbb{R}^{a \times b}$	The set of all matrices with real elements, a rows, and b columns.
$\mathbf{A}_{m \times n}$	Matrices are denoted by upright bold letters and $\mathbf{A} \in \mathbb{R}^{a \times b}$.
$\mathbf{v}$	Vectors are typed in a bold font.
$\mathbf{A} \succ 0$	Matrix is positive definite.
$\mathcal{S}_1 \backslash \mathcal{S}_2$	Elements of $\mathcal{S}_1$ which are not a part of $\mathcal{S}_2$

Abbreviations

GPS	Global Positioning System
MAPF	Multi-Agent Path Finding
MIP	Mixed-Integer Programming
MILP	Mixed-Integer Linear Programming
MIQP	Mixed-Integer Quadratic Programming
MPC	Model Predictive Control
RHS	Right Hand Side
SVM	Support Vector Machine
UAV	Unmanned Aerial Vehicles

Chapter 1

Introduction

1.1 Motivation

Unmanned Aerial Vehicles (UAVs) have become increasingly popular in applications such as search and rescue, cargo transportation, aerial imagery, surveillance, and inspection missions. The natural agility of aerial vehicles, along with modern navigation technologies, remote control, and electric batteries, have made UAVs indispensable when human presence is not achievable [1]. Widespread availability of simple low-cost fully electric UAVs has further encouraged their use. Large investments in low-emission UAV-based solutions have made them a viable platform for use in applications such as goods transportation. Increasing allocation of financial resources to the growing UAV industry and market has spurred research in various relevant fields such as aerial robotics and control engineering.

The performance of a UAV is constrained by the limits of its power source (e.g., battery) and its thrust power capability. These constraints significantly impact its

flight range and payload capacity. Increasing the battery size to address these limitations would increase the vehicle size and could lead to other design compromises [2]. The flight time can be particularly restricted in cargo transportation, and alternative designs may be needed to carry large payloads. Autonomous recharging methods have been proposed to mitigate the battery-life problems [3]. Alternatively, multiple smaller drones can be utilized to carry large payloads. This has motivated significant research in multi-drone systems in the past decade [2, 4, 5, 6].

Multi-drone systems can effectively address some of the challenges of single-drone systems. They are usually more fault tolerant and can reduce operational risks compared to their single-drone counterparts. The arrangement of multiple drones in a flock can be highly reconfigurable and flexible for cooperative tasks. In addition to enhancing robustness and reliability of the system [7], they can also improve physical capabilities, for example by increasing surveillance coverage or payload capacity in transportation [5].

Autonomous robotics systems have been studied in depth during the past two decades. The majority of the existing literature on aerial robotics research has focused on designing frameworks and controllers for scenarios where no human intervention takes place [4, 5, 6, 8]. Regular missions in controlled environments are ideal for such applications. However, many real world tasks are prone to uncertainties and unpredictability of the task environment. Surveillance, search and rescue, and many inspection tasks are examples in which fully autonomous systems may be difficult to build and may not be sufficiently reliable. Such applications could greatly benefit from human intervention particularly for high-level decision-making tasks to overcome challenges encountered due to the unpredictability of the task environment [9, 10, 11].

All things considered, the human operator can play an important role in the future of many aerial robotic applications. Research is required to resolve the challenges of human-drones interactions. This thesis focuses on the control and navigation of a flock of quad-rotors operated by a human operator. The flock of drones is expected to follow the operator's high-level velocity commands, while avoiding collisions among each other and with obstacles in their task environment. A control framework is presented to relate the human reference commands with the collective navigation of the drones as a group, and with their individual motion control objectives.

1.2 Problem Statement

The majority of the existing literature in the field is dedicated to solving the problems around point-to-point motion planning of a group of robots without involving the presence of a higher-level decision maker, such as a human operator in the procedure. While control and operation of a single UAV is well-studied in the past [12, 13], the particular problem of a single operator controlling multiple drones has received very little attention. This thesis focuses on developing a control strategy for such application scenarios.

Many drone motion planning and control methods pre-compute optimal trajectories. Their main challenge is how to create suitable constraints to ensure the safety of the navigation in the presence of obstacles. On-the-fly motion planning is more challenging since the trajectories can no longer be computed offline as the destination and environment may vary in real time. Optimization-based planning and control approaches have been popular in this domain. However, these methods tend to be computationally expensive, making their real-time implementation

challenging. The advancements in computing technologies and convex formulation of optimization problems, which guarantee deterministic convergence to the solution, have helped reduce the computation time. Optimization-based approaches can accommodate various constraints in the problem formulation, allow for consideration of uncertainty, while obtaining the best control inputs for the system. This thesis adopts an optimization-based framework for formulating and solving the problem of human-in-the-loop control of a flock of drones. In particular, it seeks to formulate the trajectory planning as a convex optimization problem.

Collision avoidance is one of the main aspects of robot motion planning. In group navigation problems, group members must maintain a safe distance from obstacles as well as from other group members while moving in the task environment. Thus, the proposed framework in this thesis must assist the operator in maintaining a minimum safe distance among drones, and between the drones and obstacles in the task environment. Another objective of interest in multi-agent robotic systems and applications is formation control, e.g. in aerial transportation [14], or surveillance and reconnaissance [15]. This thesis seeks to incorporate a formation control strategy in its proposed approach for human-in-the-loop control of a flock of drones.

1.3 Thesis Contributions

This thesis presents a novel framework for human-in-the-loop control of multiple UAVs. There are two unique aspects that distinguish the proposed method from current solutions in the literature. First, existing optimization-based frameworks are mainly to solve the point-to-point trajectory planning problems, which are often carried out offline. Second, human-supervised multi-agent systems are mostly designed

via potential fields, while this thesis approach is purely optimization-based. Potential fields are mathematically simple to implement, but they are prone to local minima and are not as flexible as optimization-based approaches. Optimization-based model predictive control schemes provide great flexibility in setting up the control objectives and constraints. They are usually more computationally expensive than potential fields. However, convex formulation of the optimization problem can reduce their computations and yield globally optimal solutions.

In this thesis, the navigation problem is formulated as a convex quadratic optimization with linear constraints. Unlike existing solutions, the proposed method in this thesis does not need the formation shape to lie within a totally obstacle-free convex area. A novel algorithm is presented that assigns suitable convex areas to each agent individually. Infeasibility could happen due to the design of linear constraints. To resolve this, multiple reference formations are considered and solved in parallel, improving real-time computation efficiency.

To avoid collisions with obstacles, polyhedral convex regions aligned with the desired direction of movement are generated. This is inspired by the work in [16, 17]. Linear inequality constraints are designed based on priority and direction of movement to keep distance between drone pairs in the flock more than a minimum safe distance. These constraints are incorporated in a model-based optimization to compute the drones reference trajectories in a receding horizon manner. Deviation from the optimal formation and reference human input velocity are both considered in the trajectory design for each drone.

The proposed controller is centralized in the sense that all the computations are carried out on a single computation unit. The optimization model is quite flexible

and can be easily modified for use with different robotic platforms. The effectiveness of the proposed control strategy is demonstrated via numerical simulations as well as real-time experiments in an indoor laboratory setting.

1.4 Thesis Organization

The rest of this thesis is organized as follows. Chapter 2 is dedicated to reviewing the existing literature where applications and state-of-the-art strategies in motion control of multi-drone systems are explored. In Chapter 3, an overview of the proposed algorithm is presented, and basic assumptions and definitions are introduced. Chapter 4 is mainly concerned with the formulation of the optimal formation. Next, a geometric path-finding method is presented in Chapter 5 along with the strategies to obtain collision avoidance constraints along these paths. Further in Chapter 6, the model predictive optimization is presented where the objective and constraints are formulated. In Chapter 7, real-time controllers, numerical simulations, and experimental results are presented. Finally, this thesis will be concluded in Chapter 8 where potential future research topics have also been considered.

1.5 Related Publication

- A. Grivani, S. Sirouspour, "Human-in-the-loop Model Predictive Trajectory Generation for Flocks of Drones", *to be submitted to* IEEE Robotics and Automation Letters.

Chapter 2

Literature Review

2.1 Unmanned Aerial Vehicles

Recent advances in manufacturing, control, and communication technologies have led to a vast range of designs for unmanned aerial vehicles (UAVs). The main idea of having an aerial vehicle fly autonomously or being controlled remotely was formed more than a century ago [18]. Aerial vehicles can occupy a space larger than a plane [1], or they might easily fit inside the palm of a hand [19]. In addition to the conventional fixed-wing and quad-copter designs, continuous developments are being carried out worldwide, and novel structures are introduced every year [20, 21]. Wide availability of UAVs in various form factors and price points has led to their increasing use in many applications.

Their maneuverability, simplicity of use, and cost-effectiveness have made them popular in civil applications, particularly multi-rotor designs such as quad-rotors. Their use in cargo and parcel delivery has already been investigated in research and industrial settings [22, 23]. Not requiring terrestrial infrastructure has also made them

a good candidate in medical supply delivery and rescue missions [24]. Furthermore, various types of sensing equipment can be mounted on drones which facilitates their usage in surveillance, search and rescue missions without humans being required to be present on site [25, 26]. Their application in urban areas extends to monitoring construction sites, buildings, and bridges where visual information can be accessed with minimal delay and inspections can be carried out at low cost and high frequencies [27].

Quad-rotors with electric propulsion have been particularly popular due to their relatively small size, ease of manufacturing, low cost, and high agility. On the downside, their battery capacity and life can severely restrict their flight time. The flight time may be extended by autonomous battery charging schemes or mechanisms that facilitate battery replacements [28, 29, 30]. Limited payload capacity is another concern for quad-rotors, especially in transportation applications. Multiple drones can be used to cooperatively carry the payload to overcome this limitation in some applications [2]. Outside transportation applications, multi-drone systems can provide the following advantages:

- **Increased Sensing Capabilities**: Surveillance, inspection, monitoring, and mapping operations would benefit from an increase in the number of sensors and coverage area afforded by having multiple drones [31, 32].

- **Robustness and Safety**: Single-drone operation is dependent on one agent's capabilities, and any failure would usually result in mission termination. Systems with multiple drones can be potentially more tolerant to faults in individual drones [33]. Smaller drones working as part of larger groups can reduce liability and risks associated with operating larger aerial vehicles.

- **Enhanced Maneuverability**: Multiple cooperating UAVs provide more degrees of freedom and allow for executing maneuvers that might be difficult to carry out using only one vehicle [6, 34]. In aerial transportation, this would allow for more control over payload orientation and can greatly help in coping with external disturbances on the system.

Multi-rotor drones have been subject of very significant research in the last two decades. They are agile and can freely move in three-dimensional space, whereas most ground robots are confined to planar movements. Under-actuation is one of the well-known limitations of the popular quad-copters and which significantly complicates their motion planning and control [2, 35, 36]. Numerous methods of localization, mapping, and trajectory planning have been proposed and deployed in both indoor and outdoor environments for multi-rotor drones [4, 17, 37].

2.2 Multi-Drone Systems Motion Control

Significant research has focused on developing efficient quad-rotor trajectory controllers [35, 38, 39, 40]. These controllers usually operate on one level higher than the actual attitude controllers commonly implemented on embedded on-board flight controllers. Motion control of a quad-rotor is generally complicated by its under-actuated dynamics. At a higher level, collision avoidance adds to the complexities of its control. These issues are also inherent in multi-drone systems using quad-rotors. Motion control for multi-drone systems involves another higher-level hierarchy to plan the trajectories and devise a strategy for safe and efficient navigation of the whole group. Motion planning and control strategies for these systems are often proposed

in hierarchical algorithms that start with a high-level abstract goal and output the individual control inputs.

A popular approach to the design of efficient and collision-free trajectories in multi-agent systems is to formulate the problem as a Mixed-Integer Program (MIP). In this type of problems, optimal trajectories are obtained by minimizing a quadratic (Mixed-Integer Quadratic Programming - MIQP) or a linear (Mixed-Integer Linear Programming - MILP) objective where some of the decision variables are integer numbers. Mellinger, Kushleyev, and Kumar proposed a centralized MIQP approach for trajectory generation in which collision avoidance was accomplished via having integer-domain constraints [41]. Their approach accommodates quad-rotors with different sizes within the group and takes into account the differential flatness property of their dynamics. Their method was tested in simple environments with a small number of drones and obstacles and the time to find a solution was in the order of hundreds of seconds. In 2013, they extended their approach to planning three-dimensional trajectories for up to twenty drones where the computation time was reduced by distributing the computation among smaller groups [5]. These approaches require the starting point and the destination to be known. Their optimization problem is non-convex, and their computational time increases exponentially with the number of integer variables. Consequently, they are not suited for real-time applications.

Trajectories can be considered as optimized and smooth versions of crude roadmaps connecting two points in the space. This perspective relies on path-planning methods and iterative trajectory refinement. Multi-Agent Path Finding (MAPF) is a well-known and challenging problem in the robotics field and solutions have been proposed to extend the existing methods for multi-agent scenarios [42, 43] without

optimizing trajectories for vehicle dynamics. This results in piece-wise linear paths which are not optimal as they will require numerous stop-and-go along the way, or otherwise, they will not be dynamically feasible. Hönig *et al.* introduced a novel approach where by combining discrete planning methods, trajectory optimization, and iterative improvements, safe and smooth trajectories were obtained [4]. They have combined sampling-based road-map planning with graph search algorithms to achieve the paths annotated with possible conflicts. The road-maps will be scheduled and time steps will be assigned based on a MAPF solver to ensure the paths do not possess any possible inter-agent collisions. In addition, obstacle avoidance is further guaranteed by generating convex safe corridors by solving hard-margin support vector machine (SVM) to generate separating hyper-planes. Safe corridors are shaped around the discrete path, and trajectory optimization will be formulated such that it enforce the results to be constrained with a certain convex hull of all the given control points. Although the resulting trajectories will be smooth, safe, and the method can be applied with large number of drones in a complex environment, this motion planning is more suited for offline planning since the computation time is within a number of seconds. Analogous approaches for obstacle-robot collision avoidance can be found in [44, 45].

Collision avoidance is a significant problem in multi-robot systems. In such systems, strategies are needed to ensure that the robots would not collide with each other or with obstacles in their environment. Artificial potential fields or barrier functions are based on mathematically simple models that produce reactive repulsive forces acting on the robots in order to prevent collisions [6, 46, 47]. While their low

computational cost is highly desirable, they are known to be vulnerable to local minima, and need to be manually tuned to avoid undesirable movements. Alonso-Mora *et al.* proposed optimization-based solutions that combine convex optimization with potential force field for increased safety [8]. They incorporate the concept of velocity obstacles to determine the velocities that culminate in a collision within a certain horizon. The states are then constrained in convex cones to avoid collisions in the optimal reference states. Their algorithm which can handle dynamic obstacles can be implemented and used in real-time. However, it can not accommodate a human operator in the loop.

Model Predictive Control (MPC) schemes have gained popularity in the realm of optimization-based planning methods in the past decade. In these methods, the control inputs are the solution to an optimization problem solved repeatedly over a rolling finite time horizon. Only the first (few) step(s) of the optimal control input sequence is given to the system after solving each optimization problem. Onboard solutions have been proposed in the literature for embedded trajectory control applications [48, 49]. MPC as a higher-level off-board controller has also received considerable attention in the literature [50]. Baca *et al.* proposed and deployed an MPC-based trajectory tracking for a group of drones in an outdoor setting [51]. In contrast to their previous work [48], the proposed MPC is not intended for controlling the aerial vehicle directly; rather, it modifies a reference trajectory for an underlying non-linear controller. Only altitude is adjusted to avoid collisions, which is sensible when drones fly in open fields. Moreover, Luis and Schoellig proposed a point-to-point trajectory generation framework that is built upon a distributed MPC [52]. They use a double integrator to model the agents and to predict their future states. An on-demand

scheme is proposed for avoiding possible detected collisions in the future. The distribution of the algorithm along with linear estimated soft collision constraints are key factors for its scalability. However, this method can not guarantee collision-free trajectories and can result in deadlocks. The same research group extended their work to an online trajectory generator which was tested on a group of 20 quad-rotors in real-time [53].

2.3 Formation Control

Nature provides numerous examples of living creatures working cooperatively in formations to achieve common objectives. Likewise, there are many advantages to driving autonomous robots in a certain spatial pattern. Maintaining a formation within a group of payload-carrying robots can increase their overall efficiency and reduce cost of operation through load sharing. Moreover, in reconnaissance and surveillance missions, formation control can help maximize coverage area and minimize intercommunication delays [54]. Formation control introduces a set of new challenges in the control of multi-agent systems.

A group formation can be defined as an array of relative positions with respect to a selected leader in the group. The idea is usually referred to as *leader-follower* formation control, where while the leader is given a reference trajectory, the other units are supposed to determine their location based on pre-defined spatial offsets from the leader. Multiple variations of this concept have been proposed in the literature [55, 56, 57]. While leader-follower methods are relatively simple to implement, they have no explicit feedback on the formation and lack robustness due to dependency on the leader. In applications such as search and rescue the formation may have to

split into multiple *clusters* each with their own leader. Hu *et al.* proposed a method [58] that separates the clusters based on task priorities using a game-theory-based approach.

Formations are defined as virtual rigid bodies in *Virtual Structure*-based approaches. The kinematic model of the virtual structure can be used to derive the robots desired motion from the reference trajectory of the virtual rigid body. Every agent then can be controlled with their custom reference trajectory using conventional trajectory tracking controllers [59, 60]. These methods lack bilateral feedback and require a central processing unit to carry out the computations. *Behavioral* methods build on artificial potential fields to compute repulsive or attractive forces that would drive each robot toward the ideal location in the formation. They are well suited for real-time applications due to their simple computations. Additionally, their computations can be distributed among the agents, eliminating the need for a central computing unit. However, they result in complex and difficult mathematical analysis or can yield undesirable outcomes if competing forces are averaged. Examples of behavioural implementation of formation control can be found in [61, 62]. The last two approaches can be combined to provide reactive collision avoidance and formation control as in [63] with pre-defined shape transitions and a formation library to allow for switching between different arrangements.

While moving in a pre-defined rigid formation can be ideal in some applications, many cooperative tasks require the group formation to be flexible and possibly deform whenever necessary. For instance, navigating through a narrow passage may not be possible unless the agents arrange in a linear formation. It is also important

to maintain communication among the agents and minimize any delay in the communication. Continuum deformation schemes can be applied to achieve such goals where each agent is described as a point inside a body able to deform under homogeneous transformations [64]. Rastgoftar and Atkins used this approach in a system of payload-carrying robots while passing through a narrow passage [11]. They used kinematic relation to define the motion equations with respect to three leaders and the payload and used linear convex approximation for obstacle-free spaces. Inter-agent collisions were avoided by imposing constraints on the homogeneous transformation, which was defined based on the leaders' position. The concept has been evaluated both in simulation [11], and real-time experiment [64]; however, the possibility of obstacles passing through the middle of formation is not discussed in this approach.

Optimization-based methods for formation transition and planning have been proposed in the past. Alonso-Mora, Baker, and Rus presented a method to obtain formation with a centralized sequential convex programming [65]. In order to prevent collisions with obstacles, the convex-shaped formation is constrained to a collision-free convex area generated in the direction of movement; the formation shape is optimized with respect to its outer vertices while the locations of the internal robot are not considered. Further down their hierarchical algorithm, inter-agent collision avoidance is considered via a lower-level trajectory planner. This work was further extended to a distributed formation control scheme in [66]. In both papers, the method was designed to allow for multiple template formations and dynamic obstacles in the environment. Nevertheless, the interaction of humans was only at its highest level, and the formations were restricted to have a convex shape.

2.4 Human-in-the-Loop Methods

In addition to fully autonomous multi-agent systems, semi-autonomous systems have also been studied in the past. In the so-called *Human-in-the-Loop* systems, a human operator drives the whole system while tasks such as collision avoidance could be performed autonomously. The operator can also facilitate high-level and cognitive decision-making processes that are critical in dealing with unforeseen situations in applications such as search and rescue. One of the main challenges of single-operator/multi-robot human-in-the-loop systems is how to interpret a single operator's commands for coordinating movement of multiple agents.

A rich set of solutions are available in the literature about how to interpret human actions for a swarm of robots. Mastellone *et al.* demonstrated a case for ground robots moving in formation where an operator commands the leader velocity [67]. Authors in [68] designed an architecture that includes an interface for the human expert at the highest level. Nagi and others reported a vision-based mechanism to interact with a swarm of UAVs through human gestures [69]. Tracking fingertips to perform a cooperative aerial manipulation is investigated in [70]. Applying an end-to-end abstraction of human touch to rectangular bounding boxes for desired positions of the robots was proposed in [71] as one possible solution. Researchers in [72] applied an interface where human would draw a rough estimation of formation shape and the flock will be brought to balance and stability complying with geodesic constraints.

Other architectures have been investigated with continuous human involvement where strategies are presented to assist the operator in safe navigation of robots. Application of direct human force input on the payload carried by a flock of UAVs has been studied in [11] where the rigid-body dynamics is used to transmit the command

to leaders in the group. Franchi *et al.* proposed a bilateral teleoperation framework in which the operator would be guiding the leader of the group via a velocity-based damper model focusing on the energetic passivity of the system [73]. Inter-robot and obstacle-robot interactions have been considered by designing proper potential fields, and the possibility of splitting or merging has also been considered. In another idea under the haptic teleoperation domain, Lee *et al.* devised a multi-layer architecture where the user command is applied to equations of motions for a set of evolving *virtual points* (VPs) [74]. Robots' interactions with each other or the environment are modeled with artificial potentials, which affects each VP's dynamics and finally, a trajectory controller will take care of following the trajectory evaluated by integrating the VP dynamics. In contrast, Zhou, Wang, and Schwager proposed an idea where the human input is interpreted as the orientation and the velocity of a flying virtual structure, and the user will be assisted with artificial potential forces to account for other motion constraints [63].

2.5 Summary

In this chapter, first an overview of applications of aerial robotics research was briefly presented. Next, the importance of multi-drone systems was highlighted, followed by a review of existing challenges and solutions for their motion control problem. Continuing on multi-agent control challenges, some of the existing approaches for formation control were reviewed. Finally, the state-of-the-art on the human-multi-robot system was surveyed.

Human-in-the-loop approach for swarms of aerial vehicles is still in its early stages, and existing works in literature are based on reactive methods and potential fields,

which are simple to implement and can be used in stability control theorems. However, they can be prone to local minima, not analytically tractable, and may result in unwanted behaviors. Application of convex optimization for human-in-the-loop control of multi-agent systems has not been investigated in the past. Optimization-based methods are usually more computationally expensive than reactive methods. Hence, the existing literature is mostly developed around offline computations. A few existing online model-based optimization schemes have not considered agents moving in formation or continuously receiving guidance from an operator's command. Finally, the current optimization-based approaches do not consider the possibility of allowing the formation surrounding the obstacles, i.e., non-convex formation shapes.

Chapter 3

Algorithm Overview, Assumptions, and Modeling

This chapter presents a high-level and general architecture of the proposed framework for human-in-the-loop control of a flock of drones. Solving real-world problems requires mathematical models justified by reasonable assumptions. The modeling assumptions are stated and the scope of the thesis is defined. Mathematical models of the environment, the robots, and the flock's formation are outlined.

3.1 Algorithm Overview

A general block diagram of the proposed control algorithm is presented in Figure 3.1. The human operator provides input to the system using a human-to-machine interface. The user input is interpreted as a reference velocity for the entire group. The agents are controlled by individual trajectory tracking controllers that move them along a given reference trajectory. The middle components of the algorithm work

cooperatively to provide well-defined trajectories that comply with the objectives mentioned in the problem statement (See Section 1.2).

The reference velocity is used as a guideline to find a section of the environment where the whole flock of drones can pass safely without colliding with obstacles. It also defines the desired speed and direction of movement for the flock. Obstacle-free regions of the environment are assigned to each agent based on the projection of reference formation. An optimization problem is formulated and solved to determine the best shape for the formation that can be fitted in the obstacle-free regions. This along with information on surrounding obstacles is used to generate convex linear collision avoidance constraints.

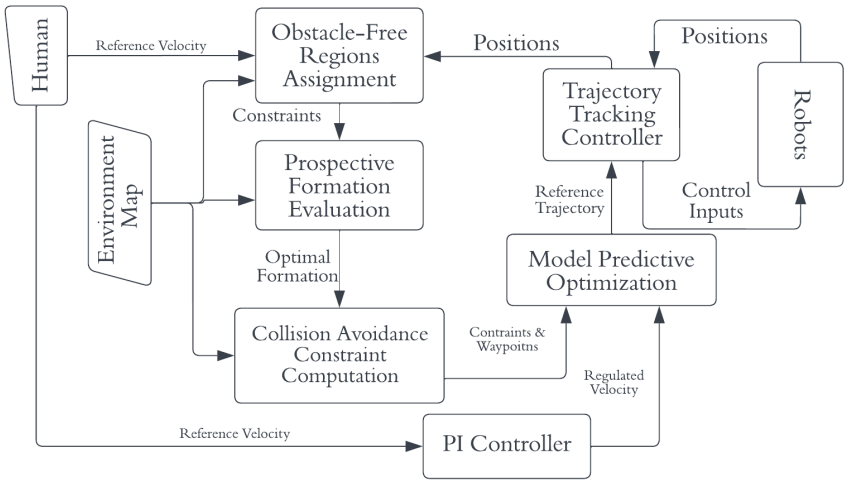

Figure 3.1: Proposed framework and its components

In the last step, a Model Predictive Controller (MPC) is formulated and solved as a convex quadratic optimization. The optimizer solves the problem based on the constraints obtained from the previous step, and the reference velocity for the whole multi-agent system's navigation. The human's input is regulated before passing it to

the optimization solver using a Proportional-Integral (PI) controller. This controller compensates for the deviation between flocks velocity and the reference velocity command.

3.2 Assumptions

The following is assumed throughout the rest of this thesis:

- The environment and its map are known and given. Any process involving the environmental perception is out of the scope of this work.
- The obstacles in the environment are static and not moving.
- The agents positions in global frame are known. In the indoor laboratory setting of this thesis, the drone positions are measured in real time using a motion capture system. Localization in outdoor environments can be achieved using the global positioning systems (GPS) such as Real-Time Kinematic (RTK) [75] technology.
- The control algorithm in Section 3.1 is implemented on a central computer without peer-to-peer communication among the drones.

3.3 System Model

This section presents some of the key mathematical models and definitions that will be used in the rest of the thesis.

3.3.1 Problem Statement

Having $N \in \mathbb{N}$ number of agents $\{\mathcal{D}_i | i \in \mathcal{I} = \{1, 2, 3, .., N\}\}$, and $N_o \in \mathbb{N}$ static convex-polygon-shaped obstacles $\{\mathcal{O}_j | j \in \mathcal{J} = \{1, 2, ..., N_o\}\}$, the goal is to find optimal real-time trajectories for the agents with four main characteristics:

1. The collective motion of the agents follows the operator's reference velocity command $\mathbf{v}_h \in \mathbb{R}^d$ where $d \in \{2, 3\}$ is the dimensions of the workspace.
2. Agents avoid collisions with each other and obstacles in the task environment.
3. The agents move within pre-defined desired formation.
4. The trajectory planning and control adhere to physical constraints of the system.

3.3.2 Agents

Collision Avoidance Model

Each agent, a drone in this thesis, is modeled as a sphere and its position $\mathbf{q}_i \in \mathbb{R}^d$ is defined as the center of the sphere. The space occupied by the i^{th} drone ($i \in \mathcal{I}$) is denoted by $\mathcal{D}_i$ which is dependent on sphere's radius $r_i \in \mathbb{R}^+$ and drones position $\mathbf{q}_i$:

$$\mathcal{D}_i(\mathbf{q}_i, r_i) = \{\mathbf{p} + \mathbf{q}_i \ : \ ||\mathbf{p}||_2 \leq r_i, \ \mathbf{p} \in \mathbb{R}^d, \ d \in \{2, 3\}\} \tag{3.3.1}$$

This model has been widely-accepted and well-adapted across the UAV literature [17, 52, 76]. More advanced models can be developed to consider aerodynamic effects such as *downwash* force [4]. These forces can be reflected in the geometrical shape by using a larger z-axis in the ellipsoid model.

Agent Dynamics

Each agent is considered to have the mass $m_i \in \mathbb{R}^+$ concentrated at the center of $\mathcal{D}_i$. A double integrator dynamics model is used to compute the evaluation of its states over time (analogous to [52]). Having the vectors defined as in Table 3.1,

Parameter	Notation	$d = 2$	$d = 3$
Position	$\mathbf{q}_i$	$[q_{x_i}\ q_{y_i}]^T$	$[q_{x_i}\ q_{y_i}\ q_{z_i}]^T$
Velocity	$\dot{\mathbf{q}}_i$	$[\dot{q}_{x_i}\ \dot{q}_{y_i}]^T$	$[\dot{q}_{x_i}\ \dot{q}_{y_i}\ \dot{q}_{z_i}]^T$
Control Input	$\mathbf{u}_i$	$[u_{x_i}\ u_{y_i}]^T$	$[u_{x_i}\ u_{y_i}\ u_{z_i}]^T$

Table 3.1: Position, Velocity, and Control Input Definitions

the dynamical model can be written as:

$$\mathbf{q}_i[k+1] = \mathbf{q}_i[k] + \Delta T \dot{\mathbf{q}}_i[k] + \frac{\Delta T^2}{2m_i}\mathbf{u}_i[k] \tag{3.3.2}$$

$$\dot{\mathbf{q}}_i[k+1] = \dot{\mathbf{q}}_i[k] + \frac{\Delta T}{m_i}\mathbf{u}_i[k] \tag{3.3.3}$$

where $\Delta T \in \mathbb{R}^+$ is the discretization step and k shows the time step.

This model is simple and advantageous for use in computationally intensive approaches such as model predictive control problems. Although not originally in this form, the dynamics of a quad-rotor can be effectively rendered into those of a double integrator through low-level controllers.

3.3.3 Environment Model

As stated in Section 3.2, the task environment is assumed to be known. In fact, at each time step, the information for a window of $\mathcal{W}[k] : \mathbf{l}_{\mathcal{W}}[k] \leq \mathbf{x} \leq \mathbf{u}_{\mathcal{W}}[k]$ with

respect to the world frame is known, where $\mathbf{x} \in \mathbb{R}^d$ and $\mathbf{l}_{\mathcal{W}}, \mathbf{u}_{\mathcal{W}} \in \mathbb{R}^d$ are the window boundaries. The obstacles $\{\mathcal{O}_j\}$ are assumed to be static and be represented as convex polyhedra, i.e.,

$$\mathcal{O}_j = \{\mathbf{x} \in \mathbb{R}^d \ : \ \mathbf{A}_{o_j}\mathbf{x} \leq \mathbf{b}_{o_j}, \ \mathbf{A}_{o_j} \in \mathbb{R}^{m_j \times d}, \ \mathbf{b}_{o_j} \in \mathbb{R}^{m_j}\} \tag{3.3.4}$$

Note that obstacles of other shapes can be enclosed inside polyhedrons. For collision avoidance purposes, a dilated version of obstacles $\{\bar{\mathcal{O}}_j\}$ will be used. The dilation is done by half of the volume of the largest robot as follows:

$$\bar{\mathcal{O}}_j = \{\mathbf{x} \in \mathbb{R}^d \ : \ \text{if } \mathbf{x} \notin \bar{\mathcal{O}}_j \implies \mathcal{D}_i(\mathbf{x}, r_{max}) \cap \mathcal{O}_j = \emptyset\} \tag{3.3.5}$$

where $r_{max} = \max\{r_1, r_2, .., r_N\}$. Consequently, the obstacle-free $\mathcal{W}_f[k]$ area can be defined by subtracting the dilated obstacles from the surrounding environment as follows:

$$\mathcal{W}_f[k] = \mathcal{W}[k] \backslash \{\bar{\mathcal{O}}_j\} \tag{3.3.6}$$

3.3.4 Formation Definition

A formation is defined by a set of vectors $\mathcal{S} = (\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_N)$ which denote the position of each agent with respect to the leader's position. If the team's leader is located at $\mathbf{q}_1 \in \mathbb{R}^d$, the position of each members would be given by:

$$\mathbf{q}_i = \mathbf{q}_1 + \mathbf{s}_i \tag{3.3.7}$$

Ideally, $\mathbf{s}_1 = \mathbf{0}$ which is the relative position of the leader with respect to itself; however, the leader's position may need adjustment due to collision avoidance constraints as it will be illustrated in Section 4.2. A reference formation $\mathcal{S}^r = (\mathbf{0}, \mathbf{s}_2^r, ..., \mathbf{s}_N^r)$ is defined as a priori to the problem. Furthermore, multiple formations can be defined as a prioritized list $\mathcal{L}_s = \{\mathcal{S}_1^r, ..., \mathcal{S}_k^r\}$ where the prospective formation can be searched based on the highest priority to the lowest. Additionally, the prospective formation is obtained through a convex optimization approach that will be presented in the next chapter.

Chapter 4

Exploring Prospective Formations

4.1 Obstacle-Free Regions Assignment

Given a reference velocity from the operator $\mathbf{v}_h$, the reference future positions of the agents can be projected in the direction of $\mathbf{v}_h$. However, these target positions $\{\hat{\mathbf{q}}_i^t\}$ where $\hat{\mathbf{q}}_i^t \in \mathbb{R}^d$ may not be always located in an obstacle-free space. A region $\mathcal{F}_i$ is assigned to any point $\hat{\mathbf{q}}_i^t$ selected from $\mathcal{W}[k]$ that would satisfy the following properties:

1. The region is free of obstacles and $\mathcal{F}_i \subset \mathcal{W}_f$.

2. It is a convex subset of $\mathbb{R}^d$. In other words, for any two distinct points $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^d$ inside $\mathcal{F}_i$, all points on the line segment connecting the two $\mathbf{y} = \tau\mathbf{x}_1 + (1 - \tau)\mathbf{x}_2$ would also lie inside the region ($\mathbf{y} \in \mathcal{F}_i$) where $0 \leq \tau \leq 1$.

3. The region is a polyhedron represented by a set of linear inequalities, i.e.,

$$\mathcal{F}_i = \{\mathbf{x} \in \mathbb{R}^d : \mathbf{A}_s\mathbf{x} \leq \mathbf{b}_s,\ \mathbf{A}_s \in \mathbb{R}^{n_s \times d}, \mathbf{b}_s \in \mathbb{R}^{n_s}\} \tag{4.1.1}$$

where $n_s \in \mathbf{N}$ is the number of hyperplanes defining the outer edges of the polyhedral spatial set.

The first property is necessary to avoid collisions with obstacles; however, it is not sufficient. To obtain an optimal formation using convex optimization techniques, the regions must be convex so they can be used as constraints in the optimization problem formulation.

The third property aims for polyhedral-shaped regions that are aligned with the environment representation (see Equation 3.3.4). Each side of a polyhedral obstacle is a part of a hyper-plane in the space where the obstacle would be located on one side of it. These hyper-planes help define an obstacle-free space assigned to a given point in the workspace. Algorithm 1 outlines the steps involved in assigning the regions.

For every given point $\hat{\mathbf{q}}_i^t$, a convex region $\mathcal{F}_i$ is initialized by the boundaries of the current workspace $\mathcal{A}_w$. The point is projected on all the hyper-planes defining the sides of an obstacle $\mathcal{O}$. Not all the projections would lead to a point on the $\mathcal{O}$; hence, only the hyper-planes with such property are returned as the output of *GetSidesWithProjection* function in Line 6. If such hyper-planes were found, the one with the shortest distance to the point would be appended to $\mathcal{F}_i$. Function *GetClosestHyperPlane* returns the hyper-plane with the shortest distance to the point. In addition, Line 9 highlights the fact that the hyper-plane must point towards the right half-space that does not include the obstacle.

There could be cases where the simple projection of $\hat{\mathbf{q}}_i^t$ may not lead to a point on $\mathcal{O}$. In such scenarios, all half-spaces described by the sides of the obstacles but pointing outward G would be considered as candidates to be appended to $\mathcal{F}_i$ (Line 11). Finally, among the members of G, which includes $\mathbf{q}_i^t$, the hyper-plane with the

Algorithm 1: Assigning Obstacle-free Regions

Data: Target Positions $\{\mathbf{q}_i^t\}$, Obstacles $\{\bar{\mathcal{O}}_j\}$, Workspace Boundary $\mathcal{A}_w$
Result: Convex Regions: $\{\mathcal{F}_i\}$

1 **for** $i \in \mathcal{I}$ **do**
2 $\quad \mathbf{q} \leftarrow \mathbf{q}_i^t$;
3 $\quad \mathcal{F}_i \leftarrow \mathcal{A}_w$;
4 $\quad$ **for** $\mathcal{O} \in \{\bar{\mathcal{O}}_j\}$ **do**
5 $\quad\quad \mathcal{A}_o \leftarrow$ GetConvexRepresentation($\mathcal{O}$) ;
6 $\quad\quad \{(\mathbf{a}_n, b_n)\} \leftarrow$ GetSidesWithProjection(q, $\mathcal{A}_o$);
7 $\quad\quad$ **if** $|\{(\mathbf{a}_n, b_n)\}| > 0$ **then**
8 $\quad\quad\quad (\mathbf{a}, b) \leftarrow$ GetClosestHyperPlane($\mathbf{q}$,$\{(\mathbf{a}_n, n_n)\}$);
9 $\quad\quad\quad (\mathbf{a}, b) \leftarrow (-\mathbf{a}, -b)$;
10 $\quad\quad$ **else**
11 $\quad\quad\quad G = \{(\mathbf{a}_n, b_n)\} \leftarrow$ ReverseAllHyperPlanes($\mathcal{A}_o$);
12 $\quad\quad\quad$ **for** $(\mathbf{a}_n, b_n) \in G$ **do**
13 $\quad\quad\quad\quad$ **if** $\mathbf{a}_n^T \mathbf{q} > b_n$ **then**
14 $\quad\quad\quad\quad\quad G \leftarrow$ RemoveFromSet(G, $\mathbf{a}_n, b_n$));
15 $\quad\quad\quad\quad$ **end**
16 $\quad\quad\quad$ **end**
17 $\quad\quad\quad (\mathbf{a}, b) \leftarrow$ GetFurthestHyperPlane(G, $\mathbf{q}$);
18 $\quad\quad$ **end**
19 $\quad\quad \mathcal{F}_i \leftarrow$ Concatenate($\mathcal{F}_i$, $\mathbf{a}$, b);
20 $\quad$ **end**
21 **end**

furthest distance (Line 17) is selected to be appended to $\mathcal{F}_i$.

Algorithm 1 iterates through all the agents and repeats the above-stated procedure for all the obstacles in $\mathcal{W}[k]$. This approach is deterministic, and would work regardless of whether the given point is inside an obstacle or not. For more illustration, Figure 4.1 demonstrates an example where one pentagonal obstacle is considered in a 2-dimensional workspace. Matching colors between the scatter points and the sides of the obstacle are used to show which side (half-space) of the obstacle is assigned to the point. Furthermore, Figure 4.2 depicts an instance where two obstacles are

present in the workspace, and the green shaded area is the result of concatenating two half-spaces $\{\mathbf{x} \in \mathbb{R}^2 | \mathbf{a}_1^T \mathbf{x} \leq b_1\}$ and $\{\mathbf{x} \in \mathbb{R}^2 | \mathbf{a}_2^T \mathbf{x} \leq b_2\}$, on top of the workspace boundary $\mathcal{A}_w$.

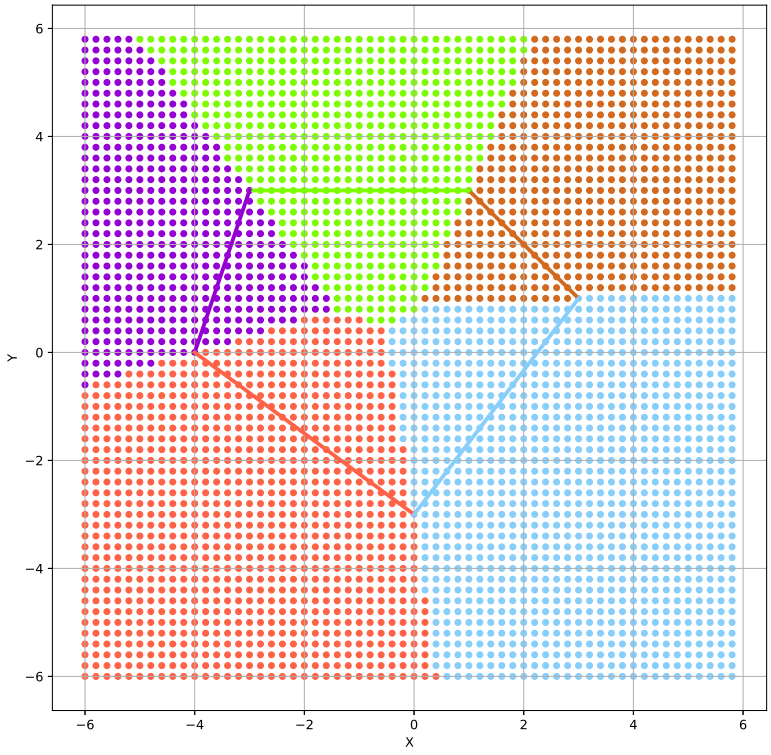

Figure 4.1: Example output of Algorithm 1. Solid lines show the boundary of the obstacle, and the color of the scatter dots indicates which side (hyper-plane) of the pentagonal obstacle it has been assigned to.

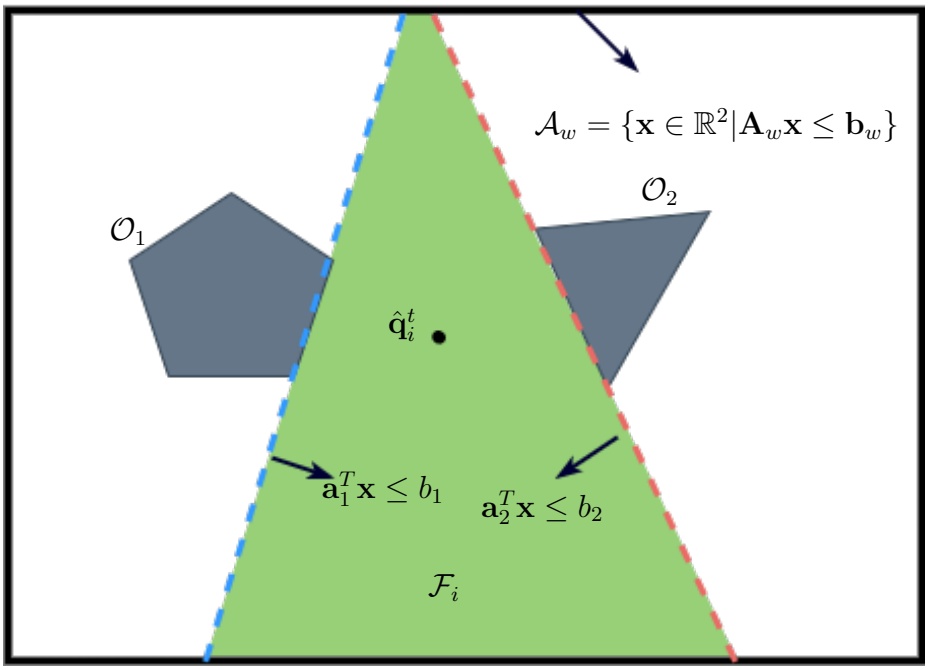

Figure 4.2: Green area ($\mathcal{F}_i$) is the concatenation of all half-spaces with $\mathcal{A}_w$ as a result of Algorithm 1 for point$\hat{\mathbf{q}}_i^t$ in between two obstacles ($\mathcal{O}_1$ and $\mathcal{O}_2$).

4.2 Optimal Formation

Obstacle-free regions $\{\mathcal{F}_i\}$ defined as a linear inequality can be used to formulate a convex optimization problem to achieve optimal formation in the next step. Maintaining a reference formation becomes critical in applications where robots sensing or communication capabilities are limited. There may be other application-related reasons that necessitate the agents following a certain formation. In adhering to a reference formation, the agents *optimal* formation must generally satisfy the following

criteria:

- The distance and the angle between the agents have a minimum deviation from the reference shape.
- All the agents must be located in obstacle-free spaces.
- All agents must maintain a safe distance from each other.

The deviation between the optimal formation $\mathcal{S}$ and the reference formation $\mathcal{S}^r$ can be denoted as $\Delta_s(\mathcal{S}^r, \mathcal{S}) = (\delta\mathbf{s}_1, \delta\mathbf{s}_2, \delta\mathbf{s}_3, ..., \delta\mathbf{s}_N)$ where

$$\delta\mathbf{s}_i = \mathbf{s}_i^r - \mathbf{s}_i, \quad i \in \mathcal{I},\ \mathbf{s}_1^r = \mathbf{0} \tag{4.2.1}$$

The optimal formation is the solution to the following optimization problem:

$$\underset{\delta\mathbf{s}_i}{\text{minimize}} \quad \sum_{i\in\mathcal{I}} \delta\mathbf{s}_i^T \delta\mathbf{s}_i \tag{4.2.2}$$

$$\text{subject to} \quad \mathbf{q}_i^t = \hat{\mathbf{q}}_1^t + \mathbf{s}_i^r + \delta\mathbf{s}_i \tag{4.2.3}$$

$$\mathbf{q}_i^t \in \mathcal{W}_f \tag{4.2.4}$$

$$||\mathbf{q}_i^t - \mathbf{q}_j^t|| \geq r_i + r_j, \quad i \neq j \in \mathcal{I} \tag{4.2.5}$$

Constraint (4.2.4) defines the obstacle-free condition and (4.2.5) represents the minimum distance requirements to avoid inter-agent collisions. Unfortunately, the optimization problem defined through equations (4.2.2) to (4.2.5) is generally non-convex making it unsuitable for real-time applications. A linear relaxation is introduced to overcome this issue.

It should be noted that the leader's initial position $\hat{\mathbf{q}}_1^t$ is not necessarily in $\mathcal{W}_f$

but the obstacle avoidance constraints would ensure that the eventual solution is in the obstacle-free space.

The safe regions $\{\mathcal{F}_i\}$ constructed using Algorithm 1 are used to estimate the obstacle-free workspace $\mathcal{W}_f$ in (4.2.4). The minimum distance constraint stated in (4.2.5) is a mutual relation that must be satisfied between the positions of any pair of agents in the flock. Having $\mathbf{s}_1$ defined as the leader's position in the formation, the distance between any two agents $i \neq j \in \mathcal{I}$ can be preserved by the following linear inequality:

$$\mathbf{e}_{ij}^T(\mathbf{q}_i - \mathbf{q}_j) \geq d_{ij} \tag{4.2.6}$$

where $\mathbf{e}_{ij}$ is the unit vector defined between the relative position of $\mathbf{s}_i$ and $\mathbf{s}_j$ as follows:

$$\mathbf{e}_{ij} = \frac{\mathbf{s}_i^r - \mathbf{s}_j^r}{||\mathbf{s}_i^r - \mathbf{s}_j^r||} \tag{4.2.7}$$

and $d_{ij} \in \mathbb{R}^+$ is the minimum safe distance between the two agents which must satisfy $d_{ij} \geq r_i + r_j$.

Using (4.1.1), (4.2.6), and (4.2.7), the optimization problem (4.2.2) can be re-written as:

$$\underset{\delta\mathbf{s}_i}{\text{minimize}} \quad \sum_{i\in\mathcal{I}} \delta\mathbf{s}_i^T \delta\mathbf{s}_i \tag{4.2.8}$$

$$\text{subject to} \quad \mathbf{e}_{ij}^T \left((\mathbf{s}_i^r + \delta\mathbf{s}_i) - (\mathbf{s}_j^r + \delta\mathbf{s}_j)\right) \geq d_{ij} \tag{4.2.9}$$

$$\mathbf{q}_i^t = \hat{\mathbf{q}}_1^t + \mathbf{s}_i^r + \delta\mathbf{s}_i \tag{4.2.10}$$

$$\mathbf{A}_{s_i}\mathbf{q}_i^t \leq \mathbf{b}_{s_i} \tag{4.2.11}$$

The constraint in (4.2.9) would keep the agents at safe distances from each other

but it would impose strict boundaries on the domain of the problem. This could lead to infeasible scenarios. In order to tackle this issue, we would be using a number of reference formations as mentioned in Section 3.3.4. The convex optimization problem (4.2.8) is solved for all these reference formations in $\mathcal{L}_s$ at the same time by taking advantage of parallel computing. This approach would enable us to choose the next available optimal solution based on a predefined priority in case the previous reference formation is infeasible. For instance, in the case of a flock with three agents, with three reference formations defined as shown in Figure 4.3, the optimization formulated in (4.2.8) can be solved in parallel with respect to each reference. They are prioritized such that $\mathcal{S}_1^r$ has the highest rank and $\mathcal{S}_3^r$ has the lowest. Considering this arrangement, Figure 4.4 depicts the result for three different scenarios. For instance, in the middle scenario, the optimal formation can be obtained both based on $\mathcal{S}_2^r$ and $\mathcal{S}_3^r$, but $\mathcal{S}_2^r$ is considered since it has a higher priority.

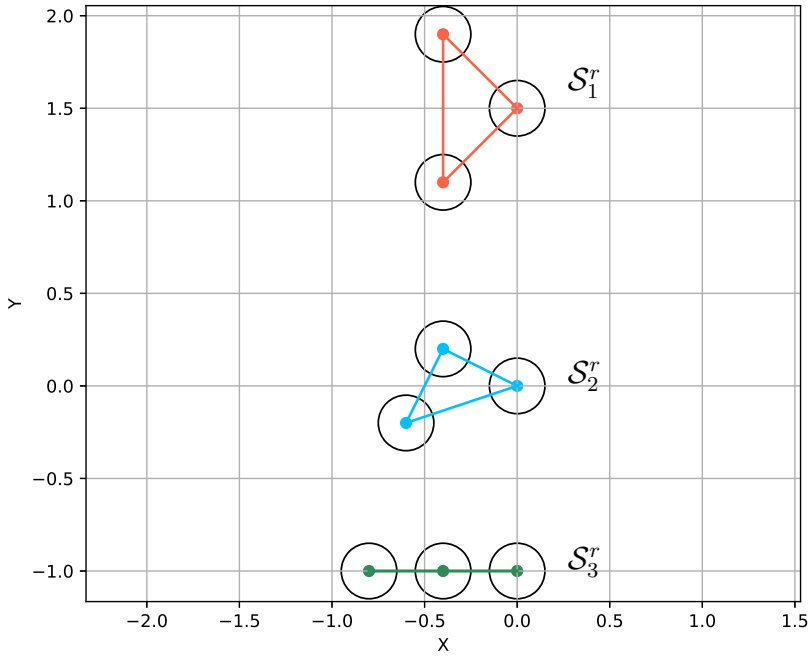

Figure 4.3: Three reference formations where $\mathcal{S}_1^r$ has the highest priority and $\mathcal{S}_3^r$ has the lowest priority.

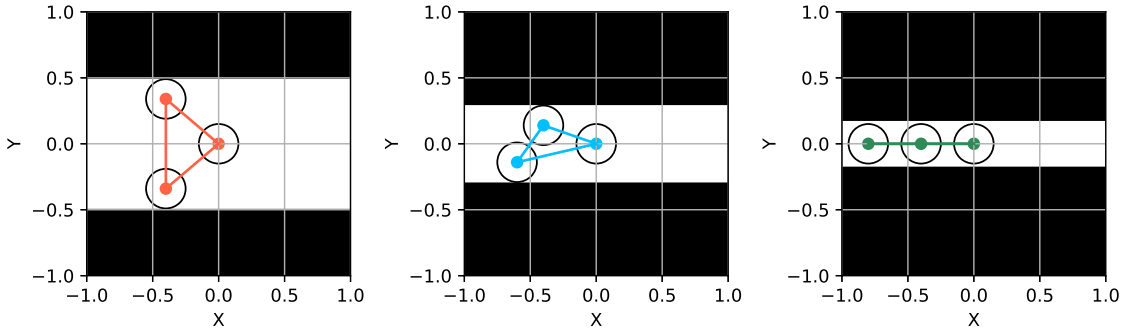

Figure 4.4: The three reference formations in Figure 4.3 can be used to find optimal formation in different scenarios.

Chapter 5

Collision Avoidance

5.1 Path Finding

The formation is defined with respect to the leader's expected position $\hat{\mathbf{q}}_1^t$ and is optimized by solving the optimization problem in (4.2.8). At the beginning of each iteration, $\hat{\mathbf{q}}_1^t$ is determined based on the operator's velocity command. The expected position is obtained using:

$$\hat{\mathbf{q}}_1^t = \mathbf{q}_1 + T_h \mathbf{v}_h \tag{5.1.1}$$

where $T_h \in \mathbb{R}^+$ is the time horizon used to predict and optimize the trajectories. This relies upon the fact that the operator would want to move the agents in the particular direction of $\mathbf{v}_h$ for the purpose of achieving the task objectives. It is reasonable to assume that the operator would guide the flock in a direction that is more or less obstacle free or feasible to move through. Moreover, equation (5.1.1) is used in optimization (4.2.8) and leader's target position $\mathbf{q}_1^t$ is re-calculated through

(4.2.9).

After determining the optimal formation through (4.2.8 - 4.2.11), the target positions of each individual agent would be uniquely determined using (4.2.10). All members of $\{\mathbf{q}_i^t\}$ would be in an obstacle-free region and the goal is to find a trajectory from current position $\mathbf{q}_i$ to the expected position $\mathbf{q}_i^t$. However, since the direct path between $\mathbf{q}_i$ and $\mathbf{q}_i^t$, denoted by $\mathcal{P}_i$, is not guaranteed to be collision-free, collision avoidance constraints cannot be directly derived based on these points. A path projection method is required to account for possible obstacles in the area.

Algorithm 2 is introduced to obtain an obstacle-free path based on the geometrical properties of obstacles in the environment. Initially, the direct line between $\mathbf{q}_i$ and $\mathbf{q}_i^t$ is considered as a candidate, and if it does not intersect with any of the obstacles, it would be selected as $\mathcal{P}_i$.

If that is not the case, the boundaries of each obstacle $\mathbf{bd}(\bar{\mathcal{O}})$ would be used to adjust the path. Algorithm 2 finds the path in a plane $\mathbf{a}_p^T\mathbf{x} = b_p$ passing through $\mathbf{p}_1$ and $\mathbf{p}_n$. The intersection of this plane and the colliding obstacle, would be a convex polygon $\mathcal{C}$ which can be represented by its vertices $\mathcal{C}_o = (\mathbf{c}_1, ..., \mathbf{c}_{o_c})$. Algorithm 2 links $\mathbf{p}_1$ and $\mathbf{p}_n$ by a feasible path based on a subset of $\mathcal{C}_o$. For any two points on a polygon, there exist two paths consisting of its vertices (line 12). After connecting the start and the destination points to these two paths, a refinement process is carried out on the given paths to make them shorter and remove the unnecessary vertices (line 15). For three-dimensional scenarios $d = 3$, this procedure is repeated across a discrete set of planes $\{(\mathbf{a}_p, b_p)\}$ which all contain $\mathbf{p}_1$ and $\mathbf{p}_2$. Ultimately, the shortest path is selected among all the paths in different intersecting planes.

As Algorithm 2 receives dilated versions of the obstacles, there would be a safe

Algorithm 2: Obstacle Boundary Guided Path Finding

Data: Start: $\mathbf{q}_i$, Goal: $\mathbf{q}_i^t$, Obstacles: $\{\bar{\mathcal{O}}_j\}$
Result: Feasible path $\mathcal{P}_i = (\mathbf{q}_i, \mathbf{p}_2, ..., \mathbf{p}_{n-1}, \mathbf{q}_i^t)$
$L(\mathbf{q}_i, \mathbf{q}_i^t) \leftarrow$ LineBetween2Points$(\mathbf{q}_i, \mathbf{q}_i^t)$;
if *IsCollisionFree*$(L(\mathbf{q}_i, \mathbf{q}_i^t))$ **then**
  $\mathcal{P}_i \leftarrow L(\mathbf{q}_i, \mathbf{q}_i^t)$
else
  $\{\bar{\mathcal{O}}_{c_j}\} \leftarrow$ FindCollidingObstacles$(L(\mathbf{q}_i, \mathbf{q}_i^t), \{\bar{\mathcal{O}}_j\})$;
  $\{(\mathbf{a}_p, b_p)\} \leftarrow$FindPlanesPassingThrough$(\mathbf{q}_i, \mathbf{q}_i^t)$;
  for $(\mathbf{a}_k, b_k) \in \{(\mathbf{a}_p, b_p)\}$ **do**
    $\mathcal{C}_o \leftarrow$ Intersection$((\mathbf{a}_k, b_k), \mathbf{bd}(\{\bar{\mathcal{O}}_{c_j}\}))$;
    $\mathbf{c}_s \leftarrow$ ClosestVertex$(\mathcal{C}_o, \mathbf{q}_i)$;
    $\mathbf{c}_e \leftarrow$ ClosestVertex$(\mathcal{C}_o, \mathbf{q}_i^t)$;
    $\mathcal{P}^{<1>}, \mathcal{P}^{<2>} \leftarrow$ TwoPossiblePaths$(\mathcal{C}_o, \mathbf{c}_s, \mathbf{c}_e)$;
    $\mathcal{P}^{<1>}, \mathcal{P}^{<2>} \leftarrow$ RefinePaths$(\mathcal{P}^{<1>}, \mathcal{P}^{<2>})$;
    $\mathcal{P}^{<1>} \leftarrow$AddVertices$(\mathcal{P}^{<1>}, \mathbf{q}_i, \mathbf{q}_i^t)$;
    $\mathcal{P}^{<2>} \leftarrow$AddVertices$(\mathcal{P}^{<2>}, \mathbf{q}_i, \mathbf{q}_i^t)$;
    $\mathcal{P}^{<1>}, \mathcal{P}^{<2>} \leftarrow$ RefinePaths$(\mathcal{P}^{<1>}, \mathcal{P}^{<2>})$;
    $\mathcal{P}_{i_k} \leftarrow$ PickShortestPath$(\mathcal{P}^{<1>}, \mathcal{P}^{<2>})$;
  end
  $\mathcal{P}_i \leftarrow$ PickShortestPath$(\{\mathcal{P}_{i_k}\})$;
end

distance between the computed path $\mathcal{P}_i$ and the original obstacles. Figure 5.1 demonstrates three examples of pathfinding in $\mathbb{R}^3$. The direct line between $\mathbf{q}_1$ and $\mathbf{q}_1^t$ is already inside $\mathcal{W}_f$. However, the paths between $(\mathbf{q}_2, \mathbf{q}_2^t)$ and $(\mathbf{q}_3, \mathbf{q}_3^t)$ are more complex due to the presence of obstacles which create a non-convex collision-free space between the start and end points.

The geometric paths obtained at this stage do not guarantee collision-free solutions since inter-agent collision was not considered during their computation. Moreover, they may not be dynamically feasible and can only be used as an starting point for what the future trajectory should be. The following section introduces suitable collision avoidance constraints to ensure the resulting agent trajectories comply with

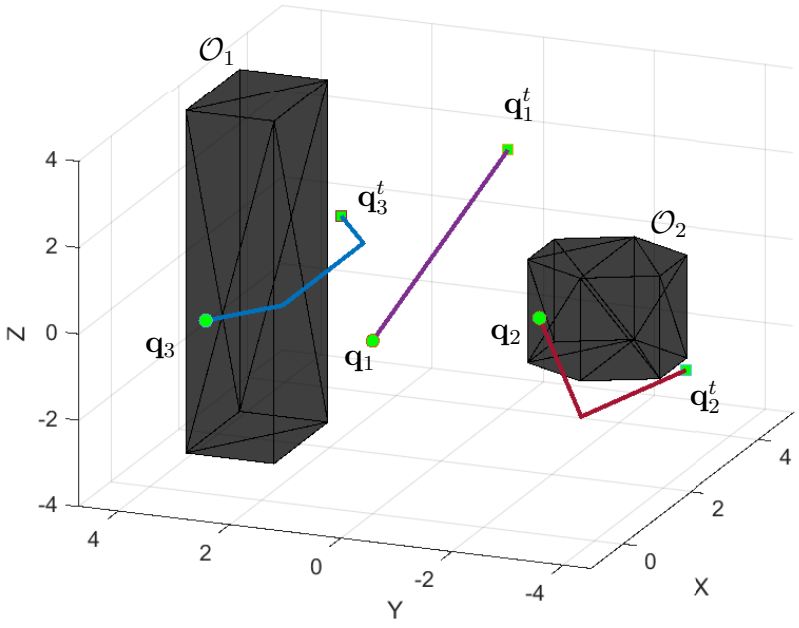

Figure 5.1: Three sample outputs of Algorithm 2 in three-dimensional space.

the goals outlined in the problem statement. Each path indicates the direction in which the agent should move to reach its temporarily-set goal $\mathbf{q}_i^t$ from its current position $\mathbf{q}_i$. This helps us consider only a smaller convex section of the generally non-convex $\mathcal{W}_f$ and enables us to optimize the trajectories using common practices of convex optimization.

5.2 Collision Avoidance Constraints

In this section, a method is presented to create convex constraints for collision avoidance around the results of the previous section. The general procedure is outlined in Algorithm 3. This algorithm is inspired by the solutions provided in [16] and [17], in which a convex polyhedral region $\mathcal{A}_i$ are generated along each path separately. Suitable inter-agent collision avoidance constraints would be generated for any two $\mathcal{A}_i$ and $\mathcal{A}_j$ that have a non-empty intersection. Finally, the path is refined to determine the next way-points position.

The paths generated by Algorithm 2 generally consist of multiple sections. This would require solving a time allocation problem for each section which is not ideal for real-time applications. Moreover, depending on the target position $\mathbf{q}_i^t$, Algorithm 2 may result in paths with non-reachable lengths within the time horizon. Consequently, each path $\mathcal{P} = (\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_{N_p})$ is truncated to its first segment as the next segments are considered in the future iterations. Additionally, the first segment is pruned to have a maximum length of $l_{max_i} \in \mathbb{R}^+$. This value accounts for the time horizon of the problem and the commanded velocity by the operator. Equation (5.2.1) shows how this value is computed.

$$l_{max_i} = \frac{T_h ||\mathbf{v}_h||^2}{|\mathbf{u}_i^T \mathbf{v}_h|} \tag{5.2.1}$$

where $\mathbf{u}_i$ is the unit vector in the direction of the first segment of the path (i.e. $\mathbf{p}_2 - \mathbf{p}_1$). This line segment is annotated by $\mathcal{P}_i^s = (\mathbf{p}_1^s, \mathbf{p}_2^s)$ as in Line 3 in Algorithm 3.

Algorithm 3: Collision Avoidance Constraints and Next Way-points Evaluation

Data: $\{\mathcal{P}_i\}$, $\{\bar{\mathcal{O}}_j\}$, $\{\mathcal{D}_i\}$
Result: Convex Regions: $\{\mathcal{A}_i\}$, Linear Constraints: $\mathcal{M}$, Next way-points:$\{\mathbf{q}_{n_i}\}$

1 $\mathcal{A}_i \leftarrow \emptyset$, $\mathcal{O}_i^c \leftarrow \{\bar{\mathcal{O}}_j\}$, $\mathbf{q}_{n_i} \leftarrow \mathbf{p}_{i_2}^s$ for all $i \in \mathcal{I}$;
2 **for** $i \in \mathcal{I}$ **do**
3 $\quad \mathcal{P}_i^s \leftarrow \text{LineSegment}(\mathcal{P}_i, l_{max})$;
4 $\quad (\mathbf{A}_{box}, \mathbf{b}_{box})_i \leftarrow \text{CreateBoundingBox}(\mathcal{P}_i^s)$;
5 $\quad (\mathbf{C}, \mathbf{d}) \leftarrow \text{EllispoidAroundLine}(\mathcal{P}_i^s)$;
6 $\quad \mathcal{A}_i \leftarrow \mathcal{A}_i \cap (\mathbf{A}_{box}, \mathbf{b}_{box})_i$;
7 $\quad$ **for** $\wr \in \{\bar{\mathcal{O}}_j\}$ **do**
8 $\quad\quad$ **if** $\wr \cap \mathcal{A}_i = \emptyset$ **then**
9 $\quad\quad\quad \mathcal{O}_i^c \leftarrow \mathcal{O}_i^c \setminus \wr$
10 $\quad\quad$ **end**
11 $\quad$ **end**
12 $\quad \mathcal{A}_i \leftarrow$AddSeparatingHyperPlanes$(\mathbf{C}, \mathbf{d}, \mathcal{O}_i^c)$;
13 **end**
14 $\{\bar{\mathcal{A}}_i\} \leftarrow \text{GetInflatedRegions}(\{\mathcal{A}_i\}, \{\mathcal{D}_i\})$;
15 $G_{coll} \leftarrow \text{CheckIntersections}(\{\bar{\mathcal{A}}_i\})$;
16 $(\{\mathcal{A}_i\}, \mathcal{M}) \leftarrow \text{InterAgentConstraints}(\{\mathcal{A}_i\}, G_{coll}, \{\mathcal{D}_i\})$;
17 $\{\mathbf{q}_{n_i}\} \leftarrow \text{EvaluateNextWayPoints}(\{\mathcal{A}_i\}, \{\mathcal{P}_i^s\})$;

5.2.1 Creating the Bounding Box

To reduce the search area and thus decrease the computation time, a bounding box is initialized around each segment $\mathcal{P}_i^s$. The idea is similar to [17] and helps concentate the convex region around the reference path and exclude the non-colliding and irrelevant obstacles (Line 4 in Algorithm 3). The box is represented by a tuple $(\mathbf{A}_{box}, \mathbf{b}_{box})_i$ which is basically a rectangular cube around $\mathcal{P}_i^s$:

$$(\mathbf{A}_{box}, \mathbf{b}_{box})_i \equiv \{\mathbf{x} \in \mathbb{R}^d | \mathbf{A}_{box_i}\mathbf{x} \leq \mathbf{b}_{box_i}\} \tag{5.2.2}$$

In case of $d = 3$, $\mathbf{A}_{box} = [\mathbf{a}_1| \ ... \ |\mathbf{a}_6]^T$ and $\mathbf{b}_{box} = [b_1 \ ... \ b_6]^T$. To compute

each hyper-plane $\mathbf{a}_k \in \mathbb{R}^3,\ k \in \{1, 2, ..., 6\}$, a local Cartesian coordinate (u, w, v) is assigned where $\mathbf{u}_i$ is aligned with $\mathcal{P}_i^s$. The resulting rectangular polyhedron has side lengths l_u, l_w and l_v. In this thesis, $l_u = ||\mathcal{P}_i^s||_2 + \epsilon$, and $l_w = l_v \geq \frac{\dot{q}_{max}^2}{u_{max}}$ [17]. $||\mathcal{P}^s||_2$ denotes the Euclidean length of the line segment $\mathcal{P}^s$, and $||\mathcal{P}^s||_2 \gg \epsilon \in \mathbb{R}^+$ is used to avoid numerical errors. Moreover, $\dot{q}_{max}$ and u_{max} refer to the maximum allowable value of velocity and acceleration, respectively. For $d = 2$, the bounding boxes can be computed by considering the same method over only two axes, which results in $\mathbf{A}_{box} = [\mathbf{a}_1|\mathbf{a}_2|\mathbf{a}_3|\mathbf{a}_4]^T$ with $\mathbf{a}_k \in \mathbb{R}^2,\ k \in \{1, 2, 3, 4\}$ and $\mathbf{b}_{box} = [b_1\ b_2\ b_3\ b_4]^T$.

5.2.2 Separating Hyper-Planes

The bounding box designed in the previous section may not be collision-free. A method similar to IRIS [16] is employed to compute separating hyper-planes from the obstacles. This approach requires an initial ellipsoid to find tangent planes onto the obstacles. An ellipsoid can be generally defined as:

$$\mathcal{E} = \{\mathbf{x} \in \mathbb{R}^d | (\mathbf{x} - \mathbf{d})^T \mathbf{P}^{-1} (\mathbf{x} - \mathbf{d}) \leq 1\} \tag{5.2.3}$$

where $\mathbf{d} \in \mathbb{R}^d$ is the center of the ellipsoid and $\mathbf{P} \succ 0$ is a symmetric positive-definite matrix. A second way to define $\mathcal{E}$ is by projecting the unit ball:

$$\mathcal{E} = \{\mathbf{C}\mathbf{x} + \mathbf{d} | \mathbf{x} \in \mathbb{R}^d, ||\mathbf{x}||_2 \leq 1, \mathbf{C} = \mathbf{P}^{\frac{1}{2}}\} \tag{5.2.4}$$

The approach in [16] involves an iterating approach to maximize the volume of the ellipsoid inscribed in the polyhedron followed by a quadratic optimization to find the separating hyper-planes. The IRIS algorithm does not generate convex regions

in a certain predefined direction, produces a large overlap between multiple regions in close proximity, and can be too slow for real-time computations because of its iterative nature. However, in this thesis, computation efficiency and the ability to generate convex regions in a specific direction (namely, $\mathcal{P}_i^s$) are important features. Therefore, the iterative procedure is skipped here and instead, a stretched ellipsoid is fitted in the direction of $\mathcal{P}_i^s$. Using this ellipsoid, the separating hyper-planes tend to be parallel with the given line segment. Hence, the designed ellipsoid must cover $\mathcal{P}_i^s$ along its longest axis. As a result, the center of the ellipsoid is placed in the middle of the corresponding line segment, i.e., $\mathbf{d} = \frac{1}{2}(\mathbf{p}_1^s + \mathbf{p}_2^s)$. The length of $\mathcal{E}$'s axes are proportional to eigenvalues of $\mathbf{P}$. In general, $\mathbf{P}$ can be decomposed as $\mathbf{P} = \mathbf{R}^T\mathbf{S}\mathbf{R}$ where $\mathbf{S}$ is diagonal and $\mathbf{R}$ is a rotation matrix.

Without loss of generality, for $d = 3$, it is assumed $\mathbf{S} = \mathbf{diag}(\lambda_1, \lambda_2, \lambda_3)$ where $\lambda_1 \geq \lambda_2 \geq \lambda_3$ are real positive numbers. The length of each axis is $2\sqrt{\lambda_k}$ for $k \in \{1, 2, 3\}$. Therefore, $\lambda_1 = \left(\frac{||\mathcal{P}_i^s||}{2}\right)^2$ is required to stretch the ellipsoid along the line $\mathcal{P}_i^s$. Moreover, the rotation matrix must satisfy the following condition:

$$\mathbf{u}_i = \mathbf{R}[1\ 0\ 0]^T \tag{5.2.5}$$

where $\mathbf{u}_i$ is the unit vector in the direction of $\mathcal{P}_i^s$. To shape the ellipsoid in the desired way, the second and third axes must be relatively small, i.e., $\lambda_1 \gg \lambda_2 \geq \lambda_3$. Similar approach can be used to compute the ellipsoid when $d = 2$ with less complexity. Lines 5 and 12 in Algorithm 3 define the ellipsoid and construct the separating hyper-planes.

Figure 5.2 shows an example of a bounding box (dashed line) around a line segment (in green). After considering the colliding obstacle, a separating plane tangent to the obstacle is calculated based on the stretched ellipsoid (in blue).

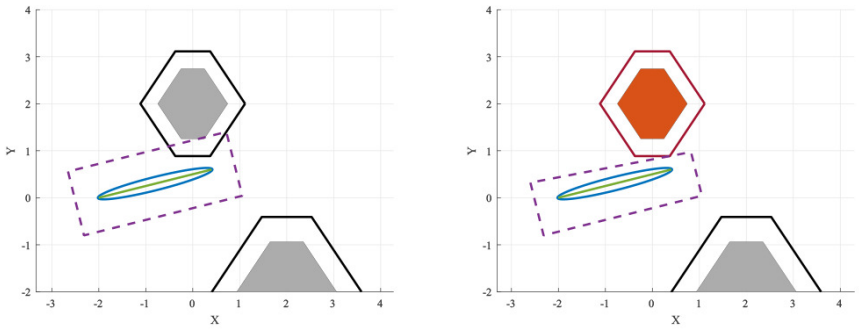

Figure 5.2: A simple 2-D demonstration of lines 4-12 in Algorithm 3. There are two obstacles and with their dilated boundary also depicted. The left image shows the initial bounding box and the stretched ellipsoid around the green line ($\mathcal{P}_i^s$). The right image depicts the detected colliding obstacle (red), and the separating plane that modifies the bounding box.

5.2.3 Inter-Agent Collision Avoidance

The polyhedra generated based on the method in Sections 5.2.1 and 5.2.2 are in obstacle-free space; nevertheless, inter-agent collisions are still possible. First, the possibility of collision (intersection) between any two convex regions $\mathcal{A}_i$ and $\mathcal{A}_k$ where $i \neq k \in \mathcal{I}$ is investigated. Next, suitable constraints for collision avoidance associated with each pair of agents are introduced.

Collision between a set of polyhedral regions $\{\mathcal{A}_i\}$ can be annotated by a undirected graph $G_{coll} = (V_r, E_{coll})$ where each $\mathcal{A}_i$ is assigned to $v_i \in V_r$ and each edge $(v_i, v_k) \in E_{coll}$ represents the collision between $\mathcal{A}_i$ and $\mathcal{A}_j$. Considering $\bar{\mathcal{A}}_i$ to be an inflated version of $\mathcal{A}_i$ by the radius of $\mathcal{D}_i$ similar to (3.3.5), if $\bar{\mathcal{A}}_i \cap \bar{\mathcal{A}}_k \neq \emptyset$, then a collision is detected (Line 14). Line 15 in Algorithm 3 computes G_{coll} based on the following:

$$(v_i, v_k) \in E_{coll} \iff \bar{\mathcal{A}}_i \cap \bar{\mathcal{A}}_k \neq \emptyset \tag{5.2.6}$$

For any $(v_i, v_k) \in E_{coll}$, where $||P_i^s||_2 \leq ||P_k^s||_2$, six different relaxations of the minimum distance constraint can be considered.

- **Priority 1**:

$$\mathbf{u}_i^T (\mathbf{q}_i - \mathbf{q}_k) \geq d_{ik} \ \vee \ \mathbf{u}_i^T (\mathbf{q}_i - \mathbf{q}_k) \leq -d_{ik} \tag{5.2.7}$$

- **Priority 2**:

$$\mathbf{u}_k^T (\mathbf{q}_i - \mathbf{q}_k) \geq d_{ik} \ \vee \ \mathbf{u}_k^T (\mathbf{q}_i - \mathbf{q}_k) \leq -d_{ik} \tag{5.2.8}$$

- **Priority 3**: Considering $\mathbf{a}_{sp}^T \mathbf{x} = b_{sp}$ as the bisector hyper-plane between $\mathbf{q}_i$ and $\mathbf{q}_k$, two parallel hyper-planes forming a slab with thickness d_{ik} and its center-plane on the bisector can be added to $\mathcal{A}_i$ and $\mathcal{A}_k$, separately. This usually is a more conservative constraint than equation (5.2.7) or (5.2.8), however, it guarantees the minimum distance of d_{ik} between two polytopes $\mathcal{A}_i$ and $\mathcal{A}_k$.

Depending on the current positions of the agents, only one of the constraints is applicable. Thus, based on feasibility and priority, one of the constraints is chosen to help avoid inter-agent collision. It should be noted that the agents are assumed not to be in collision at the beginning of the procedure ($t = 0$) and $||\mathbf{q}_i - \mathbf{q}_k|| \geq d_{ik}$.

The inter-agent constraints are divided into two categories. One is the splitting planes which result in the updated version of $\{\mathcal{A}_i\}$. The other is a set of linear inequalities where each, for two agents $i \neq k \in \mathcal{I}$ at time step κ can be represented as follows:

$$\begin{bmatrix} \mathbf{a}_{m_t} \\ \mathbf{a}_{m_r} \end{bmatrix}^T \begin{bmatrix} \mathbf{q}_t[\kappa] \\ \mathbf{q}_r[\kappa] \end{bmatrix} \leq b_{m_{tr}} \tag{5.2.9}$$

where $\mathbf{a}_{m_i}, \mathbf{a}_{m_k} \in \mathbb{R}^d$ and $b_{m_{ik}} \in \mathbb{R}$ obtained from either (5.2.7) or (5.2.8). Each of these equations will be stored in $\mathcal{M}$ as a tuple $\mathfrak{m} = (i, k, \kappa, \mathbf{a}_{m_i}, \mathbf{a}_{m_k}, b_{m_{ik}})$ as per Line

16 in Algorithm 3.

5.2.4 Next Way-Points

The finalized convex regions calculated by adding necessary collision constraints (line 16) may result in infeasible way-points. In other words, for initially-set next way-points, $\mathbf{q}_{n_i} = \mathbf{p}^s_{i_2}$, $\mathbf{q}_{n_i} \in \mathcal{A}_i$ may not be true anymore due to the possibility of adding bisector hyper-planes. By trimming the extra part of line $\mathcal{P}^s_i$, the closest point on it to $\mathbf{p}^s_{i_2}$ which also satisfies $\mathbf{q}_{n_i} \in \mathcal{A}_i$ would be used as the next way-point. Mathematically,

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^d}{\text{minimize}} \quad & ||\mathbf{x} - \mathbf{p}^s_{i_2}|| \\ \text{subject to} \quad & \mathbf{x} \in \mathcal{A}_i \\ & \mathbf{x} = \alpha \mathbf{p}^s_{i_1} + (1-\alpha)\mathbf{p}^s_{i_1} \\ & 0 \leq \alpha \leq 1 \end{aligned}$$

would provide the next way-point $\mathbf{q}_{n_i}$. However, this can be computed geometrically by obtaining the intersection between polyhedron hyper-planes of $\mathcal{A}_i$ and $\mathcal{P}^s_i$. This is computationally efficient and can be solved in linear time. Line 17 in Algorithm 3 is the final step for computing the next way-points.

Chapter 6

Model-based Optimization

As stated in Section 1.2, the goal is to generate collision-free trajectories for each agent. This can be expressed as an optimization problem. In this thesis, the problem is formulated as a Model Predictive Controller (MPC) with a Receding Horizon. In this approach at each time-step, a sequence of control inputs is obtained by solving an optimization problem over a receding control horizon T_h. The constraints of the optimization problem include the vehicles' dynamics, actuation limits, and collision-related constraints due to obstacles and/or other agents in the task environment. The computations are carried out centrally assuming all agents communicate their data with a central computation unit. In the following sections, the optimization formulation for a 3-dimensional case ($d = 3$) will be presented. The 2-dimensional case can be derived similarly by disregarding the third dimension.

6.1 Dynamic Model-Related Constraints

Equations (3.3.2) and (3.3.3) yield the following discrete-time state-space equations for each agent:

$$\hat{\mathbf{x}}[k+1] = \mathbf{A}_d \hat{\mathbf{x}}[k] + \mathbf{B}_d \mathbf{u}[k] \tag{6.1.1}$$

where the state vector is $\hat{\mathbf{x}} = [q_x \ \dot{q}_x \ q_y \ \dot{q}_y \ q_z \ \dot{q}_z]^T$ and k is the time step. The system dynamic matrices will be:

$$\mathbf{A}_d = \begin{bmatrix} \mathbf{A}_{d_s} & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{A}_{d_s} & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{A}_{d_s} \end{bmatrix}, \ \mathbf{B}_d = \begin{bmatrix} \mathbf{B}_{d_s} & \bar{\mathbf{0}}_2 & \bar{\mathbf{0}}_2 \\ \bar{\mathbf{0}}_2 & \mathbf{B}_{d_s} & \bar{\mathbf{0}}_2 \\ \bar{\mathbf{0}}_2 & \bar{\mathbf{0}}_2 & \mathbf{B}_{d_s} \end{bmatrix} \tag{6.1.2}$$

where $\bar{\mathbf{0}}_2$ is a two-element zero vector and $\mathbf{0}_2$ is a square 2×2 zero matrix. Matrices $\mathbf{A}_{d_s}$ and $\mathbf{B}_{d_s}$ are also defined as follows:

$$\mathbf{A}_{d_s} = \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix}, \ \mathbf{B}_{d_s} = \frac{1}{2m_i} \begin{bmatrix} \Delta T^2 \\ 2\Delta T \end{bmatrix} \tag{6.1.3}$$

The number of steps in the prediction horizon is $K \in \mathbb{N}$. For notation simplicity, the initial state is denoted by $\mathbf{x}_0 = \mathbf{x}[0]$, and the agent subscript $i \in \mathcal{I}$ is dropped. Using equation (6.1.1), the relation between the sequence of control inputs $\underline{\mathbf{u}}$ and state vectors $\underline{\hat{\mathbf{x}}}$ can be defined as following:

$$\underline{\hat{\mathbf{x}}} = \underline{\mathbf{A}}_d \mathbf{x}_0 + \underline{\mathbf{B}}_d \ \underline{\mathbf{u}} \tag{6.1.4}$$

where

$$\underline{\hat{\mathbf{x}}} = \begin{bmatrix} \hat{\mathbf{x}}[1] \\ \hat{\mathbf{x}}[2] \\ \vdots \\ \hat{\mathbf{x}}[K] \end{bmatrix}, \ \underline{\mathbf{u}} = \begin{bmatrix} \mathbf{u}[0] \\ \mathbf{u}[1] \\ \vdots \\ \mathbf{u}[K-1] \end{bmatrix}, \ \underline{\mathbf{A}}_d = \begin{bmatrix} \mathbf{A}_d \\ \mathbf{A}_d^2 \\ \vdots \\ \mathbf{A}_d^K \end{bmatrix} \tag{6.1.5}$$

$$\underline{\mathbf{B}}_d = \begin{bmatrix} \mathbf{B}_d & \mathbf{0}_{6\times3} & \mathbf{0}_{6\times3} & \dots & \mathbf{0}_{6\times3} \\ \mathbf{A}_d\mathbf{B}_d & \mathbf{B}_d & \mathbf{0}_{6\times3} & \dots & \mathbf{0}_{6\times3} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{A}_d^{K-1}\mathbf{B}_d & \mathbf{A}_d^{K-2}\mathbf{B}_d & \dots & \mathbf{A}_d\mathbf{B}_d & \mathbf{B}_d \end{bmatrix} \tag{6.1.6}$$

where $\mathbf{0}_{6\times3}$ defines a 6×3 dimensional zero matrix. An important early result of equation (6.1.4) is that the predicted states $\underline{\hat{\mathbf{x}}}$ can be expressed as an affine function of control inputs. This enables us to re-write the constraints of states (i.e. positions and velocities) in terms of a linear combination of control inputs $\underline{\mathbf{u}}$.

Finally, for a team of robots with individual dynamics represented by (6.1.1 - 6.1.4), the dynamics for the whole system can be written as:

$$\underline{\hat{\mathbf{X}}} = \mathbf{\Phi}\underline{\mathbf{X}}_0 + \mathbf{\Theta}\underline{\mathbf{U}} \tag{6.1.7}$$

where $\underline{\hat{\mathbf{X}}} \in \mathbb{R}^{6KN}$, $\underline{\mathbf{X}}_0 \in \mathbb{R}^{6N}$, and $\underline{\mathbf{U}} \in \mathbb{R}^{3KN}$ are obtained by stacking $\underline{\hat{\mathbf{x}}}_i$'s, $\underline{\mathbf{x}}_{0_i}$'s, and $\underline{\mathbf{u}}_i$'s on top of each other, respectively. Furthermore, $\mathbf{\Phi} = \mathbf{diag}(\underline{\mathbf{A}}_{d_1}, ..., \underline{\mathbf{A}}_{d_N})$ and similarly, $\mathbf{\Theta} = \mathbf{diag}(\underline{\mathbf{B}}_{d_1}, ..., \underline{\mathbf{B}}_{d_N})$ where **diag** is defined in Appendix Section A.1.

6.2 Objective Function

The optimization is a minimization problem with an objective function which is composed of four parts: velocity error, control effort, final-state cost, and formation cost.

6.2.1 Velocity Error

This component aims to minimize the deviation of each robot's velocity from the reference command provided by the human operator. The error penalty is defined as a quadratic function:

$$J_v = \sum_{i\in\mathcal{I}} \sum_{k=1}^{k=K} (\dot{\mathbf{q}}_i[k] - \mathbf{v}_h)^T (\dot{\mathbf{q}}_i[k] - \mathbf{v}_h) \tag{6.2.1}$$

By defining $\mathbf{I}_v$ as in Appendix A.2, $\underline{\mathbf{I}}_v = \mathbf{diag}(\mathbf{I}_v, ..., \mathbf{I}_v)$ resulting in $\underline{\mathbf{I}}_v \in \mathbb{R}^{3KN\times 6KN}$, and by concatenating the reference velocity vectors $\underline{\mathbf{V}}_h = [\mathbf{v}_h^T \dots \mathbf{v}_h^T]^T \in \mathbb{R}^{3KN}$, equation (6.2.1) can be reformulated as:

$$J_v(\underline{\mathbf{U}}) = \frac{1}{2}\underline{\mathbf{U}}^T \mathbf{Q}_v \underline{\mathbf{U}} + \mathbf{f}_v^T \underline{\mathbf{U}} + r_v \tag{6.2.2}$$

where:

$$\mathbf{Q}_v = 2\mathbf{\Theta}^T \underline{\mathbf{I}}_v^T \underline{\mathbf{I}}_v \mathbf{\Theta} \tag{6.2.3}$$

$$\mathbf{f}_v = 2\mathbf{\Theta}^T \underline{\mathbf{I}}_v^T \underline{\mathbf{I}}_v \mathbf{\Phi} \underline{\mathbf{X}}_0 - 2\mathbf{\Theta}^T \underline{\mathbf{I}}_v^T \underline{\mathbf{V}}_h \tag{6.2.4}$$

$$r_v = \underline{\mathbf{X}}_0^T \mathbf{\Phi}^T \underline{\mathbf{I}}_v^T \underline{\mathbf{I}}_v \mathbf{\Phi} \underline{\mathbf{X}}_0 - 2\underline{\mathbf{V}}_h^T \underline{\mathbf{I}}_v \mathbf{\Phi} \underline{\mathbf{X}}_0 + \underline{\mathbf{V}}_h^T \underline{\mathbf{V}}_h \tag{6.2.5}$$

6.2.2 Control Effort Cost

The following quadratic cost term is added to lower the control effort:

$$J_u(\underline{\mathbf{U}}) = \frac{1}{2}\underline{\mathbf{U}}^T \mathbf{Q}_u \underline{\mathbf{U}} \tag{6.2.6}$$

where $\mathbf{Q}_u = \mathbf{diag}(\mathbf{R}_u, ..., \mathbf{R}_u)$, $0 \prec \mathbf{Q}_u \in \mathbb{R}^{3KN \times 3KN}$, and $\mathbf{R}_u \in \mathbb{R}^{3\times 3}$ is the penalty weight matrix for control input vectors.

6.2.3 Final State Cost

Based on the information extracted from the environment in previous sections, the agents can be guided using the desired final state at the end of the prediction horizon. Letting $\mathbf{x}_{n_i} \in \mathbb{R}^6$ be the desired final state for the i^{th} vehicle, a quadratic cost function can be defined as follows:

$$J_f = \sum_{i\in\mathcal{I}} (\hat{\mathbf{x}}_i[K] - \mathbf{x}_{n_i})^T (\hat{\mathbf{x}}_i[K] - \mathbf{x}_{n_i}) \tag{6.2.7}$$

Knowing that $\hat{\mathbf{x}}[K] = \mathbf{I}_f \underline{\hat{\mathbf{x}}}$ where $\mathbf{I}_f = [\mathbf{0}_{6\times 6(K-1)}\ \mathbf{I}_6]$ and $\mathbf{I}_6$ is a 6×6 identity matrix, the objective function in (6.2.7) can be written in terms of the control inputs $\underline{\mathbf{U}}$:

$$J_f(\underline{\mathbf{U}}) = \frac{1}{2}\underline{\mathbf{U}}^T \mathbf{Q}_f \underline{\mathbf{U}} + \mathbf{f}_f^T \underline{\mathbf{U}} + r_f \tag{6.2.8}$$

where $\mathbf{Q}_f$, $\mathbf{f}_f$, and r_f are calculated as follows:

$$\mathbf{Q}_f = 2\boldsymbol{\Theta}^T \underline{\mathbf{I}}_f^T \underline{\mathbf{I}}_f \boldsymbol{\Theta} \tag{6.2.9}$$

$$\mathbf{f}_f = 2\boldsymbol{\Theta}^T \underline{\mathbf{I}}_f^T \underline{\mathbf{I}}_f \boldsymbol{\Phi} \underline{\mathbf{X}}_0 - 2\boldsymbol{\Theta}^T \underline{\mathbf{I}}_f^T \underline{\mathbf{X}}_n \tag{6.2.10}$$

$$r_f = \underline{\mathbf{X}}_0^T \boldsymbol{\Phi}^T \underline{\mathbf{I}}_f^T \underline{\mathbf{I}}_f \boldsymbol{\Phi} \underline{\mathbf{X}}_0 - 2\underline{\mathbf{X}}_n^T \underline{\mathbf{I}}_f \boldsymbol{\Phi} \underline{\mathbf{X}}_0 + \underline{\mathbf{X}}_n^T \underline{\mathbf{X}}_n \tag{6.2.11}$$

where $\underline{\mathbf{I}}_f = \mathbf{diag}(\mathbf{I}_f, \mathbf{I}_f, ..., \mathbf{I}_f) \in \mathbb{R}^{6N \times 6KN}$ is a block-diagonal matrix. In addition, $\underline{\mathbf{X}}_n \in \mathbb{R}^{6N}$ is the vertical concatenation of all $\mathbf{x}_{n_i}$'s.

Desired Final State

The final state vector $\mathbf{x}_{n_i}$ is composed of position and velocity values. The position variables can be determined according to the evaluated next way-points in Subsection 5.2.4. Each way-point $\mathbf{q}_{n_i}$ lies on $\mathcal{P}_i$, and the direction of movement at $\mathbf{q}_{n_i}$ along its associated path $\mathbf{e}_{n_i} \in \mathbb{R}^3$ is the desired direction for each agent. Thus, assuming the desired velocity at this way-point $\dot{\mathbf{q}}_{n_i}$ to be:

$$\dot{\mathbf{q}}_{n_i} = \alpha_i (\mathbf{v}_h^T \mathbf{e}_{n_i}) \mathbf{e}_{n_i} \tag{6.2.12}$$

where $\alpha_i \in \mathbb{R}$, will lead to

$$\mathbf{x}_{n_i} = \mathbf{I}_p^T \mathbf{q}_{n_i} + \mathbf{I}_v^T \dot{\mathbf{q}}_{n_i} \tag{6.2.13}$$

where the definition of $\mathbf{I}_p$ and $\mathbf{I}_v$ can be found in Appendix A.2. Moreover, the velocity coefficients α_i can be determined analytically by minimizing the following function:

$$f = (\sum \dot{\mathbf{q}}_i - n\mathbf{v}_h)^T(\sum \dot{\mathbf{q}}_i - n\mathbf{v}_h) \tag{6.2.14}$$

$$= \sum \dot{\mathbf{q}}_i^T\dot{\mathbf{q}}_i + 2\sum_{i\neq j} \dot{\mathbf{q}}_i^T\dot{\mathbf{q}}_j - 2n(\sum \mathbf{v}_h^T\dot{\mathbf{q}}_i) + n^2\mathbf{v}_h^T\mathbf{v}_h \tag{6.2.15}$$

Minimizing f with respect to α_i results the following system of equations:

$$\mathbf{A}_\alpha \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = \mathbf{b}_\alpha \implies \begin{bmatrix} a_{ij} \end{bmatrix} \bar{\alpha} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} \tag{6.2.16}$$

where

$$\begin{aligned} a_{ii} &= (\mathbf{v}_h^T\mathbf{e}_{n_i})^2 \\ a_{ij} &= (\mathbf{v}_h^T\mathbf{e}_{n_i})(\mathbf{v}_h^T\mathbf{e}_{n_j})(\mathbf{e}_{n_i}^T\mathbf{e}_{n_j}) \\ b_i &= n(\mathbf{v}_h^T\mathbf{e}_{n_i})^2 \end{aligned} \tag{6.2.17}$$

Finally,

$$\bar{\alpha} = (\mathbf{A}_\alpha^T\mathbf{A}_\alpha)^{-1}\mathbf{A}_\alpha^T\mathbf{b}_\alpha \tag{6.2.18}$$

6.2.4 Formation Cost

Inasmuch as minimum deviation from the reference formation is desired, the following term is introduced to control the shape of the flock over the prediction horizon. Given a formation $S = (\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, ..., \mathbf{s}_N)$ with $\mathbf{s}_1 = \mathbf{0}$, deviation can be expressed as $\sum_{i\neq j}((\mathbf{q}_i - \mathbf{q}_j) - (\mathbf{s}_i - \mathbf{s}_j))^2$. In the same fashion, it can be re-written in terms of

control inputs as follows:

$$\mathbf{J}_s = \frac{1}{2}\underline{\mathbf{U}}^T \mathbf{Q}_s \underline{\mathbf{U}} + \mathbf{f}_s^T \underline{\mathbf{U}} + r_s \tag{6.2.19}$$

where

$$\mathbf{Q}_s = \mathbf{\Theta}^T \underline{\mathbf{I}}_p^T \mathbf{\Lambda}_s \underline{\mathbf{I}}_p \mathbf{\Theta} \tag{6.2.20}$$

$$\mathbf{f}_s = \mathbf{\Theta}^T \underline{\mathbf{I}}_p^T \mathbf{\Lambda}_s \underline{\mathbf{I}}_p \mathbf{\Phi} \underline{\mathbf{X}}_0 + \mathbf{\Theta}^T \underline{\mathbf{I}}_p^T \underline{\lambda}_s \tag{6.2.21}$$

$$r_s = \sum_k \sum_i \sum_{j \neq i} \delta \mathbf{s}_{ij}^T \delta \mathbf{s}_{ij} + \frac{1}{2} \underline{\mathbf{X}}_0^T \mathbf{\Phi}^T \underline{\mathbf{I}}_p^T \mathbf{\Lambda}_s \underline{\mathbf{I}}_p \mathbf{\Phi} \underline{\mathbf{X}}_0 + \underline{\lambda}_s^T \underline{\mathbf{I}}_p \mathbf{\Phi} \underline{\mathbf{X}}_0 \tag{6.2.22}$$

in which $\mathbf{\Lambda}_s$ is a square matrix with $(3 \times K \times N)$ rows as following:

$$\mathbf{\Lambda}_s = 2 \times \begin{bmatrix} 2(N-1)\mathbf{I}_s & -2\mathbf{I}_s & \dots & -2\mathbf{I}_s \\ -2\mathbf{I}_s & 2(N-1)\mathbf{I}_s & \dots & -2\mathbf{I}_s \\ \vdots & \vdots & \ddots & \vdots \\ -2\mathbf{I}_s & -2\mathbf{I}_s & \dots & 2(N-1)\mathbf{I}_s \end{bmatrix} \tag{6.2.23}$$

and

$$\underline{\lambda}_s = -4 \times \begin{bmatrix} \sum_j \delta \mathbf{s}_{1j} \\ \vdots \\ \sum_j \delta \mathbf{s}_{1j} \\ \sum_j \delta \mathbf{s}_{2j} \\ \vdots \\ \sum_j \delta \mathbf{s}_{Nj} \end{bmatrix}_{(3KN)\times 1} \tag{6.2.24}$$

where $\delta \mathbf{s}_{ij} = \mathbf{s}_i - \mathbf{s}_j$ and $\mathbf{I}_s$ is the identity matrix with $3K$ rows.

6.3 Constraints

There are four categories of constraints that limit each vehicle's movement: 1) actuation limits, 2) continuity of control inputs, 3) obstacle-free convex regions of states, and 4) inter-agent collision avoidance constraints. They all will be formulated as linear convex inequality constraints in the subsequent sections.

6.3.1 Actuation Maximum Levels

Each vehicle's input range must lie inside the actuator's physical limitation which can be considered as:

$$\mathbf{U}_{min} \leq \underline{\mathbf{U}} \leq \mathbf{U}_{max} \tag{6.3.1}$$

where $\mathbf{U}_{min}, \mathbf{U}_{max} \in \mathbb{R}^{3KN}$ are stacked vector of each vehicle's actuation limits.

6.3.2 Continuity of Control Inputs

To preserve the continuity and smoothness of the control input signal, the variation of consecutive values of $\mathbf{u}$ must not exceed a certain threshold. Therefore,

$$-\begin{bmatrix} \delta u^x \\ \delta u^y \\ \delta u^z \end{bmatrix} \leq \mathbf{u}[k+1] - \mathbf{u}[k] \leq \begin{bmatrix} \delta u^x \\ \delta u^y \\ \delta u^z \end{bmatrix} \tag{6.3.2}$$

where $\delta u^e \in \mathrm{R}^+$, for $e = \{x, y, z\}$. The constraint in (6.3.2) can be written for all the robots and for $k = -1, 0, ..., K-2$ where $k = -1$ is reserved for the last time step of the previous iteration. By concatenating all the equations together, the linear

inequality in the following form is derived:

$$\mathbf{A}_{cont}\underline{\mathbf{U}} \leq \mathbf{b}_{cont} \tag{6.3.3}$$

where $\mathbf{A}_{cont} \in \mathbb{R}^{6(K-1)N \times 3KN}$ and $\mathbf{b}_{cont} \in \mathbb{R}^{6(K-1)N}$.

6.3.3 Convex Regions

Convex obstacle-free polyhedral regions generated by Algorithm 3 $\{\mathcal{A}_i\}$ can be represented by a linear inequality:

$$\mathbf{q}_i \in \mathcal{A}_i \iff \mathbf{A}_{c_i}\mathbf{q}_i \leq \mathbf{b}_{c_i} \tag{6.3.4}$$

For each single agent at one step, RHS of (6.3.4) can be written as $\mathbf{A}_{c_i}\mathbf{I}_p\mathbf{x}_i \leq \mathbf{b}_{c_i}$. In terms of $\underline{\mathbf{x}}_i$, the constraint will be $\underline{\mathbf{A}}_{c_i}\underline{\mathbf{x}}_i \leq \underline{\mathbf{b}}_{c_i}$ where $\underline{\mathbf{A}}_{c_i} = \mathbf{diag}(\mathbf{A}_{c_i}\mathbf{I}_p, ..., \mathbf{A}_{c_i}\mathbf{I}_p)$ and $\underline{\mathbf{b}}_{c_i}^T = [\mathbf{b}_{c_i}^T, ..., \mathbf{b}_{c_i}^T]$. Similarly, in terms of $\underline{\mathbf{X}}$, all constraints can be concatenated as $\mathbf{\Psi}_c\underline{\mathbf{X}} \leq \mathbf{\Gamma}_c$ where

$$\mathbf{\Psi}_c = \begin{bmatrix} \underline{\mathbf{A}}_{c_1} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \underline{\mathbf{A}}_{c_2} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \dots & \underline{\mathbf{A}}_{c_N} \end{bmatrix}, \ \mathbf{\Gamma}_c = \begin{bmatrix} \underline{\mathbf{b}}_{c_1} \\ \underline{\mathbf{b}}_{c_2} \\ \vdots \\ \underline{\mathbf{b}}_{c_N} \end{bmatrix} \tag{6.3.5}$$

As a result, using (6.1.7), we can write obstacle-free regions constraints in terms of the decision vector $\underline{\mathbf{U}}$ as follows:

$$\mathbf{H}_c\underline{\mathbf{U}} \leq \mathbf{z}_c \tag{6.3.6}$$

where $\mathbf{H}_c = \mathbf{\Psi}_c\mathbf{\Theta}$ and $\mathbf{z}_c = \mathbf{\Gamma}_c - \mathbf{\Psi}_c\mathbf{\Phi}\underline{\mathbf{X}}_0$.

6.3.4 Inter-Agent Collision Avoidance

These constraints are generated from Algorithm 3 as explained in Subsection 5.2.3. Without loss of generality, we can assume $r > t$, and thus, the inequality in (5.2.9) can be written in a more general form as:

$$\begin{bmatrix} \mathbf{0}_{i_1} & \mathbf{a}_{m_t}^T & \mathbf{0}_{i_2} & \mathbf{a}_{m_r}^T & \mathbf{0}_{i_3} \end{bmatrix} \begin{bmatrix} \mathbf{q}_1[1] \\ \vdots \\ \mathbf{q}_1[K] \\ \mathbf{q}_2[1] \\ \vdots \\ \mathbf{q}_N[K] \end{bmatrix} \leq b_{m_{tr}} \tag{6.3.7}$$

where $i_1 = 1 \times (3K(t-1) + 3(k-1))$, $i_2 = 1 \times (3K(r-t) - 3)$, and $i_3 = 1 \times (3K(N - r + 1) - 3k)$ are the dimensions of corresponding zero row-vectors. For all $\mathfrak{m} \in \mathcal{M}$, constraints can be re-written in the form of inequality in (6.3.7) and stacked together. This in terms of all state vectors is formulated as:

$$\mathbf{A}_m\underline{\mathbf{I}}_p\hat{\underline{\mathbf{X}}} \leq \mathbf{b}_m \tag{6.3.8}$$

where $\underline{\mathbf{I}}_p = \mathbf{diag}(\mathbf{I}_p, ..., \mathbf{I}_p)$, $\mathbf{A}_m \in \mathbb{R}^{|\mathcal{M}| \times 3KN}$, and $\mathbf{b}_m \in \mathbb{R}^{|\mathcal{M}|}$. Finally, using (6.1.7) and (6.3.8), the inter-agent collision avoidance constraints can be expressed in terms of the decision variables as:

$$\mathbf{H}_m\underline{\mathbf{U}} \leq \mathbf{z}_m \tag{6.3.9}$$

where $\mathbf{H}_m = \mathbf{A}_m \underline{\mathbf{I}}_p \mathbf{\Theta}$ and $\mathbf{z}_m = \mathbf{b}_m - \mathbf{A}_m \underline{\mathbf{I}}_p \mathbf{\Phi} \underline{\mathbf{X}}_0$.

6.4 Convex Optimization Problem

The optimal control inputs for the system stated in (6.1.7) is obtained by solving the following quadratic programming optimization:

$$\begin{aligned} \underset{\underline{\mathbf{U}}}{\text{minimize}} \quad & w_v J_v + w_u J_u + w_f J_f + w_s J_s \\ \text{subject to} \quad & \mathbf{U}_{min} \leq \underline{\mathbf{U}} \leq \mathbf{U}_{max} \\ & \mathbf{A}_{cont} \underline{\mathbf{U}} \leq \mathbf{b}_{cont} \\ & \mathbf{H}_c \underline{\mathbf{U}} \leq \mathbf{z}_c \\ & \mathbf{H}_m \underline{\mathbf{U}} \leq \mathbf{z}_m \end{aligned} \tag{6.4.1}$$

where $w_v, w_f, w_u, w_s \in \mathbb{R}^+$ are scalarization parameters. The optimization problem (6.4.1) has $3KN$ decision variables and the number of constraints is a variant of convex regions and inter-agent collision detection algorithm.

Chapter 7

Simulation and Experimental Results

The various components of the proposed control framework for human-in-the-loop operation of a group of drones, seen in Figure 3.1, were described in the previous chapters of this thesis. This chapter focuses on the implementation of the control strategy and presents results from simulations and experiments in a laboratory environment. The first section describes the simulation setup and also explores two operational scenarios. The second half of the chapter presents steps taken to implement the algorithm for real-time control, and presents the results of the experiments.

7.1 Controllers

7.1.1 Flock Velocity Controller

Although the length of the paths is bounded by a function of T_h based on equation (5.2.1), it does not necessarily result in an optimal path length due to other factors. For instance, the way-points generated in Section 5.2.4 and their distance from the initial position are impacted by several parameters, such as the geometrical shape of the obstacles or the constraints generated in Section 6.3. Moreover, the time horizon T_h in the trajectory optimization stage is fixed. Along with the varying path length, this might result in a deviation from the reference velocity while the time required to traverse between the initial position and the next way-point depends on both of them (path lengths and average velocity). This issue can be addressed by solving multiple optimization problems with different time horizons or having a separate optimization to find the best horizon time. Nevertheless, these approaches are computationally expensive and not well-suited for real-time implementation.

In an alternative solution, the reference velocity used in velocity error cost (6.2.1) can be modified in real-time. The reference velocity $\mathbf{v}_h$ used in section 6.2 can be regulated with a conventional PI (proportional integral) controller. Accordingly, it can be formulated as follows:

$$\mathbf{v}_h := \mathbf{v}_h^{ref} + K_P \left(\mathbf{v}_h^{ref} - \mathbf{v}_c \right) + K_I \int \left(\mathbf{v}_h^{ref} - \mathbf{v}_c \right) \tag{7.1.1}$$

where $\mathbf{v}_h^{ref} \in \mathbb{R}^d$ is the reference velocity received from the operator, $\mathbf{v}_c \in \mathbb{R}^d$ is the velocity of center of the flock, and $K_I, K_P \in \mathbb{R}^d$ are positive definite matrices. In addition, to avoid undesirable and sudden changes in the direction, K_I, K_P will be

tuned to result in an under-damped response.

7.1.2 Trajectory Controller

The control inputs ($\{\mathbf{u}_i\}$) computed by the result of optimization (6.4.1) can be translated as trajectories based on the state equation in Equation (6.1.1). In a real-time application, the first few steps of these optimal trajectories would be continuously appended to the existing reference trajectories. A controller is needed to ensure the quad-rotors follow the regularly-updated trajectories.

The controller used in this thesis is the same as the decentralized passivity-based controller developed by Mohammadi, *et al.* [2]. This controller maps errors in position and velocity to desired roll, pitch, and thrust force of the quad-rotor. The desired yaw angle is also considered as an input to this controller, and is set to zero throughout the experiments. The desired orientation and thrust values are translated into the quad-rotor actuation commands via the onboard attitude controller embedded in the drone's onboard flight controllers. In this thesis, the cascaded PID attitude controller embedded in the Crazyflie Nano Quad-copters is utilized for this purpose.

7.2 Simulations

The control code was implemented in C++ and executed on a laptop computer with Intel(R) Core i7-8750H CPU running at 2.20 GHz with 16.0 GB of RAM. Mosek optimization toolbox [77] was utilized for solving the quadratic optimization problems across the whole algorithm during the simulations. C-programming library *cddlib* was used for transformations between linear inequality definition to vertices of a

polyhedral convex region [78].

Two operational scenarios were explored in the simulations. The design of these scenarios was informed by the constraints of the experimental task environment to maintain consistency between the simulations and experiments. The proposed method is applicable to both planar (i.e., $d = 2$), or spatial ($d = 3$) movements. However, limited by computation power, the aforementioned scenarios were developed in a two-dimensional workspace ($d = 2$) with three drones (i.e., $N = 3$). In one scenario the flock is flown through a narrow passage where the formation must contract to get through. In another scenario, multiple obstacles were placed in front of the flock's heading direction. The group may be forced to allow obstacles to pass through the formation in order to save the shape and follow the commanded operator's velocity. This will be referred to as *Non-convex Formation Scenario* since the presence of the obstacles violates the convexity of the area covered by the formation.

7.2.1 Modeling Parameters Definition

The parameters used to define the models are as follows:

1. The drones are considered to have the same size with $r_i = 0.15$m and a mass of $m_i = 0.03$kg.

2. The minimum safe inter-agent distance is set to $d_{ij} = 2r_i = 0.3$m.

3. The safe distance from the drones to the obstacles is set to $r_i = 0.15$m. Added safety margin is put in place by dilating the obstacles boundaries:

   (a) For prospective formation evaluation and pathfinding purposes, the obstacle boundaries are inflated by $0.15\text{m} + \epsilon$ where $\epsilon = 0.02$m.

(b) In the computation of the convex region constraints in Algorithm 3, the obstacles boundaries are inflated by 0.15m.

4. The control time horizon is set to $T_h = 1.0$s. For the MPC optimization, this horizon is unevenly divided into 30 steps as follows:

$$T_h = [0.01,\ 0.02,\ ...,\ 0.1,\ 0.12,\ 0.14,\ ..., 0.2,$$
$$0.24,\ 0.28, ...,\ 0.60,\ 0.68,\ 0.76, ...,\ 1.0]$$

5. The dimensions of the bounding boxes discussed in Section 5.2.1 are given by $l_w = l_v = 0.2$m and $\epsilon = 0.03$m.

6. The ellipsoids inside the bounding boxes are initialized with $\lambda_2 = \lambda_3 = 0.02$m.

7. The scalarization parameters in the optimization problem are set to:

w_v	w_u	w_f	w_s
1.0	0.5	1.0	0.1

Table 7.1: Scalarization weights used in optimization (6.4.1)

8. In the Narrow Passage scenario, $K_P = \mathbf{diag}(0.5, 0.5)$ and $K_I = \mathbf{diag}(0.4, 0.4)$. In the Non-convex Formation case, the values were set to $K_P = \mathbf{diag}(0.3, 0.3)$ and $K_I = \mathbf{diag}(0.2, 0.2)$.

9. After each iteration, $k_s = 5$ steps of the optimal inputs obtained are applied to update the trajectories for each agent.

7.2.2 Narrow Passage Scenario

In this scenario, three agents start from rest ($\dot{\mathbf{q}}_i = \mathbf{0}$ for $i \in \{1, 2, 3\}$) and receive a constant command velocity along $-y$ direction with the magnitude of $||\mathbf{v}_h|| = 0.1\frac{m}{s}$. A collection of six hexagonal overlapping obstacles were used to define a narrowing passage. The workspace and the resulting trajectories are depicted in Figure 7.1 with obstacles in black color. In addition to the trajectories, the formation has been drawn at the beginning, the end, and five other intermediate stages of the whole simulation time span. The passage can be categorized with three levels of space: 1) Narrow, 2) Narrowest, and 3) Wide.

In the beginning, the flock passes through a moderately narrow entrance which requires a moderate deviation from the given original reference formation $\mathcal{S}_1^r$. A total number of three formations are considered. The optimal formation is computed in parallel based on each, and the highest available priority is selected. Figure 7.2 shows the three desired formation shapes. Moving along the passage, the middle section is the narrowest part where the flock has to switch to another different formation shape ($\mathcal{S}_2$) to avoid collisions. Finally, the agents enter the wide section where they can switch back to the original formation $\mathcal{S}_1^r$.

The operation of the collision avoidance constraints is further verified by analyzing the obstacle-robot and robot-robot distance values. Figure 7.3 presents the variations of distance to obstacles over the simulation period. It is evident that even though Robot 1 and Robot 2 were moving in close proximity to the obstacles, their trajectory never violated the safe distance constraint. Moreover, inter-robot distances are plotted in Figure 7.4 and it can be observed that inter-agent collision avoidance constraints were never violated. Even during the transition to the narrowest part

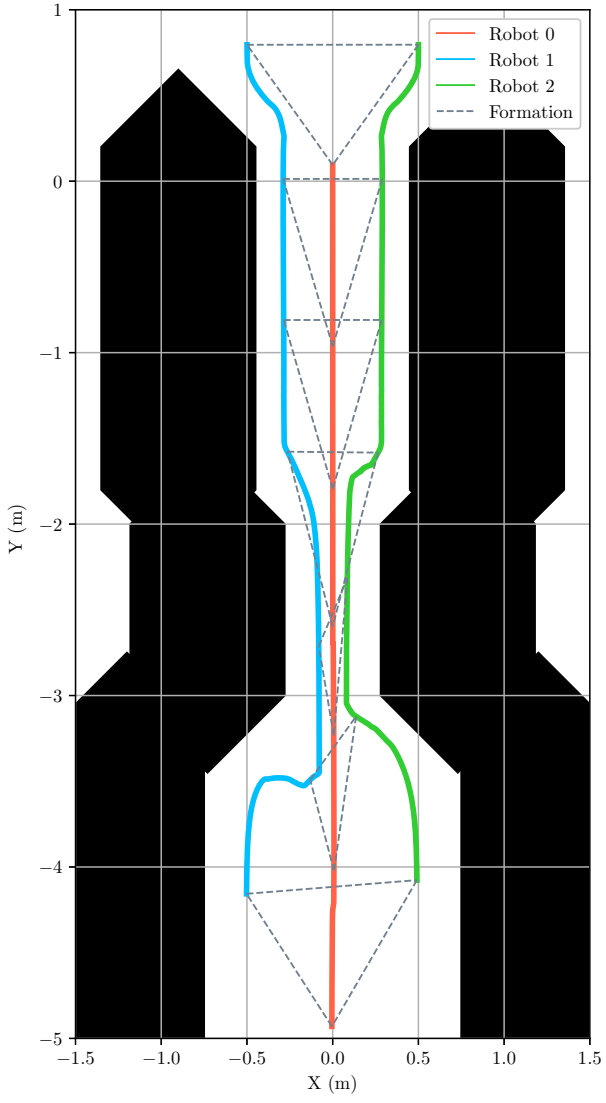

Figure 7.1: Trajectories of three drones moving towards a narrow passage in 2D workspace.

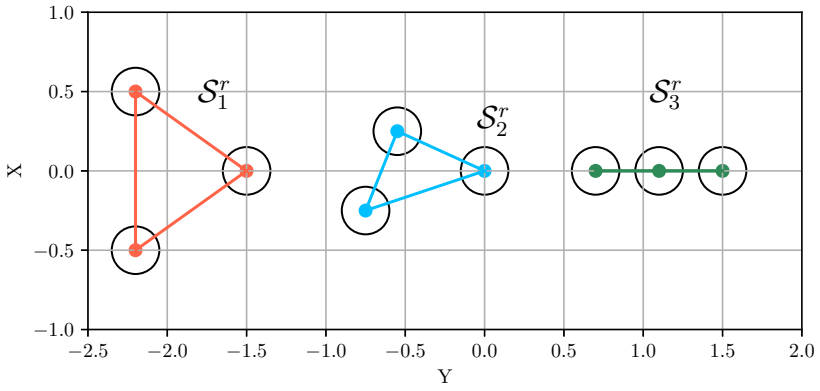

Figure 7.2: Three reference formations used in Narrow Passage Scenario where $\mathcal{S}_1^r$ has the highest priority and $\mathcal{S}_3^r$ has the lowest priority.

(around $t = 26.6$s), the distance between Robot 1 an Robot 2 remains above 0.3m.

One of the objectives of the proposed control approach is to follow the operator's reference velocity command as closely as possible. Figure 7.5 compares the flock's velocity with the operator's commanded velocity. It can be observed that the flock velocity is following this reference. Although small periods of transitions do occur due to the change in formation, the flock velocity eventually converges to the commanded velocity as expected.

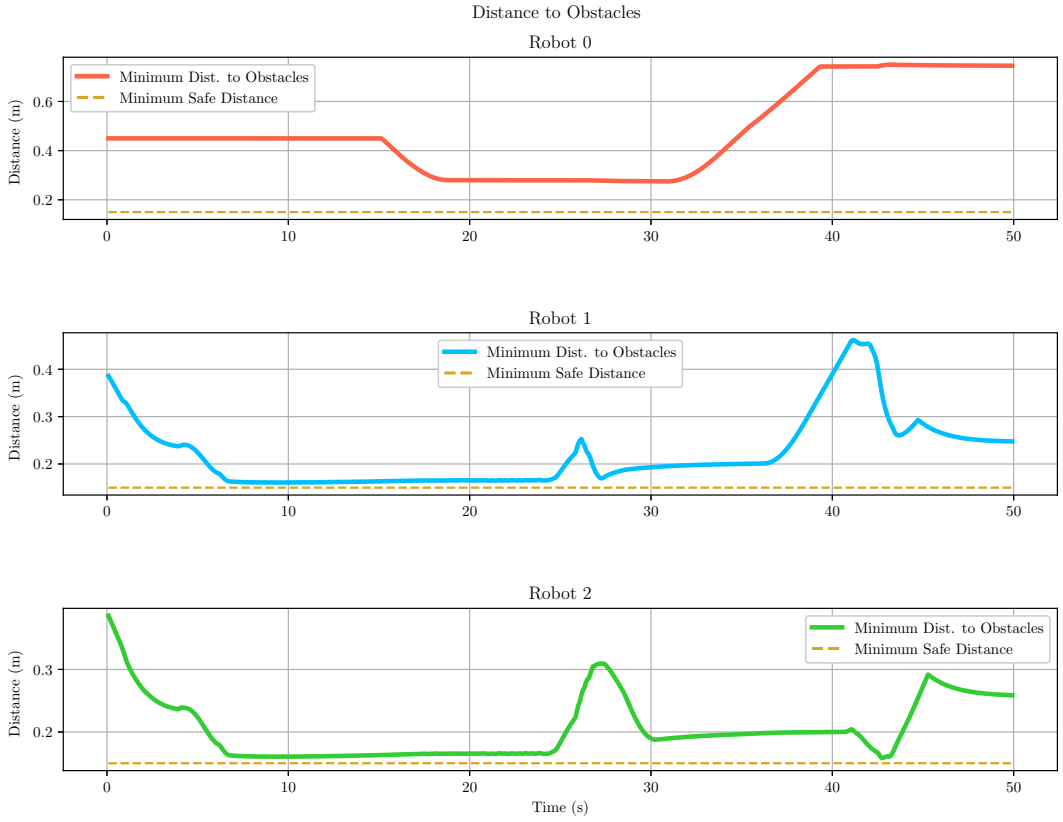

Figure 7.3: Distance between each robot and the closest obstacle in the workspace in Narrow Passage Scenario.

7.2.3 Non-Convex Formation Scenario

The performance of the proposed control strategy was partially evaluated in the previous scenario. However in that scenario, the entire flock formation could be fitted in the obstacle-free convex area with the need for allowing obstacles to pass through the formation. The next scenario explores a more challenging case where this may not always be feasible.

The number of drones, workspace dimensions, and the operator's reference velocity are the same as those in the previous case. Three small-sized hexagonal obstacles are

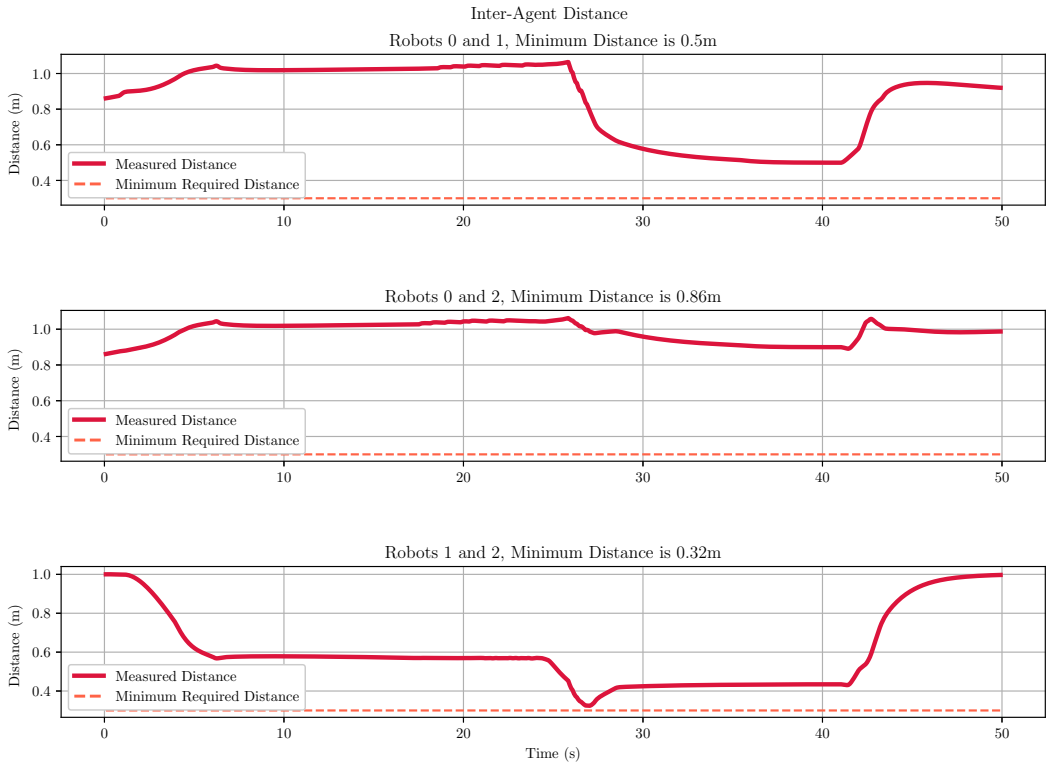

Figure 7.4: Inter-agent distance values while passing through the narrow passages shown in Figure 7.1.

defined as depicted in Figure 7.6. Considering the reference command along $-y$ direction, and the initial positions as the only reference formation, the challenge is to traverse around the obstacles and preserve the formation as much as possible. At the beginning of the simulation, Robot 0 has to sway towards $+x$ direction to avoid collision with $\mathcal{O}_1$. Simultaneously, to minimize deviation from the reference formation, the other members of the group change their direction similar to their leader (Robot 0). Further along the way, Robot 1 must avoid two obstacles ($\mathcal{O}_1$ and $\mathcal{O}_2$) which are in close proximity. After successfully moving around the obstacles and leaving them behind, the agents join together and resume their with the original formation at the

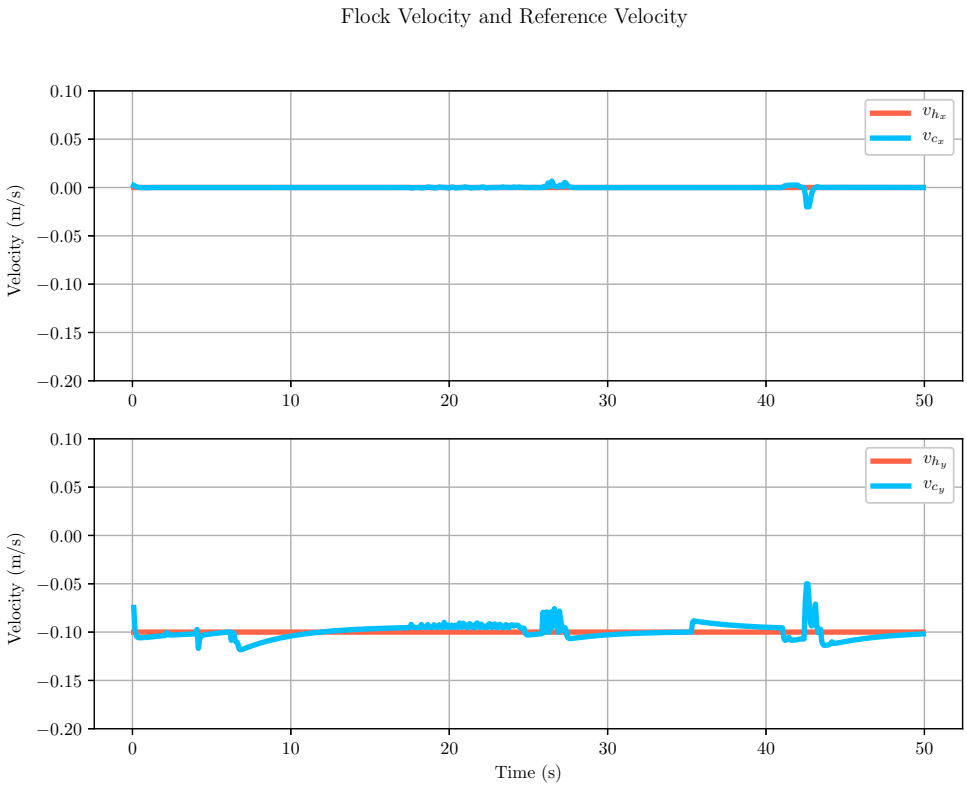

Figure 7.5: The reference velocity and the velocity of the center of the flock in the simulation which is shown in Figure 7.1.

beginning of the simulation.

The adherence to the collision avoidance constraints can be verified by reviewing the data in Figure 7.7 and 7.8. All three robots had to traverse in short distance from the obstacles and all the trajectories were obtained constrained inside $\mathcal{W}_f$ which 0.15m away from the obstacles.

As the flock moves along the x direction to avoid collision and preserve formation, its velocity is compromised since the reference command velocity is constant and in the $-y$ direction (See Figure 7.6). Figure 7.9 plots the components of the flock velocity and it shows that the x component of the velocity becomes non-zero transiently and

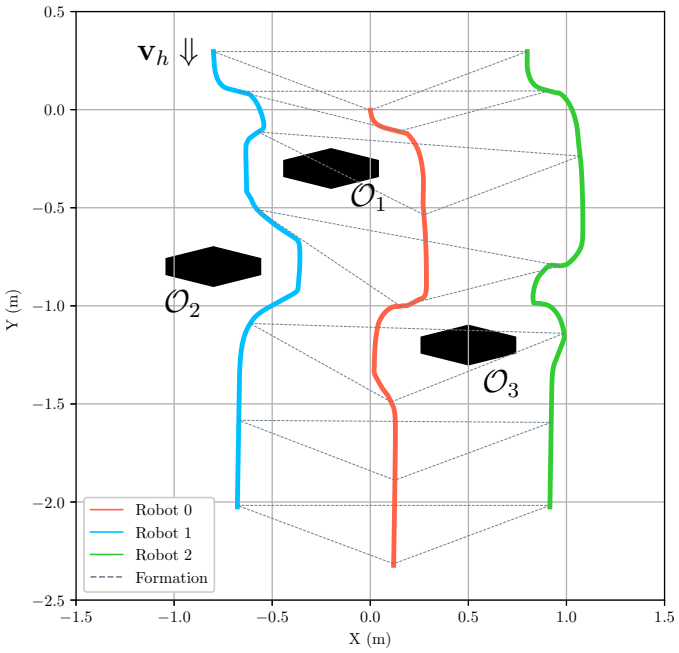

Figure 7.6: Non-Convex Formation Scenario. The trajectories are shown in solid colored lines, intermediate formation shapes with dashed grey line, and obstacles are painted in black color.

eventually returns to zero as requested by the reference $\mathbf{v}_h$. Similarly, it can be seen how the velocity was controlled along the y axis, where despite the compromises the flock had made to avoid the obstacles, it eventually converges to a velocity in the $-y$ direction with the desired magnitude.

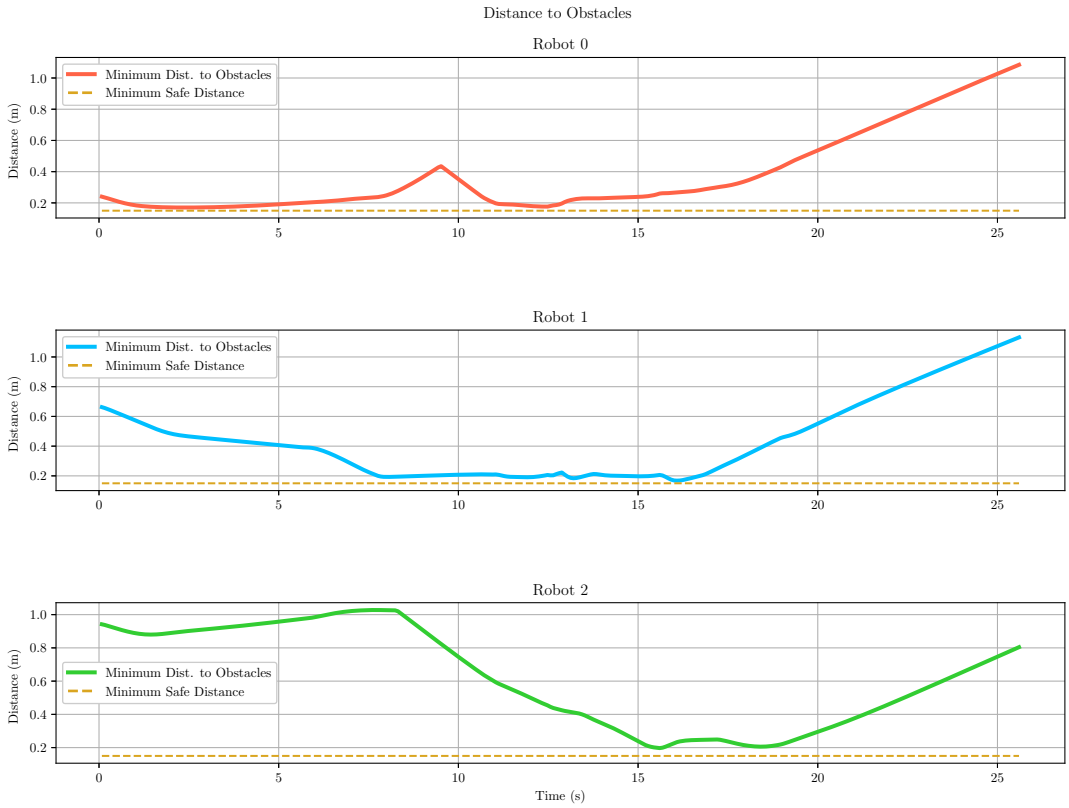

Figure 7.7: Distance between each robot and the closest obstacle in the workspace in Non-Convex Formation Scenario.

7.3 Experiments

For the purposes of real-time implementation, the same setup and processor were used similar to Section 7.2. To increase computation efficiency further, CUDA programming [79] was applied for matrix calculations to prepare the constraints and objective functions stated in Chapter 6. In addition, to solve the quadratic programming stated in (6.4.1), OSQP solver [80] was used along with Mosek optimization API [77] for solving smaller optimizations in parallel. It should be noted that to fully

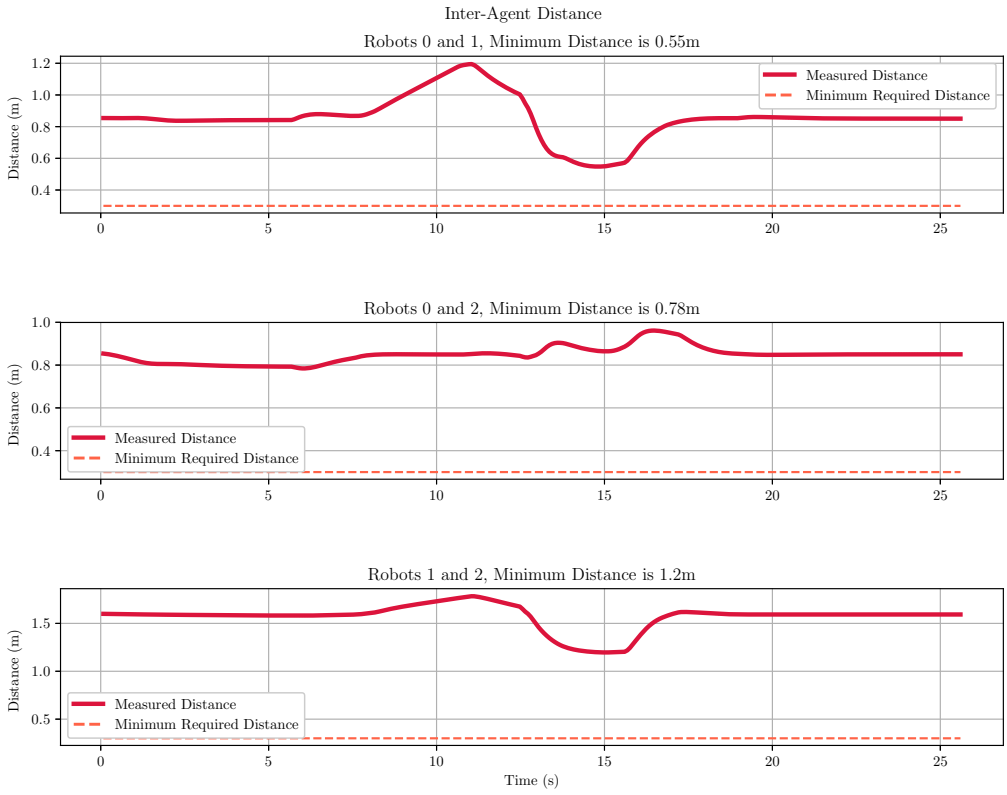

Figure 7.8: Inter-agent distance values while passing through the narrow passages shown in Figure 7.6.

exploit the potential of OSQP solver, the size of constraints for quadratic programming were fixed and the solver was initialized prior to the start of experiments in order to take advantage of the warm-start options at each iteration.

Figure 7.10 shows the experimental setup used for validation in a lab environment. The Host Computer is the same as the one mentioned in Section 7.2 and runs the whole algorithm at the rate of 25Hz. This computation rate is bounded by the hardware power and the efficiency of the implementation. Three Crazyflie Nano Quad-Copters were used as agents in the experiments. Each drone weighs slightly

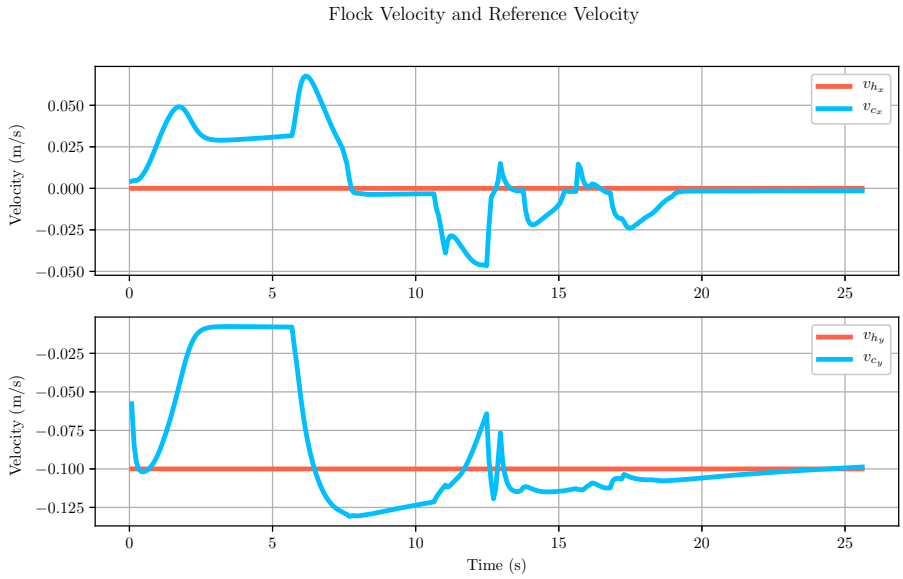

Figure 7.9: The reference velocity and the velocity of the center of the flock in the simulation shown in Figure 7.6.

less than 30g and its dimensions measure at 92mm × 92mm. The Host Computer communicates with the drones via two Crazyradio Dongles. The drones positions are measured by seven Optitrack Flex 13 cameras in the laboratory workspace. The position data is streamed to the Host Computer at the rate of 120Hz. The trajectory controller runs at 100Hz and computes the control commands based on the latest position received. This is due to the fact that the trajectories accumulated in the memory are calculated based on 10ms time steps.

The experiment scenarios are designed in a two-dimensional workspace. In the experiments, the drones first fly vertically to reach a hovering altitude. Second, the proposed algorithm runs for a certain period of time in the horizontal plane. Finally, the drones land while maintaining their horizontal position. In the following sections, only the horizontal portions of the trajectories are presented as they are relevant to the performance of the proposed algorithm.

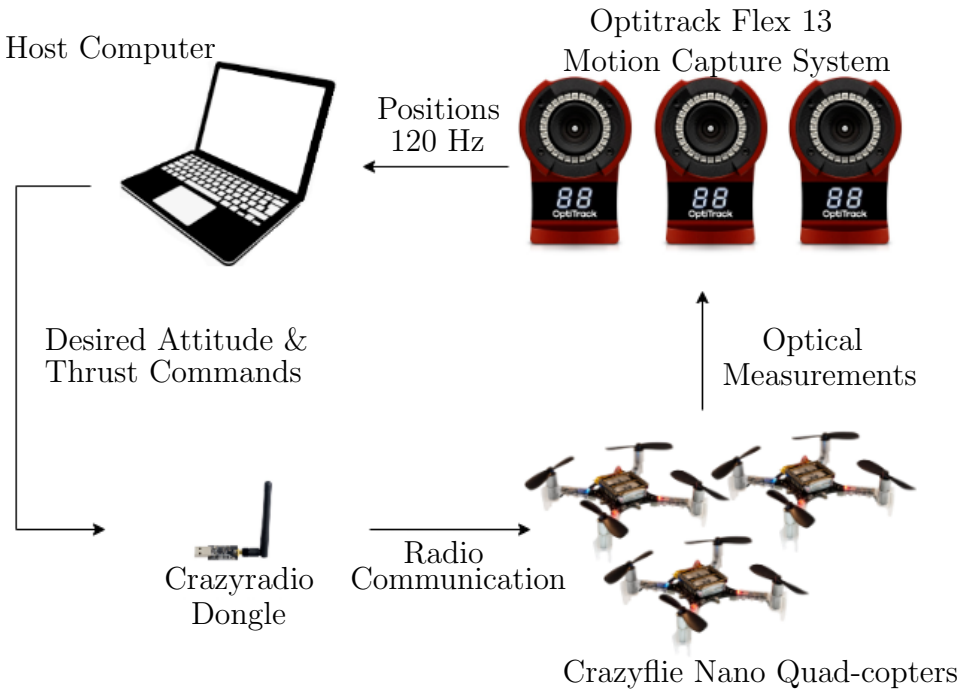

Figure 7.10: Experimental Setup

7.3.1 Narrow Passage Scenario

The scenario discussed in simulation 7.2.2 is simplified and adapted according to the limits of the lab workspace. The arrangement of robots' initial positions and obstacles location are depicted in Figure 7.11. Four obstacles are defined such that they create a three-level passage. The flock starts by entering a slightly narrow passage which necessitates small deformation in the original shape of the flock. Further down the path, the obstacles are located at a shorter distance from each other, leading to a change of formation to prevent collisions. At the end, the flock enters an obstacle-free

area where it can switch back to its original formation.

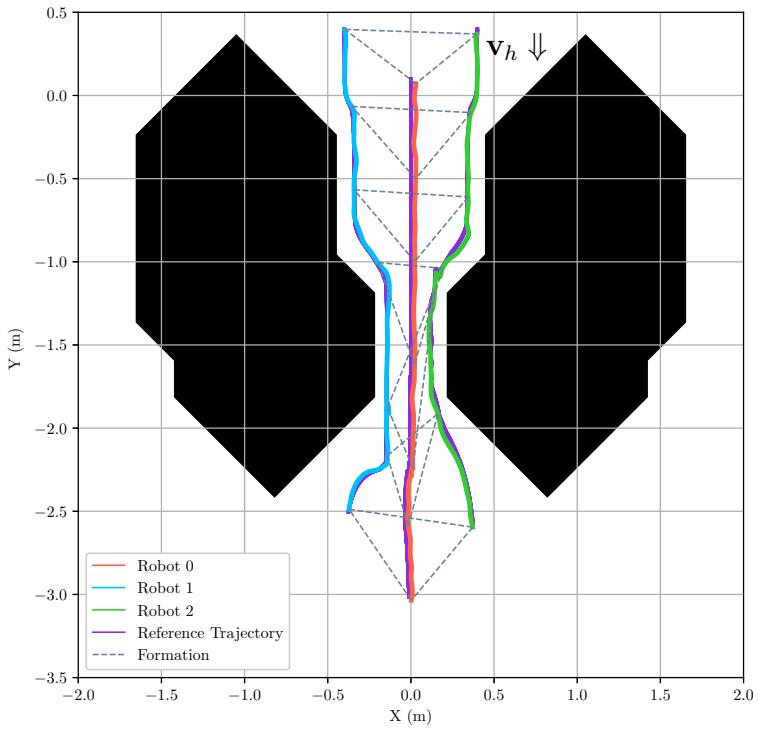

Figure 7.11: The drones trajectories in Narrow Passage Scenario experiment. This scenario is a modified version of the simulation scenario presented in Section 7.2.2.

Following the logic stated in Section 7.2.1, two levels of inflation of obstacles were used in this experiment. However, there is always a minimal amount of errors in the real world control of quad-copters and in order to account for these uncertainties, the inflation amount used in path finding and formation evaluation was set to $0.15\text{m} + \epsilon$ where $\epsilon = 0.08m$. Figure 7.12 plots the measured distance of each drone from the obstacles during the experiment. It shows the efficacy of the proposed algorithm

in the presence of static obstacles in the environment and how it can be tuned to overcome uncertainties in controlling the position of the drones.

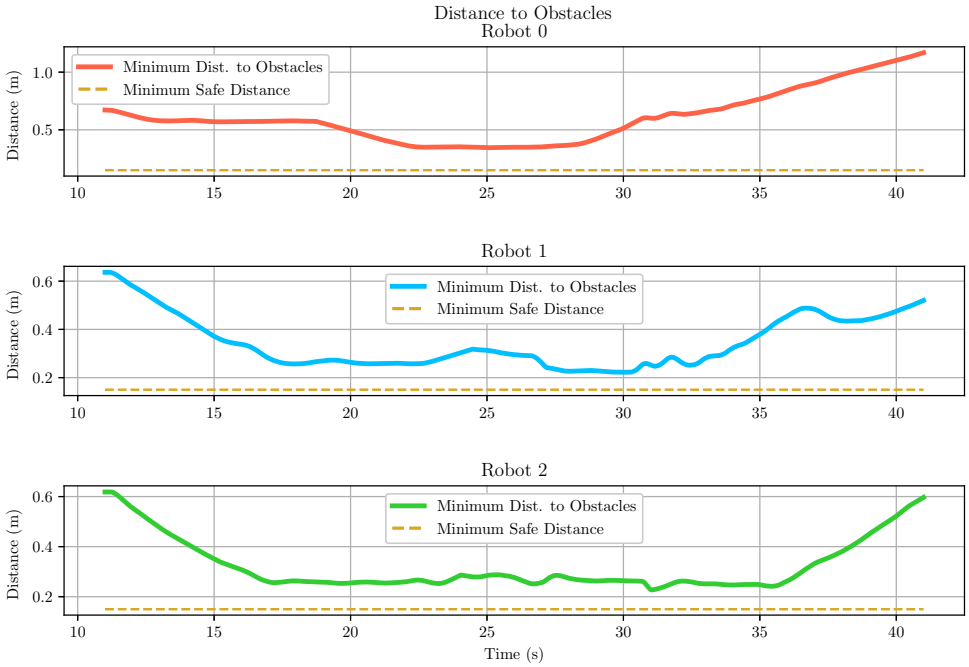

Figure 7.12: The distance to obstacles for each robot in the experiment of Figure 7.11.

The errors in the trajectory tracking controller are slightly amplified from an inter-agent distance perspective. Figure 7.13 compares the desired distance against the measured distance between each pair of drones in the experiment. The most critical moment happens around $t = 27$s where the flock experiences a formation change as it enters the narrowest part of the passage. As it can be seen in Figure 7.13, the distance between Robots 1 and 2 reached the minimum value of 0.3m for a second but never violated the minimum required distance between the agents.

The operator's reference velocity command and the velocity of the centre of the flock are compared in Figure 7.14. This confirms velocity tracking where possible

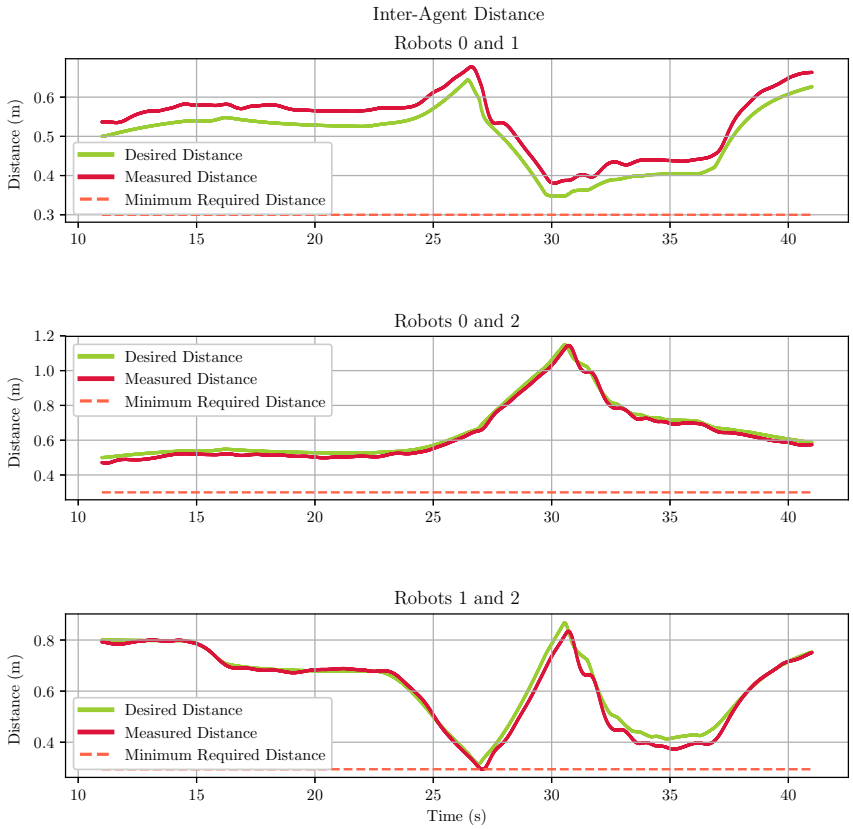

Figure 7.13: Inter-agent distances in the Narrow Passage Experiment.

with some transient behaviour due to the presence of the obstacles.

7.3.2 Non-convex Formation Scenario

The performance of the proposed control strategy was further evaluated by conducting an experiment similar to the simulation in Section 7.2.3. The goal of this experiment is to demonstrate the flexibility of solution which can be used in scenarios beyond simple contracting and expanding of the formation to fit within a certain space. Figure 7.15 depicts the result of an experiment in such a scenario. Given the positions of the

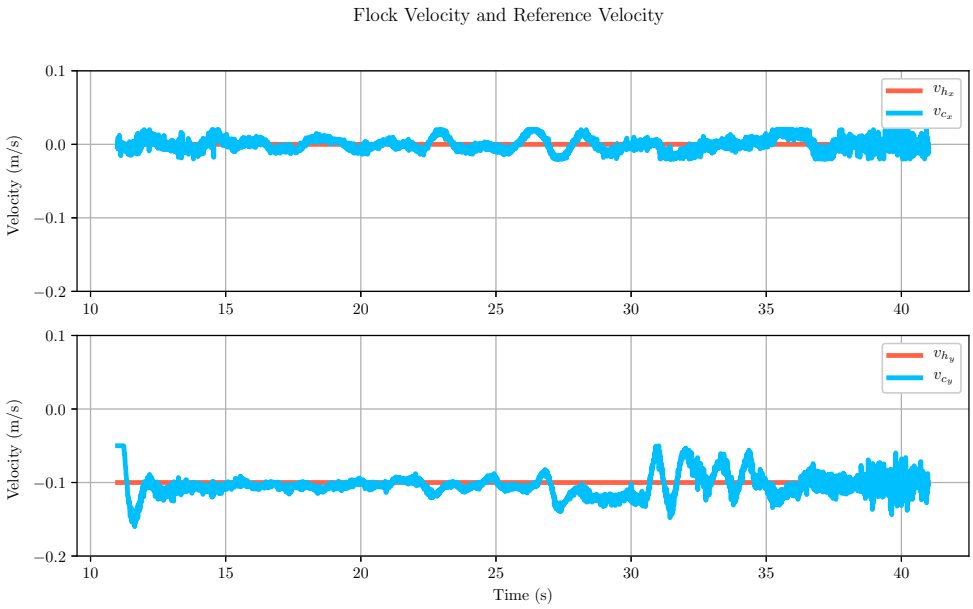

Figure 7.14: Velocity of the flock compared with operator's reference command in 7.11 experiment.

obstacles, the flock moves in the x direction while trying to maintain the formation as much as possible.

The results in Figure 7.16 show that the algorithm is successful in maintaining a safe distance between the drones and the obstacles throughout the experiment. Analogous to the previous scenario, obstacles were inflated by an additional factor of $\epsilon = 0.1$ in this case. Furthermore, from Figure 7.17, it is also evident that the agents maintained a safe distance from each other.

The deviations from reference velocities tend to be larger than those in the previous experiment. This is because the reference velocity was maintained constant through the whole experiment in Figure 7.15, and the flock had to move in other directions to avoid the obstacles while maintaining its formation. However, Figure 7.18 reveals that after any deviation, the velocity was controlled and eventually converged

to the reference velocity.

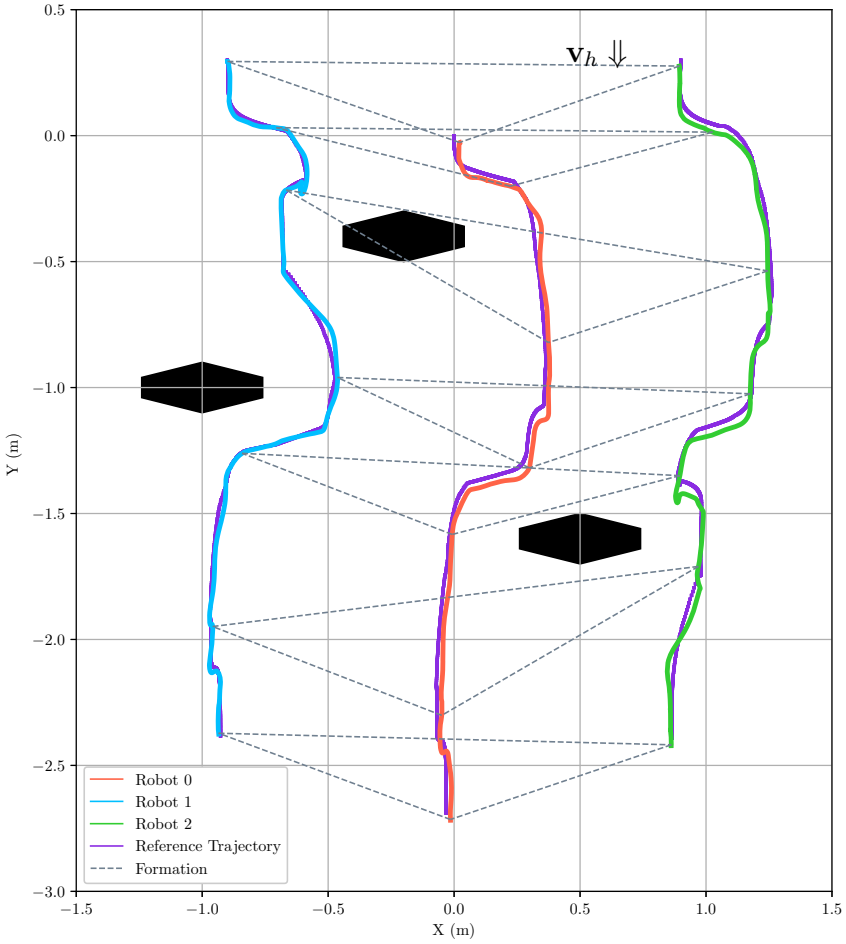

Figure 7.15: Non-convex Formation scenario experiment.

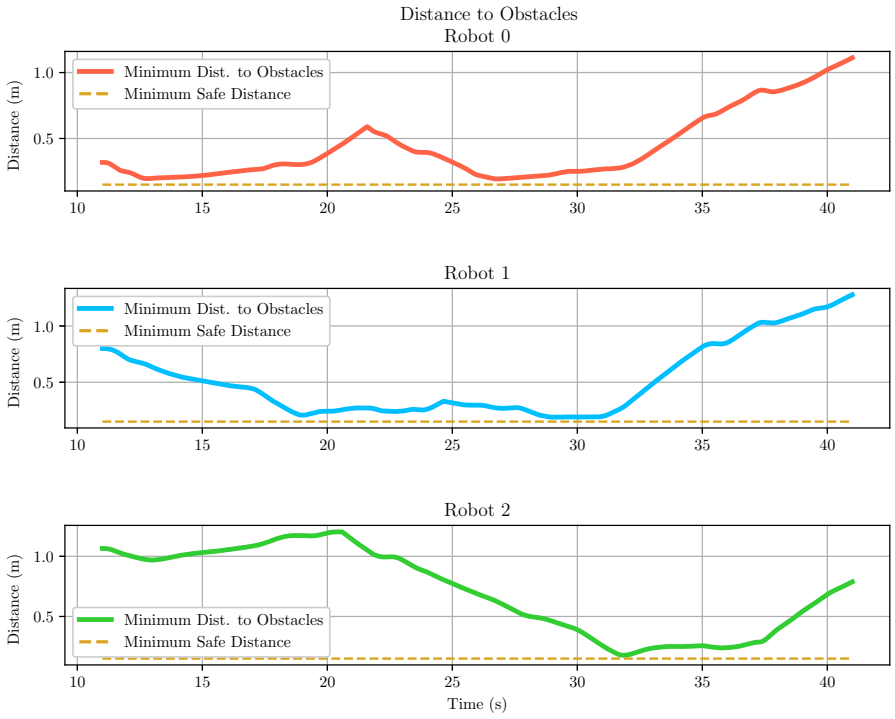

Figure 7.16: The distance to obstacles for each robot in the experiment of Figure 7.15.

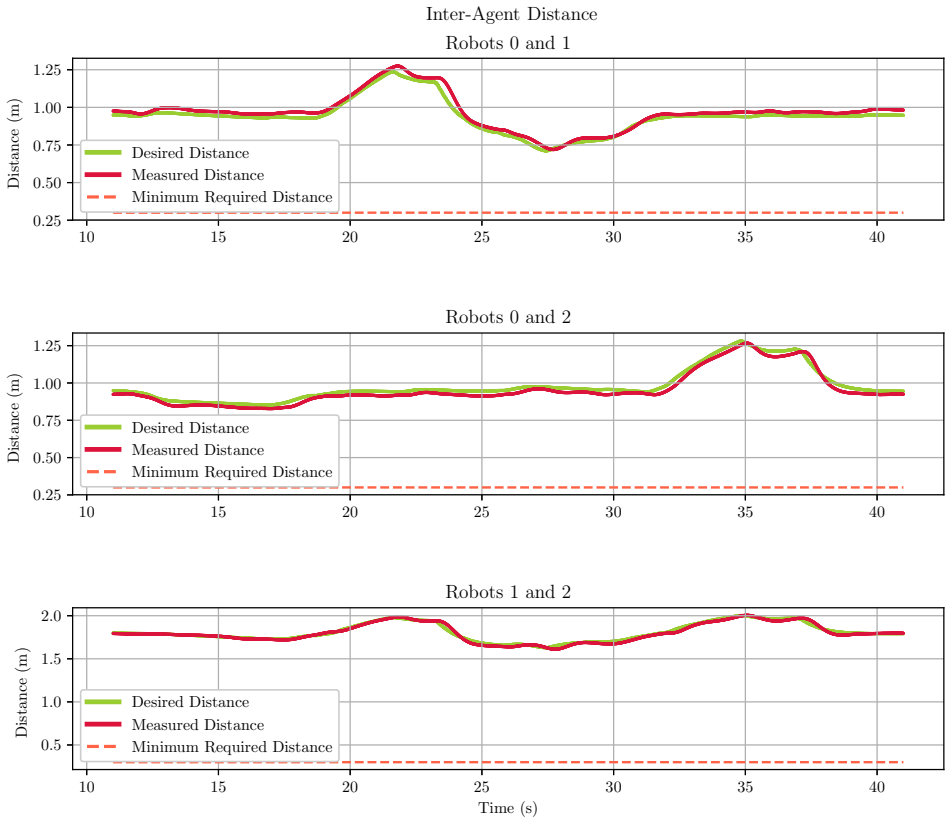

Figure 7.17: Inter-agent distances in the Non-Convex Formation Experiment.

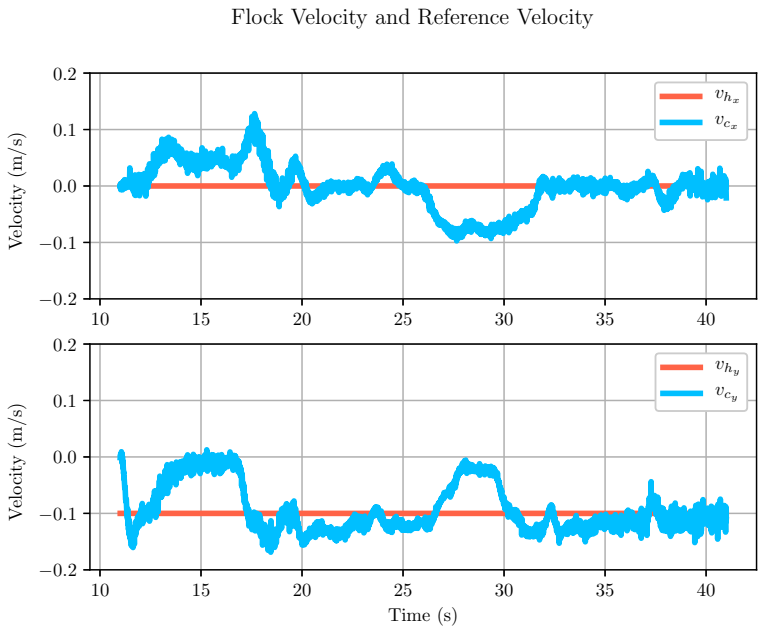

Figure 7.18: The flock velocity compared with against the operator's reference velocity in 7.15 experiment.

Chapter 8

Conclusion

This thesis presented a novel framework in which a human operator continuously controls the general motion of a flock of drones while the system autonomously adjusts itself to avoid collisions and maintain a desired formation as faithfully as possible. Unlike most prior work in the literature, the proposed algorithm is completely optimization-based, is designed for real-time applications, and incorporates the human reference velocity input in the process of finding the optimal trajectories for the drones.

The proposed framework attempts to maintain a reference formation as closely as possible while allowing for some adjustments to adapt to the task environment as the need arises. The target formation is obtained by formulating and solving a convex optimization problem. Multiple reference formations in a priority order are considered in order to deal with possible infeasibility. Transitions between these reference formations occur automatically considered. A method was proposed to eliminate the requirement for convexity of the formation shape by a novel convex region assignment algorithm.

An MPC-like optimization was formulated where linear collision-avoidance related constraints were developed. These linear constraints were generated in the direction of each agent's motion and they account for both robot-obstacle and robot-robot interactions. The trajectory of each agent was optimized to minimize the deviation from the input reference command and the optimal formation. The proposed approach was tested in both simulation and experiment and the results demonstrated the effectiveness of the algorithm for human-in-the-loop control of three drones.

8.1 Future Work

While the framework proposed in this thesis has addressed some of the challenges in the realm of human-multi-robot systems, there are still a number of challenges and avenues to extend the capabilities of this method in the future. In particular,

- **Decentralization**: Central algorithms such as the one in this thesis do not generally scale well with the number of agents when compared with decentralized methods running on the individual agents. Decentralized implementations are also more resilient and may tolerate failure in one or more agents. Therefore, decentralizing the proposed algorithm could be a step toward more reliability, efficiency, and scalability.

- **Dynamic Environment**: Many real-world applications include a mixture of static and dynamic obstacles. Multi-agent systems should be able to avoid moving objects as long as their motion can be predicted. Hence, extending the proposed control strategy to scenarios involving dynamic environments is another avenue for future research.

- **Considering Uncertainty**: Sensor measurements, robot and environment models are subject to uncertainty and noise. Examples are uncertainty in the position of the agents and shape and position of the obstacles. Moreover, environmental disturbance forces such as winds can affect the motion of each robot which is essential to consider in an unpredictable environment. Incorporating these modeling uncertainties in the proposed control framework can enhance its effectiveness in operating in more realistic task settings.

- **More Realistic Experiment Scenarios**: The experiments in this thesis were carried out in an indoor laboratory setting where the drones positions were measured with high accuracy using an optical motion tracking system. The obstacles shapes and positions were also assumed known. In a more realistic application scenario, the locations of the agents and the obstacles in the task environment must be determined using a combination of on-board sensors such RGB-Depth Cameras, IMU, LiDARs and GPS.

- **Handling Faults and Failure in Agents**: Extending the proposed control strategy to adapt in real-time to failure in one or more agents is another interesting area for future work.

Appendix A

Matrix Operations and Constants

A.1 Matrix Operations

A.1.1 Diagonally Concatenated Matrices

Assuming $\mathbf{M}_i \in \mathbb{R}^{m_i \times n_i}$ for $i \in \{1, 2, 3, .., k\}$, $\mathbf{diag}(\mathbf{M}_1, \mathbf{M}_2, ..., \mathbf{M}_k)$ will be defined as following:

$$\mathbf{O} = \mathbf{diag}(\mathbf{M}_1, \mathbf{M}_2, ..., \mathbf{M}_k) = \begin{bmatrix} \mathbf{M}_1 & \mathbf{0}_{m_1 \times n_2} & \dots & \mathbf{0}_{m_1 \times n_k} \\ \mathbf{0}_{m_2 \times n_1} & \mathbf{M}_2 & \dots & \mathbf{0}_{m_2 \times n_k} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{m_k \times n_1} & \mathbf{0}_{m_k \times n_2} & \dots & \mathbf{M}_k \end{bmatrix} \tag{A.1.1}$$

where results in $\mathbf{O} \in \mathbb{R}^{\sum m_i \times \sum n_i}$.

A.2 Constant Matrices

A.2.1 Velocity Extractor Matrix

1. If $d = 3$,

$$\mathbf{I}_v = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.2.1}$$

2. If $d = 2$,

$$\mathbf{I}_v = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.2.2}$$

A.2.2 Position Extractor Matrix

1. If $d = 3$,

$$\mathbf{I}_p = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{A.2.3}$$

2. If $d = 2$,

$$\mathbf{I}_p = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{A.2.4}$$

Bibliography

[1] Mostafa Hassanalian and Abdessattar Abdelkefi. Classifications, applications, and design challenges of drones: A review. *Progress in Aerospace Sciences*, 91: 99–131, 2017.

[2] Keyvan Mohammadi, Mohammad Jafarinasab, Shahin Sirouspour, and Eric Dyer. Decentralized motion control in a cabled-based multi-drone load transport system. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4198–4203. IEEE, 2018.

[3] Yash Mulgaonkar. *Automated recharging for persistence missions with multiple micro aerial vehicles*. PhD thesis, University of Pennsylvania, 2012.

[4] Wolfgang Hönig, James A Preiss, TK Satish Kumar, Gaurav S Sukhatme, and Nora Ayanian. Trajectory planning for quadrotor swarms. *IEEE Transactions on Robotics*, 34(4):856–869, 2018.

[5] Alex Kushleyev, Daniel Mellinger, Caitlin Powers, and Vijay Kumar. Towards a swarm of agile micro quadrotors. *Autonomous Robots*, 35(4):287–300, 2013.

[6] Keyvan Mohammadi, Shahin Sirouspour, and Ali Grivani. Passivity-based control of multiple quad-rotors carrying a cable-suspended payload. *IEEE/ASME Transactions on Mechatronics*, 2021.

[7] Soon-Jo Chung, Aditya Avinash Paranjape, Philip Dames, Shaojie Shen, and Vijay Kumar. A survey on aerial swarm robotics. *IEEE Transactions on Robotics*, 34(4):837–855, 2018.

[8] Javier Alonso-Mora, Tobias Naegeli, Roland Siegwart, and Paul Beardsley. Collision avoidance for aerial vehicles in multi-agent scenarios. *Autonomous Robots*, 39(1):101–121, 2015.

[9] Stefano Stramigioli, Robert Mahony, and Peter Corke. A novel approach to haptic tele-operation of aerial robot vehicles. In *2010 IEEE International Conference on Robotics and Automation*, pages 5302–5308. IEEE, 2010.

[10] Antonio Franchi, Cristian Secchi, Markus Ryll, Heinrich H Bulthoff, and Paolo Robuffo Giordano. Shared control: Balancing autonomy and human assistance with a group of quadrotor uavs. *IEEE Robotics & Automation Magazine*, 19(3):57–68, 2012.

[11] Hossein Rastgoftar and Ella M Atkins. Cooperative aerial payload transport guided by an in situ human supervisor. *IEEE Transactions on Control Systems Technology*, 27(4):1452–1467, 2018.

[12] Morgan Quigley, Michael A Goodrich, and Randal W Beard. Semi-autonomous

human-uav interfaces for fixed-wing mini-uavs. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2457–2462. IEEE, 2004.

[13] Sara Howitt and Dale Richards. The human machine interface for airborne control of uavs. In *2nd AIAA" Unmanned Unlimited" Conf. and Workshop & Exhibit*, page 6593, 2003.

[14] Keyvan Mohammadi, Shahin Sirouspour, and Ali Grivani. Control of multiple quad-copters with a cable-suspended payload subject to disturbances. *IEEE/ASME Transactions on Mechatronics*, 25(4):1709–1718, 2020.

[15] Thomas Kopfstedt, Masakazu Mukai, Masayuki Fujita, and Christoph Ament. Control of formations of uavs for surveillance and reconnaissance missions. *IFAC Proceedings Volumes*, 41(2):5161–5166, 2008.

[16] Robin Deits and Russ Tedrake. Computing large convex regions of obstacle-free space through semidefinite programming. In *Algorithmic foundations of robotics XI*, pages 109–124. Springer, 2015.

[17] Sikang Liu, Michael Watterson, Kartik Mohta, Ke Sun, Subhrajit Bhattacharya, Camillo J Taylor, and Vijay Kumar. Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments. *IEEE Robotics and Automation Letters*, 2(3):1688–1695, 2017.

[18] Kimon P Valavanis and M Kontitsis. A historical perspective on unmanned aerial vehicles. In *Advances in Unmanned Aerial Vehicles*, pages 15–46. Springer, 2007.

[19] Wojciech Giernacki, Mateusz Skwierczyński, Wojciech Witwicki, Paweł Wroński, and Piotr 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)*, pages 37–42. IEEE, 2017.

[20] Eric Dyer, Shahin Sirouspour, and Mohammad Jafarinasab. Energy optimal control allocation in a redundantly actuated omnidirectional uav. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5316–5322. IEEE, 2019.

[21] Raphael Zufferey, Jesús Tormo-Barbero, M Mar Guzmán, Fco Javier Maldonado, Ernesto Sanchez-Laulhe, Pedro Grau, Martín Pérez, José Ángel Acosta, and Anibal Ollero. Design of the high-payload flapping wing robot e-flap. *IEEE Robotics and Automation Letters*, 6(2):3097–3104, 2021.

[22] Dane Bamburry. Drones: Designed for product delivery, revisited. *Design Management Review*, 33(3):34–43, 2022.

[23] Amazon Staff. Amazon prime air prepares for drone deliveries. `https://www.aboutamazon.com/news/transportation/amazon-prime-air-prepares-for-drone-deliveries`, 2022. Accessed: June 13, 2022.

[24] Cornelius A. Thiels, Johnathon M. Aho, Scott P. Zietlow, and Donald H. Jenkins. Use of unmanned aerial vehicles for medical product transport. *Air Medical Journal*, 34(2):104–108, 2015. ISSN 1067-991X. doi: https://doi.org/10.1016/

j.amj.2014.10.011. URL https://www.sciencedirect.com/science/article/pii/S1067991X14003332.

[25] Sonia Waharte and Niki Trigoni. Supporting search and rescue operations with uavs. In *2010 international conference on emerging security technologies*, pages 142–147. IEEE, 2010.

[26] Ebtehal Turki Alotaibi, Shahad Saleh Alqefari, and Anis Koubaa. Lsar: Multi-uav collaboration for search and rescue missions. *IEEE Access*, 7:55817–55832, 2019.

[27] Youngjib Ham, Kevin K Han, Jacob J Lin, and Mani Golparvar-Fard. Visual monitoring of civil infrastructure systems via camera-equipped unmanned aerial vehicles (uavs): a review of related works. *Visualization in Engineering*, 4(1): 1–8, 2016.

[28] Danny Lee, Joe Zhou, and Wong Tze Lin. Autonomous battery swapping system for quadcopter. In *2015 international conference on unmanned aircraft systems (ICUAS)*, pages 118–124. IEEE, 2015.

[29] Yash Mulgaonkar and Vijay Kumar. Autonomous charging to enable long-endurance missions for small aerial robots. In *Micro-and Nanotechnology Sensors, Systems, and Applications VI*, volume 9083, pages 404–418. SPIE, 2014.

[30] Koji AO Suzuki, Paulo Kemper Filho, and James R Morrison. Automatic battery replacement system for uavs: Analysis and design. *Journal of Intelligent & Robotic Systems*, 65(1):563–586, 2012.

[31] Pietro Tosato, Daniele Facinelli, Maurizio Prada, Luca Gemma, Maurizio Rossi, and Davide Brunelli. An autonomous swarm of drones for industrial gas sensing applications. In *2019 IEEE 20th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, pages 1–6. IEEE, 2019.

[32] Dario Albani, Joris IJsselmuiden, Ramon Haken, and Vito Trianni. Monitoring and mapping with robot swarms for agricultural applications. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE, 2017.

[33] Mauro S Innocente and Paolo Grasso. Self-organising swarms of firefighting drones: Harnessing the power of collective intelligence in decentralised multi-robot systems. *Journal of Computational Science*, 34:80–101, 2019.

[34] Carlo Masone, Heinrich H Bülthoff, and Paolo Stegagno. Cooperative transportation of a payload using quadrotors: A reconfigurable cable-driven parallel robot. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1623–1630. IEEE, 2016.

[35] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE international conference on robotics and automation*, pages 2520–2525. IEEE, 2011.

[36] Mohammad Jafarinasab and Shahin Sirouspour. Adaptive motion control of aerial robotic manipulators based on virtual decomposition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1858–1863. IEEE, 2015.

[37] Giuseppe Loianno, Justin Thomas, and Vijay Kumar. Cooperative localization and mapping of mavs using rgb-d sensors. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4021–4028. IEEE, 2015.

[38] Samir Bouabdallah, Pierpaolo Murrieri, and Roland Siegwart. Design and control of an indoor micro quadrotor. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 5, pages 4393–4398. IEEE, 2004.

[39] Daniel Mellinger, Nathan Michael, and Vijay Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. *The International Journal of Robotics Research*, 31(5):664–674, 2012.

[40] Matthias Faessler, Antonio Franchi, and Davide Scaramuzza. Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories. *IEEE Robotics and Automation Letters*, 3(2):620–626, 2017.

[41] Daniel Mellinger, Alex Kushleyev, and Vijay Kumar. Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams. In *2012 IEEE international conference on robotics and automation*, pages 477–483. IEEE, 2012.

[42] Alexander Erokhin, Vladimir Erokhin, Sergey Sotnikov, and Anatoly Gogolevsky. Optimal multi-robot path finding algorithm based on a. In *Proceedings of the Computational Methods in Systems and Software*, pages 172–182. Springer, 2018.

[43] Glenn Wagner and Howie Choset. M*: A complete multirobot path planning

algorithm with performance bounds. In *2011 IEEE/RSJ international conference on intelligent robots and systems*, pages 3260–3267. IEEE, 2011.

[44] Hai Zhu, Bruno Brito, and Javier Alonso-Mora. Decentralized probabilistic multi-robot collision avoidance using buffered uncertainty-aware voronoi cells. *Autonomous Robots*, 46(2):401–420, 2022.

[45] Mark Debord, Wolfgang Hönig, and Nora Ayanian. Trajectory planning for heterogeneous robot teams. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7924–7931. IEEE, 2018.

[46] Li Wang, Aaron D Ames, and Magnus Egerstedt. Safety barrier certificates for collisions-free multirobot systems. *IEEE Transactions on Robotics*, 33(3): 661–674, 2017.

[47] Dimitra Panagou, Dušan M Stipanović, and Petros G Voulgaris. Distributed coordination control for multi-robot networks using lyapunov-like barrier functions. *IEEE Transactions on Automatic Control*, 61(3):617–632, 2015.

[48] Tomas Baca, Giuseppe Loianno, and Martin Saska. Embedded model predictive control of unmanned micro aerial vehicles. In *2016 21st international conference on methods and models in automation and robotics (MMAR)*, pages 992–997. IEEE, 2016.

[49] Moses Bangura and Robert Mahony. Real-time model predictive control for quadrotors. *IFAC Proceedings Volumes*, 47(3):11773–11780, 2014.

[50] Melissa Greeff and Angela P Schoellig. Flatness-based model predictive control

for quadrotor trajectory tracking. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6740–6745. IEEE, 2018.

[51] Tomas Baca, Daniel Hert, Giuseppe Loianno, Martin Saska, and Vijay Kumar. Model predictive trajectory tracking and collision avoidance for reliable outdoor deployment of unmanned aerial vehicles. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6753–6760. IEEE, 2018.

[52] Carlos E Luis and Angela P Schoellig. Trajectory generation for multiagent point-to-point transitions via distributed model predictive control. *IEEE Robotics and Automation Letters*, 4(2):375–382, 2019.

[53] Carlos E Luis, Marijan Vukosavljev, and Angela P Schoellig. Online trajectory generation with distributed model predictive control for multi-robot motion planning. *IEEE Robotics and Automation Letters*, 5(2):604–611, 2020.

[54] Yang Quan Chen and Zhongmin Wang. Formation control: a review and a new consideration. In *2005 IEEE/RSJ International conference on intelligent robots and systems*, pages 3181–3186. IEEE, 2005.

[55] Zhiyun Lin, Wei Ding, Gangfeng Yan, Changbin Yu, and Alessandro Giua. Leader–follower formation via complex laplacian. *Automatica*, 49(6):1900–1906, 2013.

[56] Jaydev P Desai, Jim Ostrowski, and Vijay Kumar. Controlling formations of multiple mobile robots. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, volume 4, pages 2864–2869. IEEE, 1998.

[57] Noah Cowan, O Shakerina, Rene Vidal, and Shankar Sastry. Vision-based follow-the-leader. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, volume 2, pages 1796–1801. IEEE, 2003.

[58] Junyan Hu, Parijat Bhowmick, Inmo Jang, Farshad Arvin, and Alexander Lanzon. A decentralized cluster formation containment framework for multirobot systems. *IEEE Transactions on Robotics*, 37(6):1936–1955, 2021.

[59] M Anthony Lewis and Kar-Han Tan. High precision formation control of mobile robots using virtual structures. *Autonomous robots*, 4(4):387–403, 1997.

[60] Wei Ren and Randal W Beard. Formation feedback control for multiple spacecraft via virtual structures. *IEE Proceedings-Control Theory and Applications*, 151(3):357–368, 2004.

[61] Jianan Wang and Ming Xin. Integrated optimal formation control of multiple unmanned aerial vehicles. *IEEE Transactions on Control Systems Technology*, 21(5):1731–1744, 2012.

[62] Dongdong Xu, Xingnan Zhang, Zhangqing Zhu, Chunlin Chen, and Pei Yang. Behavior-based formation control of swarm robots. *mathematical Problems in Engineering*, 2014, 2014.

[63] Dingjiang Zhou, Zijian Wang, and Mac Schwager. Agile coordination and assistive collision avoidance for quadrotor swarms using virtual structures. *IEEE Transactions on Robotics*, 34(4):916–923, 2018.

[64] Matthew Romano, Prince Kuevor, Derek Lukacs, Owen Marshall, Mia Stevens, Hossein Rastgoftar, James Cutler, and Ella Atkins. Experimental evaluation of continuum deformation with a five quadrotor team. In *2019 American Control Conference (ACC)*, pages 2023–2029. IEEE, 2019.

[65] Javier Alonso-Mora, Stuart Baker, and Daniela Rus. Multi-robot formation control and object transport in dynamic environments via constrained optimization. *The International Journal of Robotics Research*, 36(9):1000–1021, 2017.

[66] Javier Alonso-Mora, Eduardo Montijano, Tobias Nägeli, Otmar Hilliges, Mac Schwager, and Daniela Rus. Distributed multi-robot formation control in dynamic environments. *Autonomous Robots*, 43(5):1079–1100, 2019.

[67] Silvia Mastellone, Dušan M Stipanović, Christopher R Graunke, Koji A Intlekofer, and Mark W Spong. Formation control and collision avoidance for multi-agent non-holonomic systems: Theory and experiments. *The International Journal of Robotics Research*, 27(1):107–126, 2008.

[68] Cristian Vasile, Ana Pavel, and Cătălin Buiu. Integrating human swarm interaction in a distributed robotic control system. In *2011 IEEE international conference on automation science and engineering*, pages 743–748. IEEE, 2011.

[69] Jawad Nagi, Alessandro Giusti, Luca M Gambardella, and Gianni A Di Caro. Human-swarm interaction using spatial gestures. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3834–3841. IEEE, 2014.

[70] Guido Gioioso, Antonio Franchi, Gionata Salvietti, Stefano Scheggi, and

Domenico Prattichizzo. The flying hand: A formation of uavs for cooperative aerial tele-manipulation. In *2014 IEEE International conference on robotics and automation (ICRA)*, pages 4335–4341. IEEE, 2014.

[71] Nora Ayanian, Andrew Spielberg, Matthew Arbesfeld, Jason Strauss, and Daniela Rus. Controlling a team of robots with a single input. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1755–1762. IEEE, 2014.

[72] Yuyi Liu, Jan Maximilian Montenbruck, Daniel Zelazo, Marcin Odelga, Sujit Rajappa, Heinrich H Bülthoff, Frank Allgöwer, and Andreas Zell. A distributed control approach to formation balancing and maneuvering of multiple multirotor uavs. *IEEE Transactions on Robotics*, 34(4):870–882, 2018.

[73] Antonio Franchi, Cristian Secchi, Hyoung Il Son, Heinrich H Bulthoff, and Paolo Robuffo Giordano. Bilateral teleoperation of groups of mobile robots with time-varying topology. *IEEE Transactions on Robotics*, 28(5):1019–1033, 2012.

[74] Dongjun Lee, Antonio Franchi, Hyoung Il Son, ChangSu Ha, Heinrich H Bülthoff, and Paolo Robuffo Giordano. Semiautonomous haptic teleoperation control architecture of multiple unmanned aerial vehicles. *IEEE/ASME transactions on mechatronics*, 18(4):1334–1345, 2013.

[75] Rob A Holman, Katherine L Brodie, and Nicholas J Spore. Surf zone characterization using a small quadcopter: Technical issues and procedures. *IEEE Transactions on geoscience and remote sensing*, 55(4), 2017.

[76] Robin Deits and Russ Tedrake. Efficient mixed-integer planning for uavs in

cluttered environments. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 42–49. IEEE, 2015.

[77] MOSEK ApS. *MOSEK Optimizer API for C 10.0.26*, 2022. URL `https://docs.mosek.com/10.0/capi/intro_info.html`.

[78] Komei Fukuda. cddlib: C implementation of the double description method of motzkin et al. `https://github.com/cddlib/cddlib`, 2016.

[79] NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. Cuda, release: 8.0.61, 2017. URL `https://docs.nvidia.com/cuda/archive/8.0/`.

[80] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, 2020. doi: 10.1007/s12532-020-00179-2. URL `https://doi.org/10.1007/s12532-020-00179-2`.