FORMATION CONTROL OF UNMANNED AERIAL VEHICLES FOR DYNAMIC FORMATION RESHAPING TO ABRUPTLY ARISING ENVIRONMENTAL FEATURES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

SİNAN ÇİMEN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

JULY 2023

Approval of the thesis:

FORMATION CONTROL OF UNMANNED AERIAL VEHICLES FOR DYNAMIC FORMATION RESHAPING TO ABRUPTLY ARISING ENVIRONMENTAL FEATURES

submitted by **SİNAN ÇİMEN** in partial fulfillment of the requirements for the degree of **Master of Science** i**n Electrical and Electronic Engineering, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. İlkay Ulusoy
Head of the Department, **Electrical and Electronics Engineering** _____

Prof. Dr. Aydan Müşerref Erkmen
Supervisor, **Electrical and Electronics Engineering, METU** _____

Examining Committee Members:

Assoc. Prof. Dr. Emre Özkan
Electrical and Electronics Engineering Department, METU _____

Prof. Dr. Aydan Müşerref Erkmen
Electrical and Electronics Engineering Department, METU _____

Assist. Prof. Dr. Serkan Sarıtaş
Electrical and Electronics Engineering Department, METU _____

Assoc. Prof. Dr. Ali Emre Turgut
Mechanical Engineering Department, METU _____

Assoc. Prof. Dr. Muhammad Umer Khan
Mechatronics Engineering Department, Atılım University _____

Date: 24.07.2023

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name Last name : Sinan Çimen

Signature :

ABSTRACT

FORMATION CONTROL OF UNMANNED AERIAL VEHICLES FOR DYNAMIC FORMATION RESHAPING TO ABRUPTLY ARISING ENVIRONMENTAL FEATURES

Çimen, Sinan
Master of Science, Electrical and Electronics Engineering
Supervisor : Prof. Dr. Aydan Müşerref Erkmen

July 2023, 142 pages

A group of UAVs flying in formation in crowded environments must make decisions on changes to their formation in order to adapt to sudden emergence of new environmental features with minimal delays. In addition to typical tasks such as area coverage, exploration, and collective transportation, an alert formation control method is required to quickly adapt to new features. This approach builds upon an existing method that utilizes constrained optimization and multi-robot consensus to calculate formation parameters. We have enhanced this formation control method with a novel ability to split and merge in order to reconstruct the formation when disruptions occur during adaptation to environmental features. In this approach, the robots collectively decide on a direction to move, and then identify obstacle-free convex regions to determine a volume where the formation can fit, aligned with the chosen direction. Next, an optimization process is performed by considering the intersection of these regions with constraints imposed by the desired formation shape, in order to determine suitable formation parameters that align with the environmental conditions. Once the formation parameters are determined, target positions are assigned to each robot with the objective of minimizing the overall

mission completion time. The validity of the system is demonstrated through simulations conducted in the Microsoft AirSim environment.

Keywords: Multi-agent Systems, Formation Control, UAV, Obstacle Avoidance, Swarm Robotics

ÖZ

ANİDEN ORTAYA ÇIKAN ÇEVRESEL DURUMLARA GÖRE ŞEKİLLENEN DİNAMİK FORMASYONLAR İÇİN İNSANSIZ HAVA ARAÇLARININ FORMASYON KONTROLÜ

Çimen, Sinan
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği
Tez Yöneticisi: Prof. Dr. Aydan Müşerref Erkmen

Temmuz 2023, 142 sayfa

Kalabalık ortamlarda formasyon halinde uçan bir grup İHA, ani ortaya çıkan yeni çevresel özelliklere hızlı bir şekilde uyum sağlamak için formasyonlarını değiştirme konusunda kararlar almalıdır. Alan kaplama, keşif ve kolektif taşımacılık gibi tipik görevlere ek olarak, yeni özelliklere hızlıca adapte olabilen, alarm halinde bir formasyon kontrol yöntemi gerekmektedir. Bu yaklaşım, formasyon parametrelerini hesaplamak için kısıtlı optimizasyon ve çoklu robot uzlaşısı kullanan mevcut bir yöntemi temel almaktadır. Bu formasyon kontrol yöntemini, çevresel özelliklere adapte olma sırasında meydana gelen aksamalarda formasyonu yeniden yapılandırabilmek için ayrılma ve birleşme yeteneği ile geliştirdik. Bu yaklaşımda, robotlar birlikte hareket etmek için bir yön belirler ve ardından engelden arınmış konveks bölgeleri belirleyerek seçilen yöne hizalı bir hacim hesaplar. Sonrasında, istenen formasyon şekli tarafından belirlenen kısıtlarla bu bölgelerin kesişimini dikkate alarak optimizasyon süreci gerçekleştirilir ve çevresel koşullara uygun formasyon parametreleri belirlenir. Formasyon parametreleri belirlendikten sonra, her robotun hedef pozisyonu, görevin tamamlanma süresini en aza indirecek şekilde

atanır. Sistemin geçerliliği, Microsoft AirSim ortamında gerçekleştirilen simülasyonlarla gösterilmiştir.

Anahtar Kelimeler: Çok Elemanlı Sistemler, Formasyon Kontrolü, İHA, Engelden Kaçınma, Sürü Robotiği

To my mother

ACKNOWLEDGMENTS

I want to express my deepest gratitude to my supervisor Prof. Dr. Aydan M. Erkmen for her endless support, guidance and her belief in me. Her assistance has been helpful in every step of my research and thesis writing journey. Working with her has been a tremendous honor.

I would like to express my gratitude to the members of the dissertation committee for their valuable comments and suggestions, which have provided valuable insights and have contributed to further enhancing the quality of this study.

I want to thank my colleagues in Kuartis and Aerotim Engineering including Serdar Üşenmez, Ali Karakaya, Utku Yücel, Deniz Bardakçı and Ahmet Alp Yılmaz for everything I have learned from them. I would like to thank my team leaders Onur Tarımcı, Prof. Dr. İlkay Yavrucuk and Berker Loğoğlu for their valuable guidance in my career.

I want to thank my friends Haktan Bozkurt, Emre Ağkoç, Ozan Taş, Ekin Doğan, Göker Korel, Berk Ayduğan, Yusuf Eren Tunç, Onur Tekfiliz, Meriç Aydın, Özgür Akdağ, Alper Şanlı and Semih Gündüz for their support and patience throughout the process.

Thank you to İlay Korkmaz, for all her love and support.

Last but not least, I want to express my love to my family, my parents Jale and Mustafa, and my brother Süleyman for their endless support throughout my life; also, my nephew Deniz Taylan for shining to my life in the recent years.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGMENTS x

TABLE OF CONTENTS xi

LIST OF TABLES xiv

LIST OF FIGURES xv

CHAPTERS

1 INTRODUCTION 1

1.1 Motivation 3

1.2 Problem Definition and Objectives 7

1.3 Goals 8

1.4 Methodology 10

1.5 Contribution 12

1.6 Outline of Thesis 12

2 LITERATURE SURVEY 15

2.1 Formation Control Methods 15

2.1.1 Leader-follower Approach 19

2.1.2 Virtual Structure Approach 21

2.1.3 Behavior-based Approach 23

2.1.4 Artificial Potential Field Approach 24

2.1.5 Consensus-based Approach 26

2.1.6 Comparison and Our Approach 27

2.2 Obstacle-free Convex Regions 29
2.3 Constrained Nonlinear Optimization 32
2.4 Assignment Problems 35
3 PROPOSED METHOD 39
3.1 System Description 39
3.1.1 Direction Decision 40
3.1.2 Splitting and Merging 41
3.1.3 Obstacle-free Convex Region Generation 42
3.1.4 Formation Shape Generation 43
3.1.5 Formation Optimization 44
3.1.6 Assignment 45
3.1.7 Assumptions 45
3.2 Formation Shape Generation 47
3.3 Direction Decision 53
3.4 Splitting and Merging Conditions 59
3.5 Obstacle-free Convex Region Generation 70
3.6 Formation Optimization 78
3.6.1 Translational cost 83
3.6.2 Size cost 84
3.6.3 Rotational Cost 84
3.6.4 Switching Cost 89
3.7 Assignment to Target Positions 91
4 RESULTS AND DISCUSSION 95
4.1 Simulation Environment 95

4.2 Scenario 1 98
4.3 Scenario 2 112
5 SENSITIVITY ANALYSIS 127
6 CONCLUSION AND FUTURE WORK 133
6.1 Conclusion 133
6.2 Future Work 135
REFERENCES 137

LIST OF TABLES

TABLES

Table 1.1: Goals and requirements 9
Table 2.1: Comparison of formation control approaches 28
Table 3.1: Positions of UAVs, the formation center and the goal in the example case. 55
Table 3.2: UAV positions, center and target positions. 62
Table 3.3: Best angles of UAVs. 62
Table 3.4: UAV positions, center and target positions for the case with odd number of UAVs. 65
Table 3.5: Best angles of UAVs. 65
Table 3.6: Positions of UAV and intermediate waypoint. 71
Table 3.7: Obstacle centers and sizes. 72
Table 3.8: Target positions. 92
Table 3.9: Assigned target of each UAV. 93
Table 3.10: New assigned target of each UAV. 93
Table 4.1: Parameters for scenario 1. 101
Table 4.2: Path lengths of UAVs for scenario 1. 102
Table 4.3: Parameters for scenario 2. 115
Table 4.4: Path lengths of UAVs for scenario 2. 116
Table 5.1: Results for different translational cost weights. 128
Table 5.2: Results for different rotational cost weights. 129
Table 5.3: Results for different size cost weights. 130
Table 5.4: Results for different $\beta = wa/wd$. 131

LIST OF FIGURES

FIGURES

Figure 1.1: A team of quadrotors, (a) showing a square formation and (b) showing a line formation [8]. 5
Figure 1.2: A team of ground robots forming different letters [9]. 6
Figure 2.1: Two robots, leader shown in blue and follower shown in red, maintaining formation while avoiding obstacles [15]. 16
Figure 2.2: (a) Centralized control scheme (b) Distributed control scheme [16]. .. 18
Figure 2.3: Formation control approaches. 19
Figure 2.4: A leader-follower formation scheme with branches [47]. 21
Figure 2.5: Steps of navigation in a virtual structure approach [19]. 22
Figure 2.6: Simple behaviors for multiple robots (a) collision avoidance (b) velocity matching (c) flock centering [22]. 24
Figure 2.7: A graphical representation of the artificial potential field approach, with (a) magnitude and (b) direction of potential fields, where red crosses are agents and green cross is the goal position. [24]. 26
Figure 2.8: Top-view of an example obstacle-free convex polytope [6]. 30
Figure 2.9: An obstacle free polytope generated around a starting point shown as red dot in a 3D environment [38]. 32
Figure 2.10: A balanced assignment scheme dots on the one side represent robots and dots on the other side are targets. 35
Figure 2.11: Assignment of robots, shown as crosses, to targets, shown as circles, with two different objectives (a) LSAP (b) LBAP. 38
Figure 3.1: Overview of the formation control system. 39
Figure 3.2: Steps of the proposed method while navigating in formation. 40
Figure 3.3: Overview of direction decision process. 41
Figure 3.4: Overview of process of deciding upon splitting and merging. 42
Figure 3.5: Process of reaching consensus on obstacle-free convex region. 43
Figure 3.6: Process of shape generation. 43

Figure 3.7: Process of reaching optimization of formation configuration. 44
Figure 3.8: Process of assignment. 45
Figure 3.9: Coordinate system in an example scenario. 46
Figure 3.10: An example representation of (a) polygon and (b) lattice shapes with 9 agents, centered at the origin. 47
Figure 3.11: An example case with (a) initial positions of robots and (b) robots in formation after initialization of the algorithm. 48
Figure 3.12: Hexagon with circumscribed circle and vertices. 50
Figure 3.13: A team of four agents (a) before rotating, in XY cross section, (b) before rotating, in XZ cross section, (c) after rotating, in XY cross section and (d) after rotating, in XZ cross section. 52
Figure 3.14: The angle to the goal, α, where blue marks are agents and red cross is the goal position. 54
Figure 3.15: An example 2D case with 3 UAVs. 55
Figure 3.16: An example case where a team of 6 robots navigate around an obstacle (a) with splitting and (b) without splitting. 60
Figure 3.17: A team of 6 UAVs trying to pass an obstacle. 61
Figure 3.18: A team of robots (a) before splitting, (b) after splitting decision, (c) going around an obstacle while split and (d) merging again after passing the obstacle. 69
Figure 3.19: Difference of convex regions with respect to starting points, green area is the generated region, black area is the obstacle and red dots are starting points. 71
Figure 3.20: Example environment. 72
Figure 3.21: Convex regions generated by (a) UAV 1, (b) UAV 2 and (c) UAV 3. 74
Figure 3.22: The intersection of areas generated by 3 UAVs. 77
Figure 3.23: An example optimization case for a team of 4 UAVs. 80
Figure 3.24: The optimized formation configuration for an example weight set from (a) XY cross section and (b) YZ cross section. 86

Figure 3.25: The value of size parameter at each iteration for an example weight set. 86
Figure 3.26: The value of rotation parameter at each iteration for an example weight set. 87
Figure 3.27: The optimized formation configuration for the second example weight set. 88
Figure 3.28: The value of size parameter at each iteration for the second example weight set. 88
Figure 3.29: The value of rotation parameter at each iteration for the second example weight set. 89
Figure 3.30: A group of 6 UAVs (a) at starting position in polygon shape, (b) in a lattice formation to pass the obstacle (c) in polygonal shape again after passing the corridor. 90
Figure 3.31: A case where a team of 4 UAVs are in formation flight in a rectangular shape, with their current positions shown in black and calculated targets for the next step are colored in orange. 92
Figure 4.1: Simulation architecture. 98
Figure 4.2: An example screenshot of the environment of scenario 1. 99
Figure 4.3: Simplified presentation of the environment in scenario 1. 99
Figure 4.4: Paths covered by UAVs while executing scenario 1. 103
Figure 4.5: Example snapshots of convex regions generated by the team in scenario 1 at step = (a) 5, (b) 10, (c) 15, (d) 20 and (e) 25. 105
Figure 4.6: X positions of UAVs while executing scenario 1. 107
Figure 4.7: Y positions of UAVs while executing scenario 1. 107
Figure 4.8: Z positions of UAVs while executing scenario 1. 108
Figure 4.9: Splitting state of the team while executing scenario 1. 109
Figure 4.10: Size output of the algorithm at each planning step while executing scenario 1. 110
Figure 4.11: Rotation output of the algorithm at each planning step while executing scenario 1. 110

Figure 4.12: Minimum step length with respect to preferred velocity. 112
Figure 4.13: An example screenshot of the environment of scenario 2. 113
Figure 4.14: Simplified presentation of the environment in scenario 2. 113
Figure 4.15: Paths covered by UAVs while executing scenario 2. 117
Figure 4.16: Example snapshots of convex regions generated by the team in scenario 2 at step = (a) 6, (b) 16, (c) 26, (d) 36 and (e) 46. 119
Figure 4.17: X positions of UAVs while executing scenario 2. 121
Figure 4.18: Y positions of UAVs while executing scenario 2. 122
Figure 4.19: Z positions of UAVs while executing scenario 2. 122
Figure 4.20: Splitting state of the team while executing scenario 2. 123
Figure 4.21: Size output of the algorithm at each planning step while executing scenario 2. 124
Figure 4.22: Rotation output of the algorithm at each planning step while executing scenario 2. 125

CHAPTER 1

INTRODUCTION

Multi-robot or multi-agent systems became a popular research topic in robotics field in recent years. The main objective of this area is to achieve complex objectives using simple agents by utilizing the collaboration between them. One of the most important advantages of multi-robot systems is to eliminate the problem of single point of failure, which occurs when a system relies on a single component that, upon failure, causes the entire system to degenerate. Multi-robot systems overcome this problem by distributing the responsibilities and decision-making among multiple robots, so that if one robot fails, the system can still function through the actions of the remaining robots. This advantage is achieved when the multi-robot system is designed to be scalable in terms of number of robots. Another main advantage is the ability to have cost-effective solutions in cases where tasks to be achieved become expensive for single robot operations.

Multi-robot systems can be used to perform many tasks. And more importantly they can be beneficial in search and rescue for operations in human inaccessible areas. Similarly, they can be used for exploration, which can be completed faster by multi-robot teams than by a single robot. Surveillance is another usage area for multi-robot systems, for example for security purposes like intrusion detection. Collaborative construction is another operation that multi-robot teams perform efficiently [1]. Modular structures can be constructed by multiple robots more efficiently than a single robot. Recently, entertainment became a popular usage of multi-robot teams. Light shows performed by a team of aerial robots is an example of usage of multi-robot teams for entertainment [2].

Unmanned aerial vehicles (UAVs) are utilized in many multi-agent systems. UAVs, known also as drones, are aerieal vehicles that are either remotely controlled by a human pilot or flown autonomous without any operator. They can carry equipment like sensors, cameras or any other device to execute the task that they are designed for. There are several types of UAVs in terms of their configuration. There are fixed-wing UAVs, also known as unmanned airplanes, which generate lift from their wings by utilizing the forward speed of the aircraft. Fixed-wing aircrafts usually have longer ranges than other types of aircrafts but they mostly lack maneuverability and require a certain area for take-off and landing. Another type is single-rotor UAVs, commonly known as unmanned helicopters. The most common configuration for single-rotor UAVs is the one that consists of a main rotor and a small tail rotor. Advantages of this type are high maneuverability and ability of vertical take-off and landing. However, single-rotor aircrafts are usually quite complex and expensive to utilize in many areas and frequently encounter stability issues. Multi-rotor UAVs on the other hand, overcome this problem of complexity while keeping the advantages like high maneuverability and vertical take-off and landing. Also, because of their symmetrical design and redundancy of lift generating rotors, they overcome the stability issue. Multi-rotor UAVs, which are the vehicles focused in this thesis work, have more than two rotors that are generating lift and they are popular for commercial activities since they are relatively low cost. However, multi-rotor UAVs generally have limited flight times because of their high energy consumption and limited payload capacity. In this thesis work, quadcopters, which are multi-rotor vehicles with four rotors, are the vehicles to be used for the verification of the designed system since they are mainly preferred in tasks that require a formation control method like search and rescue, exploration and entertainment because of their ease of use and high maneuverability.

This thesis work focuses on designing a formation control system for a team of quadcopters, to safely navigate in obstacle-filled environments while having limited detection capabilities and maintaining formation shape. Formation control is the task

of maintaining a required geometric configuration while they advance through the environment. Formation control increases flexibility of the system since it may allow robots to reconfigure their formation while on mission, to adapt to a variety of environmental conditions. The main simulation environment used in this thesis is similar 1) to an urban environment, and 2) to a forest area where obstacles with different sizes and shapes are present like buildings and trees. Also, environments designed to emphasize specific features of the algorithm are used, such as a narrow passage to show shrinking ability of the formation. Formation control decreases chances of inter-robot collisions in the system since it tries to ensure that robots maintain a safe distance between them. Also, formation control can satisfy the communication requirements between agents for tasks that may need inter-robot communication like collective exploration or collective localization.

1.1 Motivation

Formation control behavior is a behavior that can be seen in nature in many different ways. An example is the flocking behavior of birds, which may enable them to fly in an aerodynamically favorable formation and may lead to less energy expenditure while traveling [3]. Another example is the schooling behavior of fish, to avoid predators and to locate food sources [46].

In multi-robot systems, there are several benefits of being in a formation, depending on the task assigned to the system. Being in a formation can lead to ensure a communication structure such that robots can either communicate with each other or with an external entity. Also, formation control can ensure that a group of robots covers an area which can be needed for tasks like exploration or searching for targets. Being in a formation also decreases the chance of inter-robot collisions; therefore, enabling a safer navigation.

In [8], a decentralized formation control scheme for a team of quadrotors is proposed to handle changing formation shapes by efficiently calculating collision-free

trajectories. In that method, formation is defined by shape vectors which are vectors between any two connected robots. Robots have the knowledge of shape vectors before executing the mission and plan their trajectories based on local interactions with other robots. They also propose a decentralized assignment method for safe switching of shapes. In order to move along a trajectory, the team needs to have a leader robot or a virtual member. Disadvantages in this approach are that all shape vectors need to be specified before the execution, which limits the adaptability of the formation. Also, the method proposed does not provide a collaborative obstacle avoidance scheme, instead each robot must handle obstacles on its own. With such an approach, the team can end up in a large separation between robots, which may distort the communication topology and end up in a mission failure.

(a)

(b)

Figure 1.1: A team of quadrotors, (a) showing a square formation and (b) showing a line formation [8].

In [9], another formation control method is developed, and tested with a team of ground robots. The aim is to achieve complex formation shapes while being only dependent on local interactions between robots. In this method, formation is defined as a point cloud and a directed acyclic graph is generated from the desired formation, with each node having a set of child nodes. This graph encodes two pieces of

information: the relative distance between a node and its parent nodes, and the orientation of the point with respect to the vector connecting its parent nodes. Then, using this graph, robots try to achieve the formation based on local interactions. Robots try to occupy a node in graph by asking robots occupying the parent nodes: If the robot is accepted by its parents, it goes to the assigned position. This method is able to work with different number of robots in the team, but it does not guarantee the uniformity of the formation in such a scenario as the formation shape is specified as a point cloud before the executing the mission. Also, it does not provide a navigation strategy, therefore, it only handles static formations.

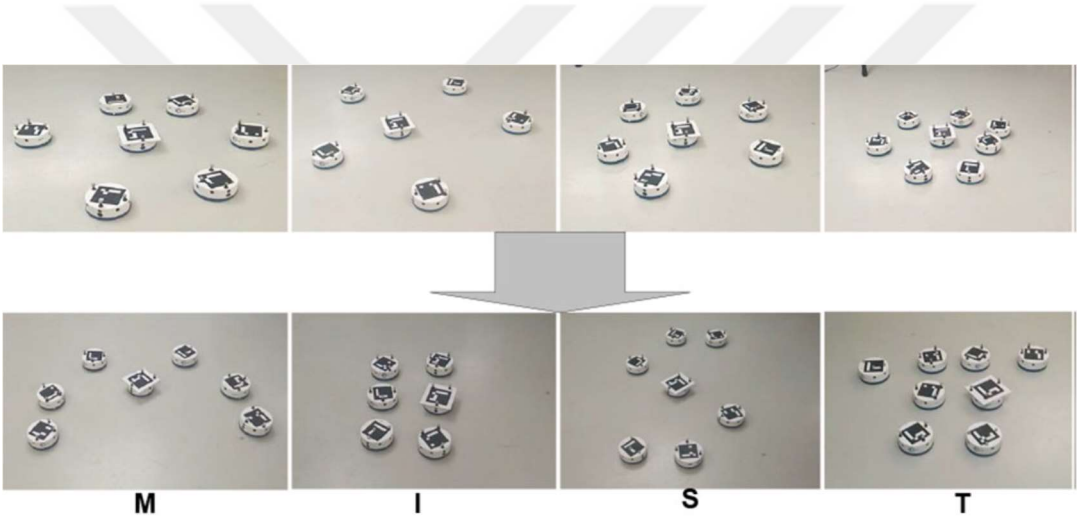


Figure 1.2: A team of ground robots forming different letters [9].

Multi-robot formation control problem can be treated by a central processing unit and such an approach greatly simplifies the problem since the central unit possesses the knowledge about the state of all agents and the environment. However, in many cases, a central unit may not be available. In that case, robots should be able to decide on the next step on their own, and should also be aware of the network status. Consequently, collaboration between robots is needed due to network aware control strategy of each robot, under its own task allocation approach. Another concern in

multi-robot systems is that typically robots in these systems have limited computational resources in order to solve formation control problem. Also, if the number of agents in the formation is enough and the task does not provide any restriction, formation should be able to split and then merge when it is feasible, in order to navigate more optimally in the environment according to changing features of the area encountered.

1.2 Problem Definition and Objectives

The main aim of our thesis work is to design a formation control system suited for quadrotors, that enables UAV formation to dynamically adjust its formation parameters, within bounds specified by the user in order to adapt to environmental conditions. Environment is a critical shaping entity for the determination of the formation. In this thesis, the algorithm is tested in an urban-like cluttered environment and also in a forest area which bring several challenges for the team as UAVs in the cluster may need to modify their path to pass through an obstacle, split to go around an obstacle, and shrink or rotate to pass a narrow passage. Whenever the corresponding challenge disappears, the formation has to recover its original status. Such behaviors bring the advantage of keeping connectedness of robots as much as possible.

Formation control algorithms that are able to achieve complex formations usually include a central computing unit, usually on-ground and stationary, to generate a waypoint for each robot. Being dependent on a central unit limits the system to work in a communication range such that all robots have to, at all times, communicate with the unit. For many applications like fire detection or surveillance for military purposes, ensuring a communication for all robots is hard to achieve because of restricted communication schemes and environmental configurations. Therefore, an approach that does not require a central unit is aimed in this thesis.

Many formation control algorithms in the literature focus on keeping steady, the formation that is formed before the algorithm is executed, and many are not capable of dynamically changing the formation while the mission is being executed. This results in limited mission capabilities. Such systems may end up in deadlocks while navigating since they cannot change their configuration. Therefore, our approach needs to consider different configuration options and select an appropriate one based on the environmental conditions.

In this work, we propose a dynamically shape changing formation control able to adapt to sudden obstacles of a cluttered environment during flight such as sudden appearance of trees to pass through or high rises to go around. To achieve this, UAVs collectively decide on a formation configuration while flying. This decision encompasses actions like splitting the formation, switching between different formation shapes, narrowing or widening the formation, rotating the formation or recovering to the original formation.

1.3 Goals

The major objective in the formation control problem is to achieve collaboration between robots in order to form a specific shape or cover an area. While doing these, several goals need to be addressed. Goals and their corresponding requirements of this work can be seen in Table 1.1.

Table 1.1: Goals and requirements

Goal	Requirement
Collision avoidance	Robots should avoid collisions while navigating.
Unknown environments	Robots should not need any prior knowledge about the environment.
Distributedness	The proposed approach should be able to be implemented in a distributed manner.
Dynamic formation	Robots should be able to manipulate the formation shape based on environmental features.
Minimal mission time	Robots should aim to minimize the mission time.

While navigating in a formation, in addition to keeping the formation shape, the system should avoid collisions. Collision avoidance can be handled with task allocation such as assigning individual roles within a geometrical shape.

Another objective that is considered in the design of this system is to be able to work in unknown environments by changing the shape of the formation. In this design, robots do not need any prior knowledge about the environment and the system we propose needs to be equipped with several decision makings on when to break the formation, when to reestablish the formation or change dynamically to another formation or even when the need to narrow down the same formation or enlarge it. Such decisions are the primary goals of the objective that are needed when flying in an uncertain cluttered environment. This gives the system flexibility to work in any kind of environment, especially cluttered.

The proposed approach should not require any central processing unit, since being dependent on a central unit or a leader robot limits the flexibility of the system and requires robots to be closer to either the central unit or the leader and leads to the single point of failure, where the entire system fail, due to failure of the central unit

or the leader. A decentralized robot network that is not dependent on dynamical capabilities of each single robot is a crucial aim of our problem.

In many approaches in the literature, the shape of the formation is limited to particular shapes [4, 5]. This limits the operational capabilities of the system since different tasks may need different formation shapes. In our approach, however, robots can decide and utilize any type of formation as long as they detect the required knowledge of the shape based on the environment structure during mission. This knowledge is the representation of the shape with variables like size and rotation. Each robot should be able to generate the required shape given the number of robots, size and rotational parameters.

In addition to the features discussed previously, robots should be able to manipulate the formation shape in order to safely navigate and keep the structure. In our work, robots are able to rotate the formation shape and change the size of the formation shape based on the need of the environmental passage features. Furthermore, robots need to be able to decide upon whether they should split into groups and then merge, according to the obstacles emerging in the crowded flight environment.

Another concern is the mission duration. Because of concerns like power consumption or having time-critical tasks, robots should minimize the mission time, which is reaching the target while navigating in a formation in an optimized flight time.

1.4 Methodology

The designed formation control method consists of several steps. While robots are navigating, the first step is direction finding. For a discrete set of angles, each robot calculates a cost associated with each angle that measures the risk of collision with obstacles in the environment. Robots have then to reach a consensus on the direction to move by communicating costs with each other. This information also serves as

one of the most important features of the method: deciding upon splitting and merging. If there appear two groups to move in separate directions, splitting decision is made and merging decision is executed whenever two separate groups desire to get closer to each other. Detailed conditions are explained in depth in Chapter 3.

To avoid collisions with obstacles, robots need to know the area that they can freely move inside. To achieve this, each robot calculates an obstacle-free convex region using Iterative Regional Inflation by Semidefinite Programming (IRIS) algorithm [6]. IRIS is a method to quickly compute large obstacle-free polytopic regions with a series of convex optimizations. Since each robot has different sensing capabilities, their knowledge about obstacles are different. To utilize this property, they reach a consensus about the safest area by exchanging their own calculated regions.

In this work, lattice and polygonal shapes are used as geometrical shapes for the formation. A set of equations are executed by robots to generate a set of vertices and a set of equality constraints defining the shape. These equations can generate shapes with respect to parameters like number of robots, size, and rotational configuration.

To enable robots to optimize their formation configuration by parameters like size and rotation, formation optimization process is defined. This is a nonlinear optimization problem with both equality and inequality constraints. This problem is solved through Sequential Quadratic Programming (SQP) [10]. This optimization process enables robots to adapt to environmental conditions like shrinking or rotating in narrow corridors, going around an obstacle by translating the waypoint, switching to different formation shape or restoring to original formation configuration if the risk ahead is cleared.

An assignment process is executed to minimize the mission time. The goal of this assignment is to minimize the maximum path length of robots. For quadrotors, the hover state, which means staying still in the air, is not an energy-efficient state. Therefore, most of the times, optimal mission time results in optimal energy consumption in such systems. Such a problem is called a bottleneck assignment

problem [11]. In this work, the assignment problem is solved using the framework of strong spanning trees [12].

1.5 Contribution

The main contribution of this work is a splitting and merging method for robot teams while navigating in a formation. This capability allows robots to navigate in separate teams when going around sudden obstacles where passages between them are too small for the whole formation to fit. This decision depends on the state of the formation at the time and distances of UAVs to the obstacles emerging during formation flight.

The splitting and merging capability is combined with a set of other behaviors like shrinking and enlarging the formation, rotating the formation and switching between different formation shapes in order to adapt to complex and crowded environments. The proposed novel control algorithm offers a combination of such behaviors and results in optimized performance while navigating.

1.6 Outline of Thesis

This thesis consists of 6 main parts. Chapter 1 introduces the definitions for key concepts like multi-robot systems, formation control and UAVs. After that, a problem definition is given to describe the aim of the method developed in this thesis. Then, objectives, goals and contribution of the thesis are given in this chapter.

Chapter 2 consists of a literature survey in which prior research on the formation control problem is discussed. Advantages and disadvantages of the method used in this thesis are also discussed in this section. Different aims and approaches of a variety of formation control methods are discussed in detail.

Chapter 3 proposes the theoretical background and the system breakdown of our method in detail. Since our work proposes a complete solution for the formation control problem, all steps used in the subsystems of our proposed method are given, illustrated with demonstrative examples.

Chapter 4 provides implementation results and discussions on the simulation environment. Parameter determinations, parameter bounds and assumptions are also discussed in this section, together with performance analyses of the method during implementation in different environmental cases.

Chapter 5, dwells on the sensitivity analysis of the method due to changing parameters. More specifically, the effects of parameters used in formation optimization process and parameters used in direction finding step are examined.

In Chapter 6, concluding comments upon the proposed method are given and possible future steps are discussed.

CHAPTER 2

LITERATURE SURVEY

Formation control is an adaptation of a robot colony to the environmental features. According to environmental character, the formation control has to decide how and when a shape has to be shrunk down or change the orientation while avoiding obstacles. In this thesis, UAV colonies are considered and a clear intelligent control carrying out a decision making to shape the formation according to environmental features is proposed. In the balance of this thesis, the formation control problem is handled through processes like formation shape generation and shape adaptation according to environmental features with obstacle-free convex region generation to find a free-to-move area for robots, minimization of a constrained nonlinear objective function in order to find an optimal formation configuration and task assignment of robots in the team according to a set of target positions which constitutes the desired formation shape. In this chapter, a general overview of the formation control methods existent in the literature is given relating to the need of our thesis work and to a review of subproblems handled in this thesis such as computing an obstacle-free area, finding an optimal formation configuration suitable to the environment and task assignment of robots.

2.1 Formation Control Methods

Formation control is a highly significant cooperation problem in multi-robot systems. Flight control with a formation enables the system to fly in a specific geometric shape to achieve advantages like safe navigation, keeping desired

communication structure and carrying out cooperative tasks while in formation flight.

Formation control problem include several critical issues such as deciding on a formation shape, achieving the decided formation, maintaining formation and switching between different formation shapes [13]. Decision making can be simplified into a leader follower approach such as [14] offers a method for maintaining the formation of mobile robots, where a designated leader robot guides the other robots, based on model predictive control. [15] proposes a set of artificial potential field functions to avoid obstacles and reach the goal for a team of robots using leader-follower strategy as depicted in Figure 2.1, where the leader robot is assigned to a target position and follower robot is trying to keep its position relative to the leader robot.

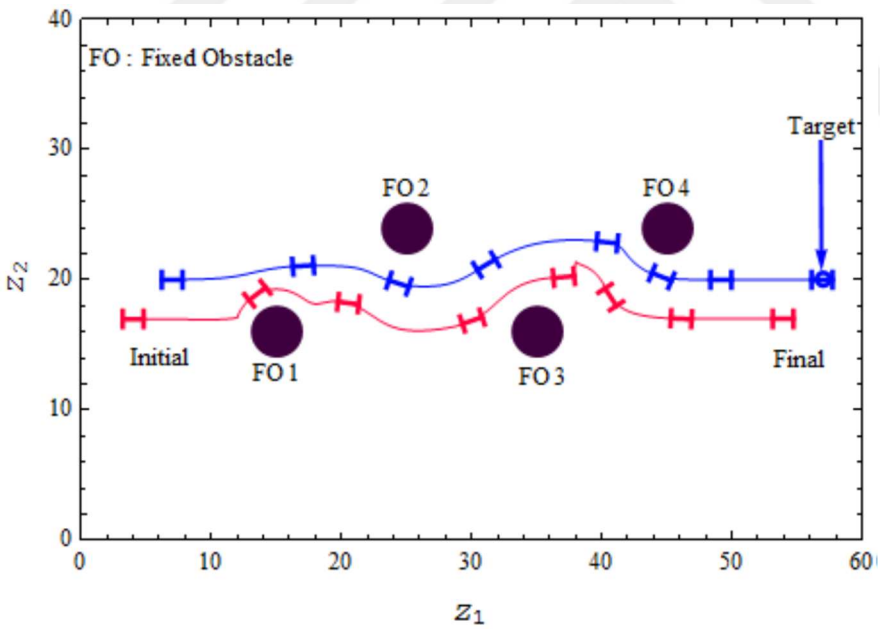


Figure 2.1: Two robots, leader shown in blue and follower shown in red, maintaining formation while avoiding obstacles [15].

The communication and computation structure of multi-robot systems are two of the most important aspects in these systems since such systems generally involve simple robots which have limited communication and computation capabilities. These structures may either be centralized or decentralized. In the centralized structure, a central processing unit is present and all computations are done by this unit. Central unit can be a base station on the ground or an agent with strong computational ability in the formation [16]. Robots communicate with the central unit from which they also receive commands. Centralized structure can guide the formation more optimally since all information is handled by one center. However, there are several limitations coming with this structure: The central unit needs high computational capabilities and needs to handle all communications with robots. Also, if the central unit is stationed on the ground, the range that the robot team can execute missions is limited by the reception distance. To overcome these problems, decentralized approaches are implemented to formation control problem. In decentralized control, each robot has the capability to decide on its own and robots are able to exchange information between each other. Implementation of the decentralized control scheme can be difficult, depending on the complexity of the mission, such as communication constraints present in the environment or the difficulty of dividing the computational needs to robots. The difference between a centralized control scheme and distributed control scheme can be seen in Figure 2.2, in which the ground station communicates with all UAVs in the centralized scheme and UAVs are not communicating with each other. Whereas in the distributed scheme, the ground station communicates with some of UAVs and they cooperate with each other. In the distributed scheme, communication with the ground station is generally limited to receiving high-level commands like the desired terminal position of the team.

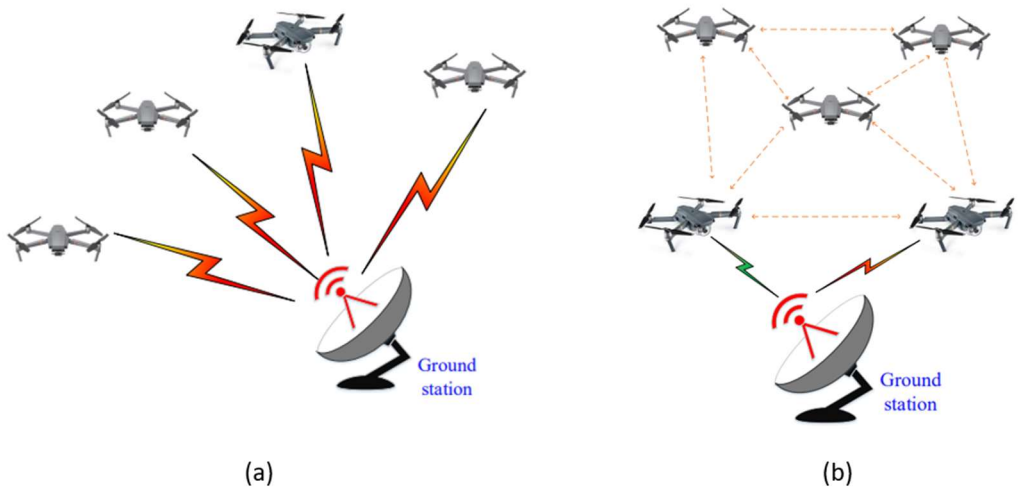


Figure 2.2: (a) Centralized control scheme (b) Distributed control scheme [16].

There are several well-known strategies to handle formation control problem, such as leader-follower strategies, virtual structure approaches, artificial potential field approaches, behavior-based approaches, and consensus-based approaches. The diagram of these approaches can be seen in Figure 2.3. Advantages and disadvantages of each of these approaches are discussed in this chapter and our motivation to handle these disadvantages in this thesis is given.

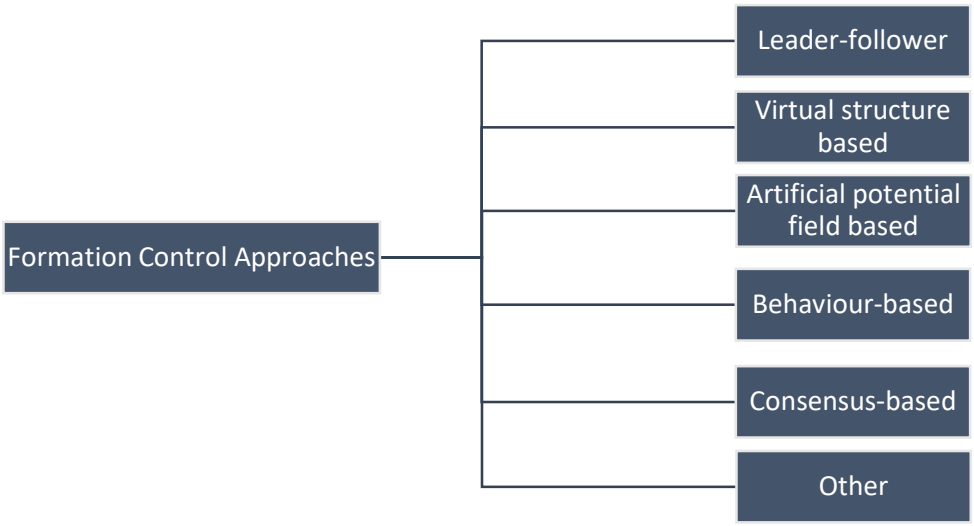


Figure 2.3: Formation control approaches.

2.1.1 Leader-follower Approach

In the leader-follower approach to formation control, a designated leader robot is used to guide the other robots in the group to maintain the desired formation shape. The leader robot is responsible for determining the desired formation shape and generating reference signals for the follower robots to follow. The follower robots use these reference signals to adjust their positions and orientations relative to the leader robot and maintain the desired formation shape.

The leader-follower approach generally is a decentralized control strategy, where each robot in the group is only responsible for orienting itself with the leader robot, rather than communicating with and coordinating with all the other robots in the group. This can make the leader-follower approach more scalable and easier to implement in large groups of robots. However, performance of the leader-follower methods depends on the robustness of the communication between leader and followers. This property can restrict the size of leader-follower approaches, but it is possible to overcome the restriction by having multiple leaders or virtual leaders.

In the literature, there are mainly two types of control for leader-follower approach. These are l- φ controller and l-l controller [17]. In l-φ, also called separation-bearing control, there are one leader and one follower robot in consideration. The aim is to maintain the desired distance between two robots, l_d and the desired angle between two robots, φ_d. In l-l, also called separation-separation control, there are two leader robots and one follower robot. The aim is to maintain distance between first leader robot and follower robot, l_1 and distance between second leader robot and follower robot, l_2.

[18] proposes a l-φ type leader-follower based formation control scheme of multiple ground robots. A kinematic model for leader-follower scheme is given in Cartesian coordinates and a controller using the idea of integrator backstepping is designed. In [47], a cascaded guidance law for multiple UAVs is proposed in a leader-follower scheme. In this method; a global leader, branches and local leaders of these branches are present, as it can be seen in Figure 2.4 where an example case of with these elements is given. The guidance law is verified via Lyapunov stability theory and guaranteed synchronization of formation flight is shown. Such methods bring the vulnerability of being dependent on leaders which causes failure of the system when these elements fail. In our approach, there is no such hierarchical relationship between robots.

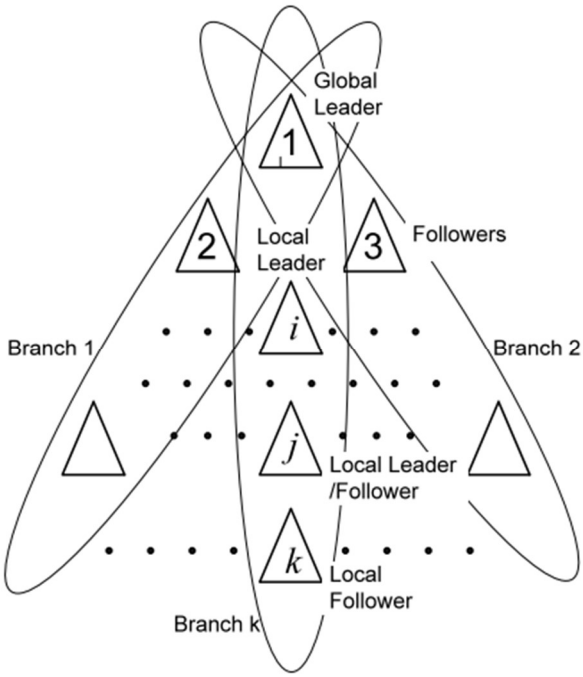


Figure 2.4: A leader-follower formation scheme with branches [47].

2.1.2 Virtual Structure Approach

In the virtual structure approach to formation control, virtual structures that are used to represent the desired formation shape of a group of robots, can be thought as virtual connectors that connect the robots in the group and define the desired relative positions and orientations between them.

To maintain the desired formation shape, robots in the group use control algorithms to adjust their positions and orientations relative to the virtual structures, which are not physical entities and are only used as a reference for the robots to follow. Virtual structure method can be implemented in a decentralized manner, where agents need to track their relative position to other robots. It is much easier to implement decentralized virtual structure control strategy when dealing with fixed structures; however, in such a decentralized approach, some problems arise when considering

time-varying formation shapes since robots in such systems need to agree on a suitable formation shape.

The virtual structure approach can be used to maintain a variety of different formation shapes, including linear, circular, and more complex shapes. It can also be used to handle external disturbances and uncertainties, by adapting the virtual structures in real-time to compensate for these effects.

In [19], a simple method based on virtual structure approach to formation control problem is proposed. In this method, robots act like fixed points on a rigid structure. There are mainly 4 steps involved in this approach and those steps can be seen in Figure 2.5. First, a virtual structure is generated based on current positions of robots, labeled as step 1 in the figure. Then, the virtual structure is displaced by some desired angle and linear distance, labeled as step 2 in the figure. After that, in step 3, individual robot trajectories are generated by the new pose of the virtual structure. Finally, in step 4, robots are commanded to track these trajectories. [20] proposed an enhanced virtual structure based formation control method for multiple UAVs. In this method, they integrated synchronization of relative positions between UAVs and results showing better topology maintanance performance are given. These methods, however, are designed to maintain a specific formation, which limits the adaptability of the system to changing environmental conditions. Our method, on the other hand, can dynamically modify the formation shape to adapt to environmental conditions.

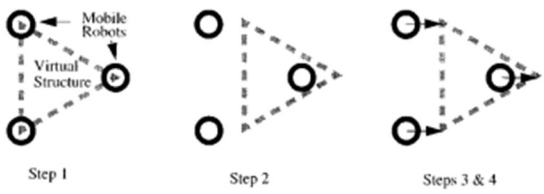


Figure 2.5: Steps of navigation in a virtual structure approach [19].

2.1.3 Behavior-based Approach

In the behavior-based approach to formation control, the robots in a group are programmed to exhibit certain behaviors that contribute to the overall formation control of the group. These behaviors can be simple, such as moving in a particular direction or maintaining a certain distance from other robots, or more complex, such as avoiding obstacles or adapting to changing conditions in the environment.

The behavior-based approach is usually a decentralized control strategy, where each robot in the group is responsible for exhibiting its own behaviors and does not need to communicate with or coordinate with the other robots in the group. This can make the behavior-based approach a simple and scalable approach for formation control, particularly in large groups of robots.

However, the performance of the behavior-based approach may be affected by the accuracy and reliability of the behaviors exhibited by the robots, as well as the ability of the robots to adapt to changing conditions in the environment. Since they are constructed by a combination of simple behaviors, covering all needs of a dynamic environment is highly difficult. In our approach, because of being predictive by its nature, formation is allowed to reconfigure itself and adapt to environmental changes more quickly and safely.

In [21], a Null-Space-Based behavioral formation control method is proposed. In this method, simple behaviors for robots are defined such as keeping the formation shape with the neighboring robots, avoiding obstacles and moving to target location. A priority rule is defined for these behaviors and combinations of these behaviors are simulated achieve flocking in a crowded environment. [22] proposes a similar approach for flocking of multiple aerial vehicles. In Figure 2.6, a representation of basic behaviors used in this method such as collision avoidance, velocity matching and flock centering is given. In collision avoidance behavior, the aerial vehicle in consideration tries to get away from other vehicles, whereas in flock centering behavior, it tries to move to the center of neighboring vehicles. Therefore, the

combination of these two behaviors results in moving as a group while having a safe distance between vehicles. The velocity matching behavior tries to match velocity vectors of all robots, in order to maintain the formation structure while navigating. In addition to basic behaviors for flocking, more behaviors for complex maneuvers are defined such as seek, flee, pursue and evade. These behaviors add capabilities of observing a stationary target while being radially aligned with it, tracking a moving target and evading the moving target.

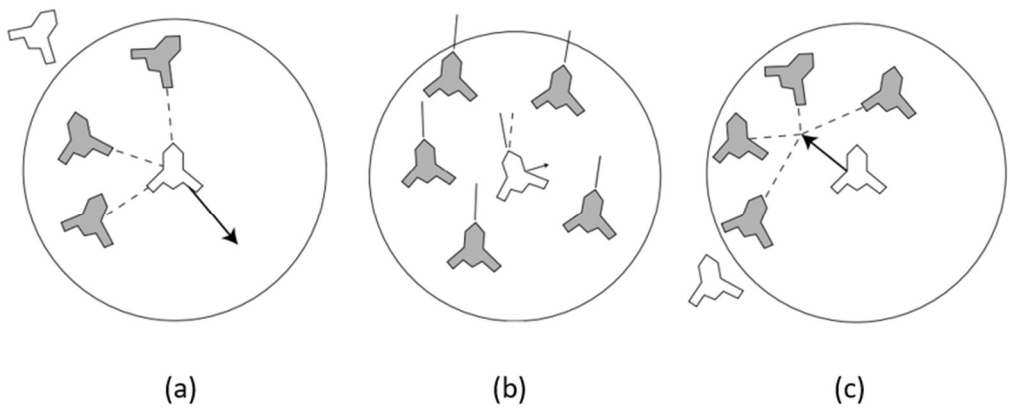


Figure 2.6: Simple behaviors for multiple robots (a) collision avoidance (b) velocity matching (c) flock centering [22].

2.1.4 Artificial Potential Field Approach

The artificial potential field approach implemented to formation control, guides robots to accomplish tasks such as collision avoidance or formation maintenance. The artificial potential field is a mathematical function that defines a set of attractive and repulsive forces acting on the robots in the group. Such that, attractive forces guide robots towards the desired formation shape, while repulsive forces prevent robots from colliding with each other or obstacles in the environment.

To maintain the desired formation shape, robots in the group use control algorithms to navigate through the artificial potential field and adjust their positions and orientations relative to the desired formation shape. Mostly, the artificial potential field approach is a decentralized control strategy, where each robot in the group is only responsible for navigating through the artificial potential field and does not need to communicate with or coordinate with the other robots in the group. However, this is only true when the team is navigating in a known environment. For unknown environments, robots need to sense and identify potential field sources and such a task may need a collaboration between robots in order to obtain more information about the environment by sharing identified sources with each other. This procedure is generally called cooperative detection or sensing.

In [23], a potential field-based formation control algorithm which is inspired from electric fields is proposed. In this method, potential functions are based on distance between agents and distance between agents and obstacles. Then, a state feedback control law is utilized for each agent. The designed algorithm is decentralized and analyses were made under assumptions such that obstacles are considered static and goal position is chosen at a safe distance from obstacles. In [24], an algorithm utilizing the potential field method in combination with virtual leader method is proposed. In this algorithm, there are 4 actors effecting the overall potential function. These are; influence of virtual leader to track navigate on the path, formation influence to keep the desired formation by considering distances between agents, inter-agent collision factor to prevent collisions between agents and obstacle avoidance factor. The algorithm is simulated with a group of UAVs. In Figure 2.7, potential fields in an example scenario can be seen. Agents emit repulsive potential forces in order to ensure collision avoidance and the goal position acts as a sink in order to attract agents. These methods do not guarantee maintaining the formation topology since agents act as individuals, not as a group. In our approach, robots reach on a decision about the state of the group; therefore, they always try to maintain the shape of the formation.

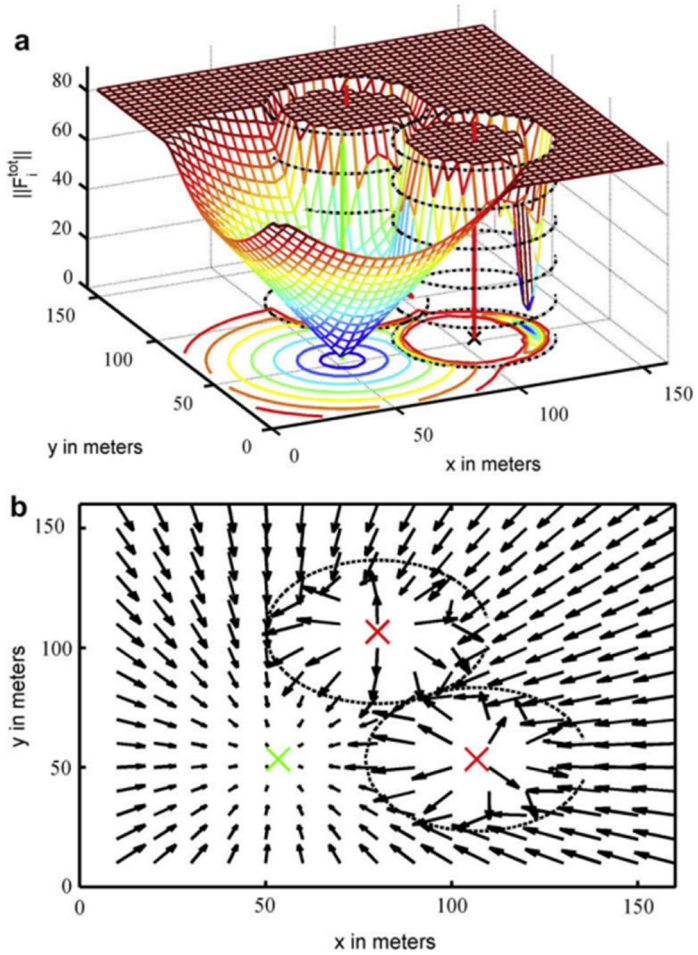


Figure 2.7: A graphical representation of the artificial potential field approach, with (a) magnitude and (b) direction of potential fields, where red crosses are agents and green cross is the goal position. [24].

2.1.5 Consensus-based Approach

In the consensus-based approach to formation control, robots in a group use communication and coordination to reach a consensus on parameters about the navigation plan such as the desired formation shape that fits environmental conditions and keeping it stable under environmental disturbances. This can be achieved through the exchanging states of robots with each other, and the use of control algorithms to adjust the positions and orientations of the robots based on this

information in order to align themselves with the rest of the group, based on mission requirements.

The consensus-based approach is a decentralized control strategy, where each robot in the group is responsible for communicating and coordinating with the other robots in the group in order to reach a consensus on the parameters about the formation. This can make the consensus-based approach a flexible and scalable approach for formation control, particularly in large groups of robots.

However, the performance of the consensus-based approach is affected by the accuracy and reliability of the communication and coordination between the robots. It often becomes challenging to design and implement the consensus-based approach compared to other approaches (the virtual structure or leader-follower approaches), since agents must reach on a common agreement on the desired formation through iterative information exchange and cooperative decision-making rather than relying on a predefined structure or a designated leader.

[25] proposes a consensus-based control law for a team of ground robots. The method allows robots to navigate and avoid obstacles cooperatively, with a simplistic single integrator model in which control actions determine the velocity of robots directly. [26] designs a formation protocol for a team of quadrotors. In this method, time-varying and predefined formation shapes are considered. Necessary and sufficient conditions for robots to carry out the formation flight and handle time-varying formations are given, such as reaching on a common velocity vector and having an asymptotically stable formation error system.

2.1.6 Comparison and Our Approach

A comparison of the formation control approaches discussed previously is given in Table 2.1.

Table 2.1: Comparison of formation control approaches

Control Approach	**Advantages**	**Disadvantages**
Leader-follower	• Ease of implementation • Scalable	• Single point of failure
Virtual Structure	• Ease of implementation	• Less flexibility
Behavior-based	• Emergent behavior • Flexibility	• Hard to tune • Design complexity
Artificial Potential Field	• Fast responsiveness	• Local minimas • Design complexity
Consensus-based	• Decentralized • Scalable	• Communication overhead • Design complexity

The approach presented in this thesis is an optimization-based method while some steps in the algorithm are based on reaching consensus among the team. Although consensus-based formation control algorithms struggle to exhibit complex behaviors since they are generally focused on low-level control of robots, our approach utilizes consensus for high-level planning requirements such as deciding on a common direction to move or generating an obstacle-free region in the environment to navigate inside of it. We handled the formation control problem in a similar way to [48]; with additional capability of splitting and merging, and with different target assignment scheme. [50] proposed a formation control algorithm which can manipulate the formation size and has splitting and merging capability, using distributed model predictive control and improved alternating direction multiplier method. However, the method is restricted for 2D cases and validated with a team of ground vehicles. [51] proposed a distributed formation control algorithm which satisfies collision avoidance between UAVs

with a repulsive function based on Hooke's law with damping and obstacle avoidance with a splitting and merging strategy. However, in this method, UAVs are not able to manipulate the formation shape in terms of its size and orientation. [49] also proposed a splitting and merging method for an approach similar to ours, however, it only considers splitting when the team is not able to find a suitable configuration without splitting. In our method, splitting can also occur when the team is able to navigate together, if such a decision can lead to more efficient navigation. For example, when an obstacle is encountered that is nearly aligned with the center of the formation and the team is able to pass the obstacle without splitting, splitting into two groups and going around the obstacle from different sides leads to a more optimal solution.

2.2 Obstacle-free Convex Regions

In a crowded environment, it is crucial to sense and react upon environmental changes quickly. In our approach, robots generate regions in which they can safely navigate without colliding with obstacles. Therefore, an efficient method for computing obstacle-free regions is needed. Since reaction to environmental changes needs to be fast, we require these regions to be convex as it enables efficient computing in the later steps of the algorithm. Convex functions has a unique, global minimum; therefore, it is generally easier to optimize them. In 2D, convex regions can be represented as polygons consisting of a set of lines. In 3D, these regions can be represented as polytopes consisting of a set of separating planes. In this thesis, we use a colony of UAVs flying in formation in a crowded environment and we critically need an efficient method to generate obstacle-free convex polytopes. In addition to being efficient, the regions should be as large as possible because methods that are too conservative may lead to inefficient path planning. For example, having a very small region may prevent UAVs from advancing and reaching their goal. In our approach, each robot generates a convex region based on their own sensing capabilities and these regions are shared with other robots. This is essential since some robots may not detect an environmental feature while others are able to sense

it. This way, robots are able to combine their sensing capabilities and have more information about the environment. This process is described in details in Chapter 3.

The problem of generating an obstacle-free convex region is given below. From detected environmental features, obstacles, and a point around which an obstacle-free region is needed, the aim is to find a convex polytope which does not include any obstacles in it and is maximum in terms of size. In Figure 2.8, a top-view representation of this problem is given. A starting point is shown in green and a set of obstacles shown as black polygons, a region, shown in red, is given which is the top-view of the polytope we aim to find. Note that, the polytope does not have any obstacles inside it but edges are tangent to obstacles since the volume of the polytope is desired to be maximum.

Problem: Given a set of obstacles G and a starting point P, find a maximum convex polytope represented with a set of linear inequalities $Ax \leq b$.

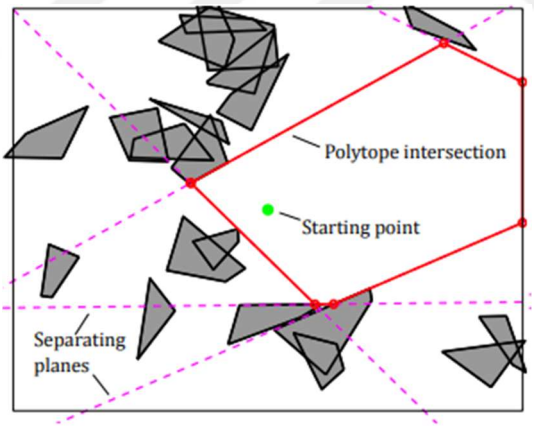


Figure 2.8: Top-view of an example obstacle-free convex polytope [6].

[37] proposes an algorithm to find maximum convex polygons in an area. The algorithm requires a starting point, which is an interior point or on the boundary of the generated region, a set of points which represents the area and labels of these

points as “positive” or “negative”, where positive means the point is obstacle-free and negative means the point is in an obstacle. The algorithm works in 2D space and requires discretization of the area; therefore, it is not suitable for our case.

In [38], another approach to generate convex obstacle-free regions is proposed, based on stereographic projection, a technique to map a sphere or a portion of a sphere onto a plane. This approach works with point clouds and it is applicable for 2D and 3D environments. In Figure 2.9, an example result of the method is given. In a point cloud represented by blue dots and a starting point given by a red dot, an obstacle-free polytope is generated around the starting point. [39] proposes a method for a similar problem, which is generating large convex regions based on point clouds in 3D. The method is based on mapping a set of unordered points to a nonlinear space which is an invertible nonlinear transformation. These methods are not iterative, which restricts us from having a trade-off between solution optimality and computation time. On the other hand, [6] develops a method to compute polytopic and ellipsoidal obstacle-free regions iteratively. The method begins with a starting point and expands the region at every iteration using quadratic programming and semidefinite programming. The algorithm is able to work with 2D or 3D spaces and also with higher dimensions. Since this approach is iterative, it is possible to obtain solutions that are less optimal but with less computation time. Therefore, depending on mission requirements about reaction time to environmental changes, such that short computation times are required, it is possible to tune the algorithm. This ability can be quite beneficial for real-time implementations of our system in future. Therefore, in our approach, we utilized the method given in [6].

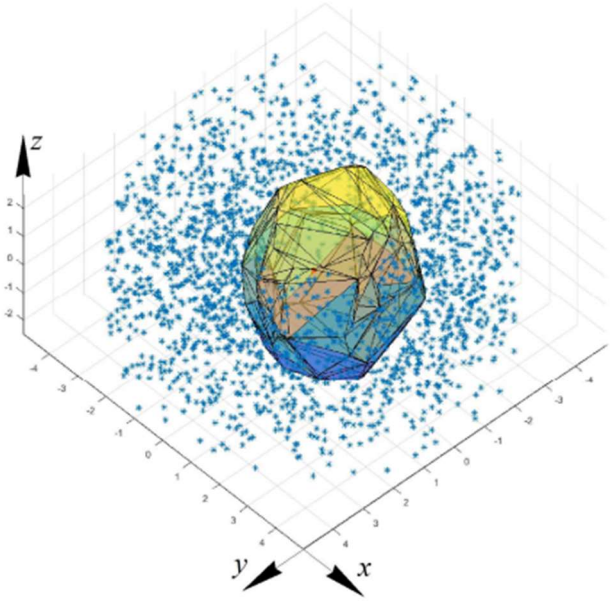


Figure 2.9: An obstacle free polytope generated around a starting point shown as red dot in a 3D environment [38].

2.3 Constrained Nonlinear Optimization

In this thesis, to determine a safe formation configuration in accordance with changing environmental features such as suddenly appearing obstacles, a nonlinear optimization problem is solved to obtain formation parameters like the size of the formation or the orientation of the formation while reacting to those environmental changes. To avoid suddenly appearing obstacles, we utilize obstacle-free convex regions generated by robots in a way that they constrain positions of robots to be inside. This way, it is guaranteed that the target formation does not intersect with any obstacles. Therefore, this constraint is present in the optimization problem as an inequality constraint. There are also equality constraints which dictate relative positions of agents to preserve the formation shape. This is a classical optimization problem for which many methods are developed, but, since every agent runs the

optimization process at each timestep, the optimization algorithm should be computationally fast in order to react to environmental changes quickly, before a collision may occur. Each robot runs the optimization process on their own after reaching on consensus about obstacle-free regions. We require robots to reach the same resulting formation configuration which is the result of the optimization problem, and this is expectable since they execute the optimization problem with the same knowledge on objective function and constraints. However, probabilistic optimizers like genetic algorithms do not guarantee to reach the same result with the same problem since they involve sampling from random variables. Therefore, we need a deterministic solver for this problem and probabilistic approaches like genetic algorithms are not feasible for such our case.

A general representation of optimization problem in this thesis can be given as

$$\begin{aligned} &\min_{x} f(x) \\ &subject\ to\ Ax = b \\ &\qquad\qquad Cx \leq d \\ &\qquad\qquad x_{min} < x < x_{max} \end{aligned}$$

where x is the vector of parameters to be optimized consisting of target positions of robots, the size of the formation and the orientation of the formation. $f(x)$ is the nonlinear objective function which, in our approach, penalizes the system by the amount formation is translated, shrunk and rotated in a quadratic form. A and b are coefficients of equations that constrain robots to be in a specific shape. C and d are coefficients of equations of hyperplanes which constitute the obstacle-free convex region that robots are required to be inside. x_{min} and x_{max} are bounds of parameters to be optimized determined by bounds of the environment for target positions and minimum size of the formation to have a safe distance between robots.

The Gradient Projection Method [40] can be used to minimize a continously differentiable objective function since it utilizes gradient of the objective function to find a search direction in which the objective function is decreasing. The basic idea behind the method is to project the current iterate onto the feasible set, which is the set of all points that satisfy constraints of an optimization problem, at each iteration, in order to ensure that the iterate remains feasible. The method requires the choice of suitable step size or a line search algorithm along the search direction to converge to a local minimum and it is sensitive to initialization. Another popular technique is to utilize penalty functions which are quantifications of constraint violations. These functions are integrated into the objective function to eliminate constraints and convert the problem to an unconstrained nonlinear optimization problem. Therefore, the problem becomes simpler to solve since unconstrained optimization problems require less effort. Many approaches to construct the penalty function are proposed in the past [41][42]. Sequential Quadratic Programming [44] is an iterative method which uses quadratic programming subroutines to obtain a new search direction at each iteration. The method starts with linearizing the equality and inequality constraints. Then the linearized version of the problem is solved with a quadratic programming algorithm to obtain the direction of the new iterate and the Lagrange multiplier. Then, it executes a search through that direction. After computing the new iterate with the search direction found by the quadratic subproblem and step size found with the line search in that direction, the algorithm begins a new loop with the new iterate and continue until a given stopping criteria such as the maximum number of iterations or a minimum improvement in the objective function is met. The complete method described in [45] is utilized in this work in order to find an optimal formation configuration since it outperforms gradient-based methods and penalty function based methods in many constrained nonlinear optimization problems similar to the one considered in this thesis [43].

2.4 Assignment Problems

The assignment problem deals with how to distribute tasks among many agents. This distribution is done by minimizing the objective function. The reason we want to minimize the objective function is to find the most efficient assignment of resources that minimizes the overall cost. In some cases, the objective function represents a measure of distance, time, or effort required to complete the tasks. The assignment problems are generally classified into two types: balanced assignment problems where number of tasks is equal to number of agents, unbalanced assignment problems where number of tasks is not equal to number of agents [28].

In this thesis, n target positions, which constitute the target formation shape, are generated after deciding on the formation parameters, such as the size and orientation, at each timestep, for a team of n robots. The assignment problem deals with how to assign n target points to n robots. The resulting assignment should be in a way that each robot is assigned to one unique target since robots are going to form the desired shape. Since the number of target points is equal to the number of robots, the problem is a balanced assignment problem, such as the case shown in Figure 2.10.

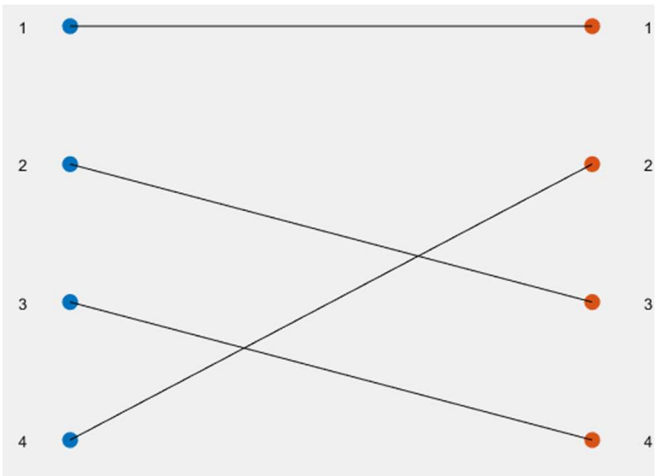


Figure 2.10: A balanced assignment scheme dots on the one side represent robots and dots on the other side are targets.

Depending on the objective function, assignment problems can be categorized as linear assignment problems, quadratic assignment problems and multi-index assignment problems [27]. In a linear assignment problem, the objective is to minimize or maximize a linear function of the assignment variables; whereas, in quadratic assignment problems, a quadratic function is considered. In a multi-index assignment problem, the objects and targets are indexed by multiple indices, like a timetabling problem where assigning rooms to events for each time slot is required. In this thesis, single index problem is present since at each step, we are only interested about assigning robots to targets for that time instance.

If we define c_{ij} as the cost of assigning robot i to target j; then, nxn cost matrix becomes

$$C = \begin{bmatrix} c_{11} & \cdots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{n1} & \cdots & c_{nn} \end{bmatrix}$$

The goal in the assignment problem is to pick n elements from the cost matrix in a way that there is exactly one element selected from each column and from each row since each robot needs to be assigned to a unique target position. There are mainly two types of linear assignment problems based on the goal of the assignment. Linear Sum Assignment Problems (LSAP) deals with minimizing the sum of costs selected in the assignment. From a multi-robot system perspective, the aim of LSAP is to minimize the total path taken by robots, if costs are defined as distances between robots' positions and target positions. If we define a logical matrix $S = (s_{ij})$ that represents the result of the assignment as

$$s_{ij} = \begin{cases} 1 & \text{if robot } i \text{ is assigned to target } j, \\ 0 & \text{otherwise} \end{cases}$$

Then, LSAP optimization problem, which tries to minimize the total cost incurred from the resultant assignment scheme, becomes,

$$\min \sum_{i=1}^{n} \sum_{i=1}^{n} s_{ij} c_{ij}$$

where c_{ij} is the distance between i^{th} robot and j^{th} target. There are several methods developed to solve LSAP. The Hungarian algorithm [29] is a well-known algorithm that solves the LSAP in polynomial time. There are also simplex method based solutions [30] and cost operator algorithms [31] to solve LSAP. All of these methods find optimal solutions LSAP problems, in different time complexities. In the literature, there are many formation control methods that treat assignment problem as an LSAP [32][33]. Since the selected assignment objective is LSAP in these methods, they try to minimize the total path of robots. However, such an approach can lead to longer mission times.

Another type of linear assignment problems is Linear Bottleneck Assignment Problems (LBAP). The goal of LBAP is to minimize the mission completion time. From a multi-robot system perspective, it can be seen as minimizing the maximum path length travelled by a robot. The LBAP objective function can be defined as

$$\min \max_{1 \leq i,j \leq n} s_{ij} c_{ij}$$

Many methods are developed in order to solve LBAP. Some approaches are threshold algorithms [34] with computational complexity $O(n^4)$ for the case of n agents and n tasks, augmenting path algorithms [35] with computational complexity $O(n^4)$, strong spanning trees approach [12] with computational complexity $O(n^3)$ and combination of threshold method with augmenting paths [36] with computational complexity $O(n^3)$. Since it is easier to implement, strong spanning trees approach is utilized in this thesis.

In Figure 2.11, an example assignment scheme, where optimizing with objectives LBAP and LSAP gives different results, is given. Robots are shown as crosses and

targets are shown as circles. The goal is to assign each robot to a unique target position. On the left, the resultant assignment from selecting the LSAP objective is given and on the right, the case with LBAP objective is given. Lines connecting robots to targets mean that the target is assigned to the connected robot. For both cases, optimal results are given. The method in [29] is used to solve the LSAP case and [12] is used to solve the LBAP case. If the LSAP objective is selected, assignment result in a total path length of 21.56m and maximum length of the path taken by an agent is 12.53m, whereas LBAP objective results in a total path length of 22.18m and maximum length of the path taken by an agent is 8.49m. Although using LBAP objective increases the total path, it is able to decrease overall mission time drastically.

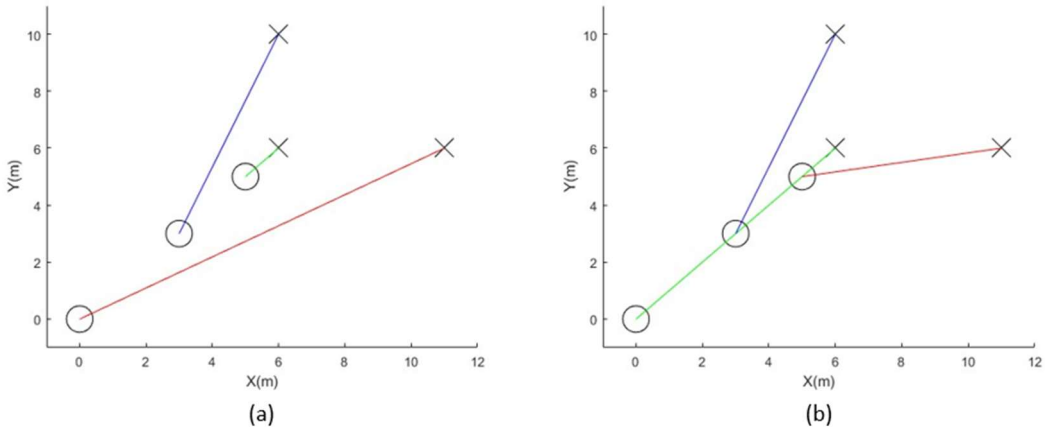


Figure 2.11: Assignment of robots, shown as crosses, to targets, shown as circles, with two different objectives (a) LSAP (b) LBAP.

CHAPTER 3

PROPOSED METHOD

3.1 System Description

In this thesis, a complete formation control method which acts as a local motion planner is proposed. In Figure 3.1, a high-level overview of the centralized system is given. This diagram starts with a goal assigned to the local motion planner. Then, the focus of this thesis, the local motion planner receives sensor data from environment such as distances to obstacles and robot positions from the low-level controller to compute a feasible waypoint for each robot while avoiding collisions. Then, these waypoints are tracked with the low-level controller by computing control inputs for each UAV.

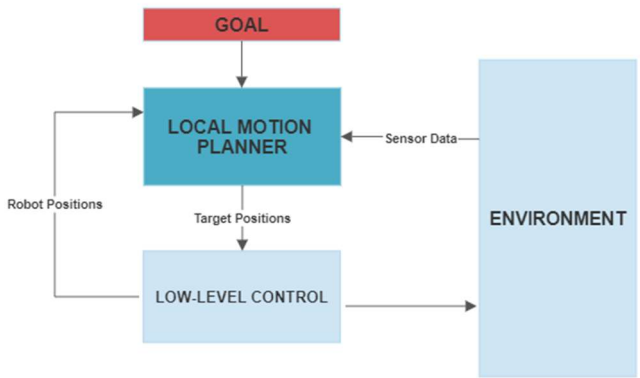


Figure 3.1: Overview of the formation control system.

The proposed local motion planner has several steps in order to compute a suitable target formation configuration according to environmental features, shown in Figure

3.2 in the order of execution. These steps are formation direction decision, splitting and merging decision, obstacle-free convex region generation, formation shape generation, formation optimization and assignment of robots to target positions. Each step of the proposed method is explained later in this chapter in detail.

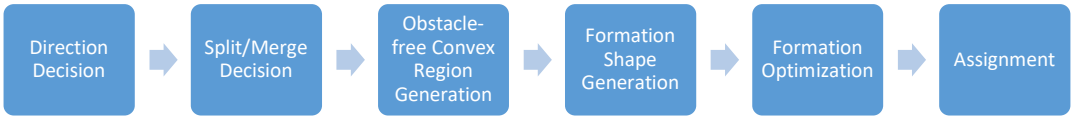


Figure 3.2: Steps of the proposed method while navigating in formation.

3.1.1 Direction Decision

The first step of our proposed local motion planner is direction decision. In Figure 3.3, an overview of direction decision process is given. Using distances of robots to obstacles obtained from lidar sensors, robots calculate a cost for each angle by taking into account the distance to obstacles through that angle and the deviation of the angle from the angle between the current position and the goal position. These cost values are also used by splitting and merging decision which is the next step of our proposed method. They exchange these costs with each other in order to reach consensus on the angle they are going to advance through. Using that direction and the step length, an intermediate waypoint is calculated for the formation.

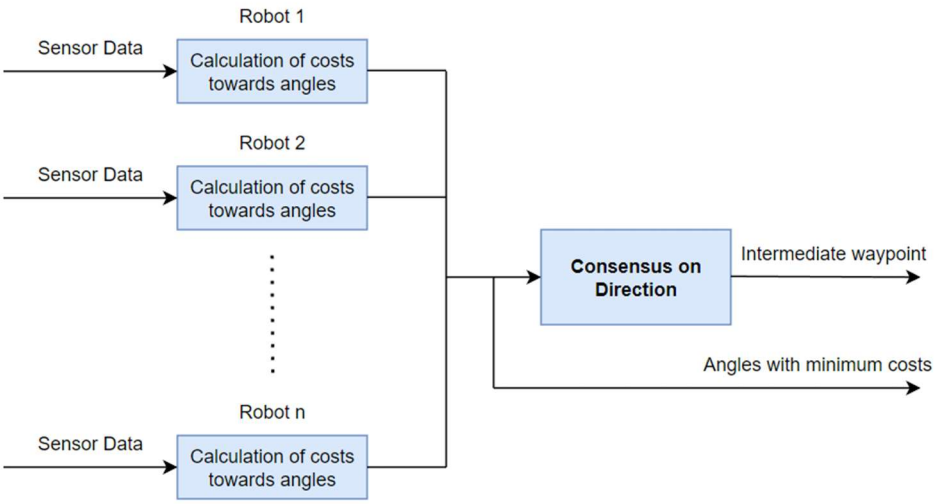


Figure 3.3: Overview of direction decision process.

3.1.2 Splitting and Merging

By using the costs associated with a set of angles calculated at the previous step, each robot first finds the best angle for it by taking the angle with minimum cost, they go through an individual splitting and merging checking which considers the position of corresponding robot with respect to whole team, by checking whether they lie on the right side or the left side of the formation. Each robot reaches a decision of splitting or not and exchanges these decisions with each other. If all robots agree on splitting, splitting occurs and robots are split into groups. The same sequence is true for merging, too. This sequence is given as a diagram in Figure 3.4.

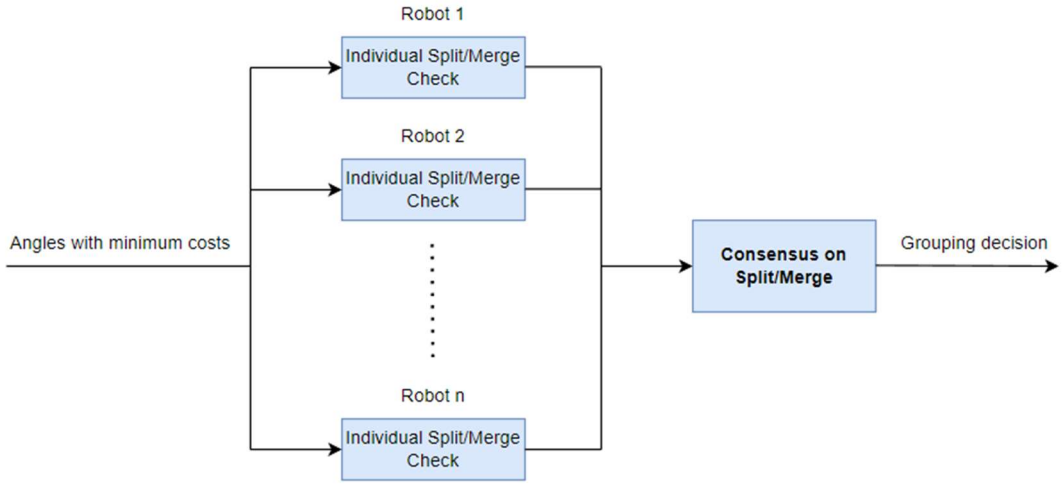


Figure 3.4: Overview of process of deciding upon splitting and merging.

3.1.3 Obstacle-free Convex Region Generation

In order to prevent collisions with obstacles, it is crucial for robots to determine the areas within which they can freely navigate. To accomplish this, each robot employs the Iterative Regional Inflation by Semidefinite Programming (IRIS) algorithm [6] to calculate an obstacle-free convex region. These regions are generated around the intermediate waypoint, which is determined during the direction decision step. Due to variations in their sensing capabilities, each robot possesses different knowledge about the location of obstacles. Exploiting this diversity, robots engage in a process of information exchange to reach a consensus on the safest area. By sharing their individually calculated regions, the robots collaboratively determine the region that provides the highest level of safety, accounting for the collective knowledge of all participating robots. This process is depicted in Figure 3.5.

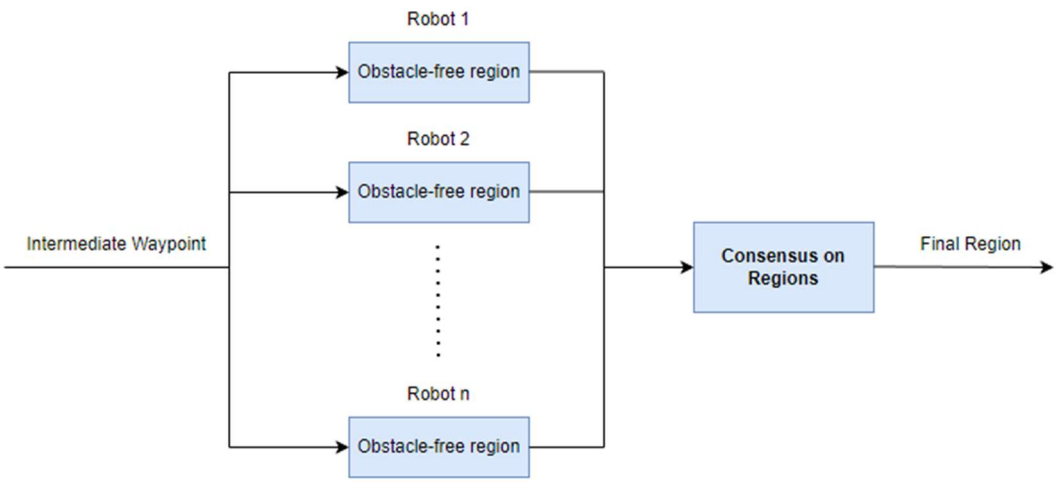


Figure 3.5: Process of reaching consensus on obstacle-free convex region.

3.1.4 Formation Shape Generation

Our work employs lattice and polygonal shapes as geometric formations for robots. To generate these formations, a set of equations is executed by robots, resulting in the generation of a corresponding set of vertices, as given in Figure 3.6. These equations consider various parameters, such as the number of robots, preferred size, and preferred rotational configuration, depending on the mission requirements. By manipulating these parameters, the equations allow for the generation of different shapes that conform to the desired formation requirements.

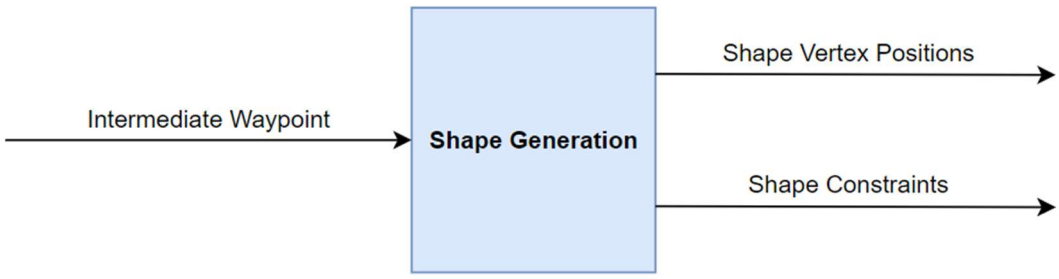


Figure 3.6: Process of shape generation.

3.1.5 Formation Optimization

In order to empower robots to tune their formation configuration based on parameters such as size and rotation, a formation optimization process is introduced. The process of optimization with its inputs and outputs is given in Figure 3.7. This process addresses a nonlinear optimization problem that encompasses both equality and inequality constraints. To tackle this problem, Sequential Quadratic Programming (SQP) [10] is employed as the optimization method. The optimization problem incorporates the intermediate waypoint, calculated during the direction decision step, as the target waypoint. Additionally, it integrates the obstacle-free convex region to ensure that the robots remain within this region in the subsequent timestep. By leveraging this optimization process, the robots become capable of adapting to diverse environmental conditions. For instance, they can adjust their formation size or rotation to maneuver through narrow corridors. They can also navigate around obstacles by translating the waypoint, switch to a different formation shape, or revert to the original formation configuration once the anticipated risk ahead has been cleared. Each UAV solves the optimization on their own and it is assumed that since all UAVs have the same information about obstacle-free region, shape constraints and optimization weights, they reach the same optimized formation configuration. There is no difference between UAVs in the team in terms of responsibilities in this decision step.

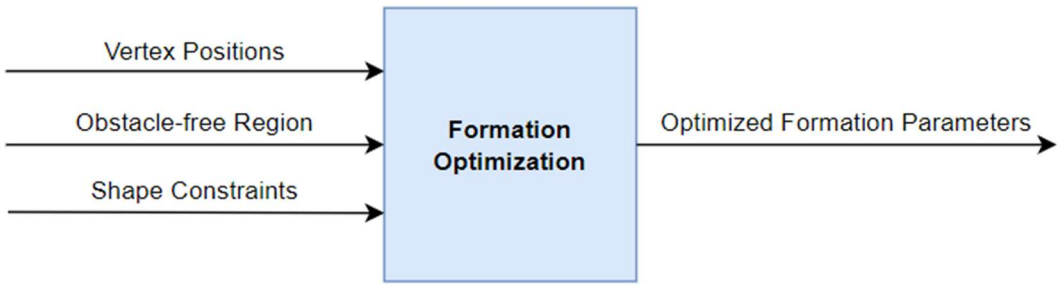


Figure 3.7: Process of reaching optimization of formation configuration.

3.1.6 Assignment

To minimize the mission time, an assignment process is carried out, aiming to reduce the maximum path length of the robots. The primary objective of this assignment is to optimize energy efficiency, particularly for quadrotors. It is important to note that the hover state, where the quadrotors remain stationary in the air, is not energy efficient. Consequently, achieving an optimal mission time often corresponds to optimal energy consumption in these systems. By minimizing the maximum path length, the assignment process contributes to improving the overall efficiency and energy consumption of the robots during the mission. The input/output diagram of the assignment process is given in Figure 3.8. Note that, each UAV carries out the assignment process separately; therefore, it is a duplicate operation for the UAV team. It is assumed that since all UAVs have the same set of UAV positions and shape vertex positions, they will reach the same results.

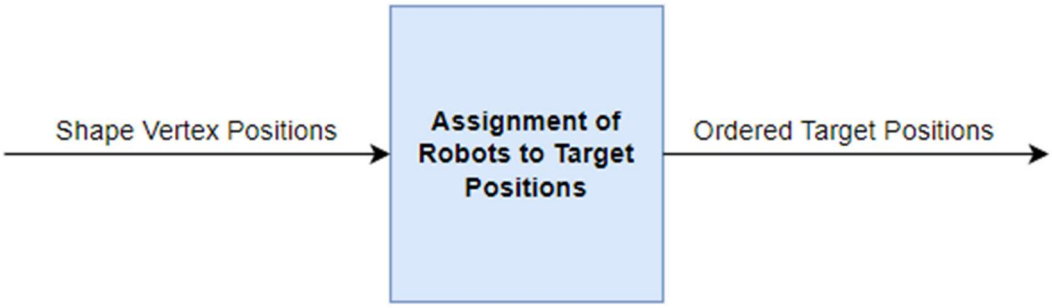


Figure 3.8: Process of assignment.

3.1.7 Assumptions

In this thesis, a connected communication topology is assumed. Let $G = (V, E)$ be the communication graph. Robot i is in the neighborhood of robot j if there exist an edge $(i, j) \in E$. Each robot $i \in V$ has a set of neighboring robots, denoted by $\mathcal{M}_i$. A connected communication graph means that there is a path between every robot pair, with a length of one or more hops. UAVs that are in the same group, which are UAVs

that belong to the same formation are able to be in the neighborhood of each other. For example, if the formation splits into two formations, then there are two different group of UAVs exists in V. If the formation does not split, which means there is one group, all UAVs are in the same group. A spherical activation region of communication is assumed for each agent with a radius R_{com}. Therefore, the neighborhood for robot i can be defined as $\mathcal{M}_i = \{j \in V_i \mid \|X_i - X_j\| \leq R_{com}\}$ where $\|.\|$ is the Euclidean distance operator, $X_i \in R^3$ and $X_j \in R^3$ are positions of robot i and robot j respectively. V_i is the set of UAVs that i^{th} UAV is in the same group with. The diameter of the communication graph d is the longest path that exists between any two robots. We utilized the same coordinate system as Unreal Engine 4, in which simulations are done in this thesis, as shown in Figure 3.9.

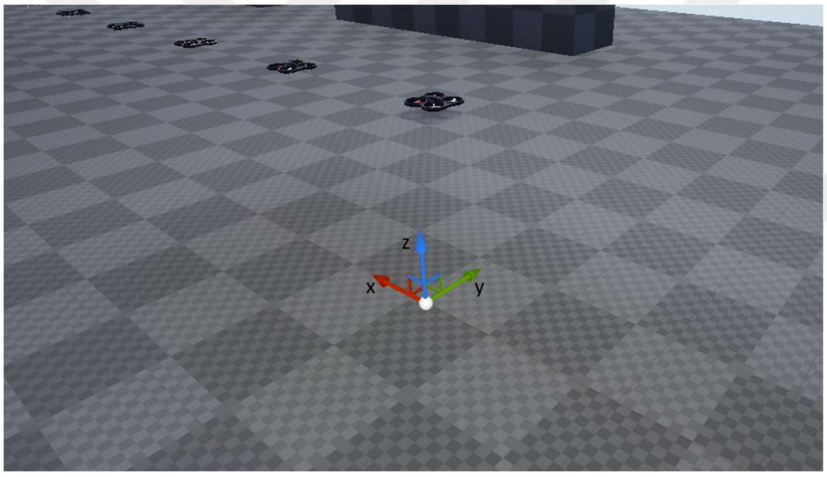


Figure 3.9: Coordinate system in an example scenario.

A spherical detection region, with radius R_{det} which is the detection radius, is assumed for each UAV in this thesis. UAVs are only able to detect obstacles that are inside that region. The spherical detection region of robot i is denoted as $D_i \subset R^3$. Also, obstacles in 3D are assumed to be static and denoted by $O \subset R^3$.

3.2 Formation Shape Generation

In this work, there are two types of shapes used as the geometric configuration of UAVs. These are polygons and lattices, these are shown in Figure 3.10 for a team of 9 agents. Since the method proposed in this thesis is able to work in a distributed manner, a generic formulation of formation shapes is needed such that each robot can generate the desired formation and store a list of target points based on that formation shape. This formulation should be able to generate the shape with a variable number of robots, variable size of the shape and variable rotational parameters. In addition to generating vertices of the shape, a set of equality constraints is also generated in this step. The generated set of equations is then used in formation optimization step, which is described later in this chapter.

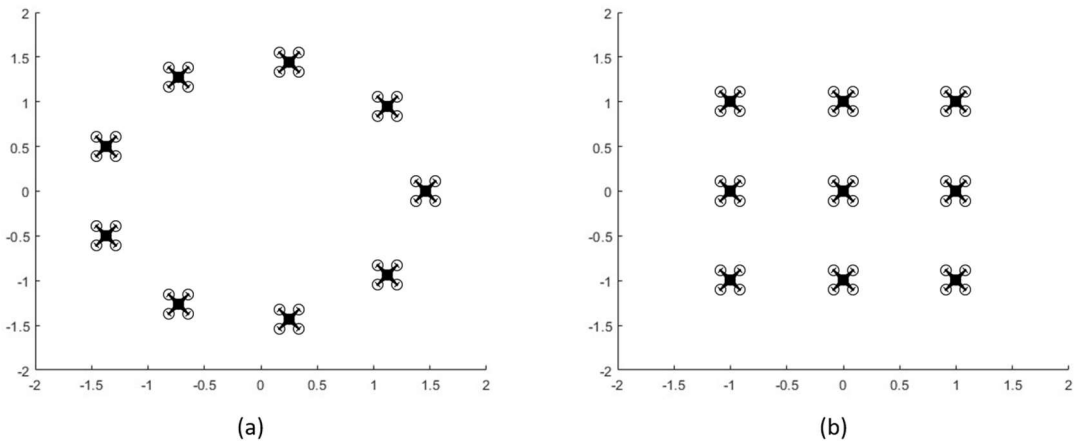


Figure 3.10: An example representation of (a) polygon and (b) lattice shapes with 9 agents, centered at the origin.

Therefore, the formulation used takes the following inputs:

- Formation shape (polygon or lattice)
- Number of robots that are in the formation

- Edge length of the shape
- Orientation of the shape

It outputs the following:

- A set of vertices
- A set of equations that describes the formation shape

In a case where robots start in a position which is not in accordance to the desired formation, robots first generate a set of vertices using this step. The shape is generated such that center of the shape is coincident with the center of initial positions of robots. Then, they go through the assignment step and go to the assigned position. After this initializing process, they start to execute the complete algorithm proposed in this thesis. In Figure 3.11, a case in which robots' initial positions does not form the desired formation is given. Also, their shape after initializing the algorithm is given, which is polygonal.

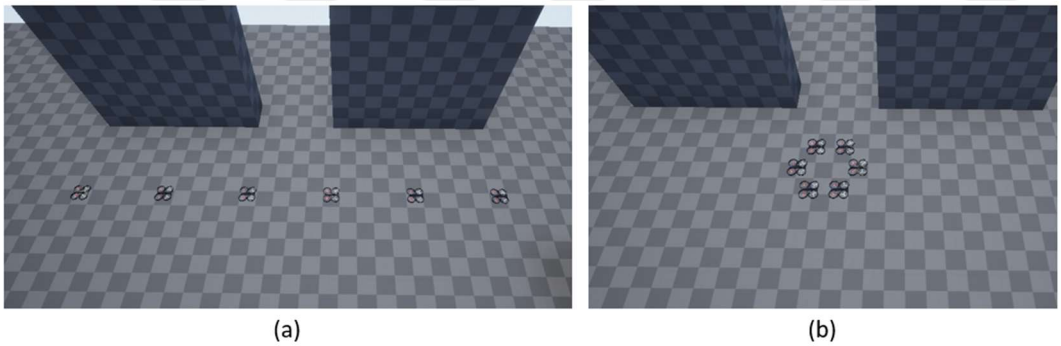


Figure 3.11: An example case with (a) initial positions of robots and (b) robots in formation after initialization of the algorithm.

This shape generation step is not only used for the initialization process, but also at each step of the navigation, since robots decide different size and orientation at each step according to environmental features.

To generate the vertices for polygonal formation, first the radius of the circumscribed circle is found and then sample points are generated from the circle by equally spaced angles.

$$r_c = \frac{s}{(2 * \sin(\frac{\pi}{n}))} \tag{3-1}$$

where s is the desired distance between UAVs, n is the number of UAVs in the formation and r_c is the radius of the circumscribed circle. The vertices are generated on the XY plane first, then they are rotated. In Figure 3.12, a hexagon is given with 6 UAVs ($n = 6$) on its edges and its circumscribed circle.

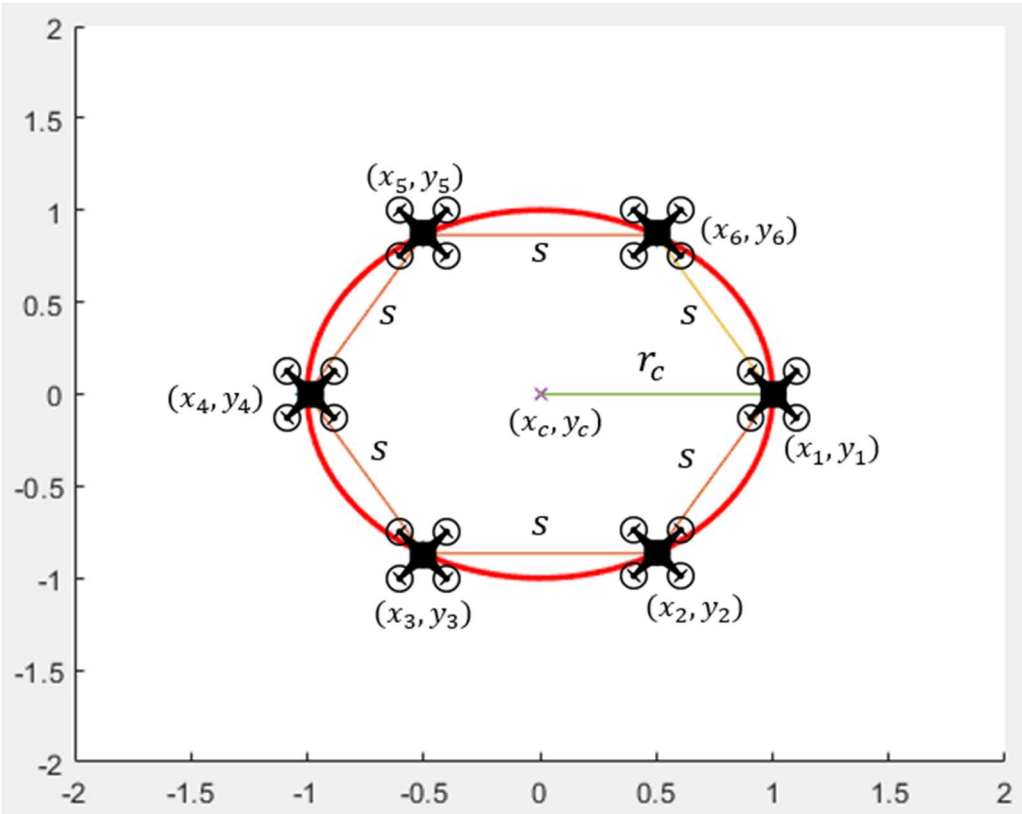


Figure 3.12: Hexagon with circumscribed circle and vertices.

$$x_i = x_c + r_c * \sin\left(\frac{\pi}{n} * (1 + 2i)\right) \tag{3-2}$$

$$y_i = y_c + r_c * \cos\left(\frac{\pi}{n} * (1 + 2i)\right) \tag{3-3}$$

where x_i and y_i are the x and y coordinates of the ith vertex, respectively. x_c and y_c is the x and y coordinates of the center, respectively.

In this work, rotation around z axis is not considered since it does not reduce the projected area of the formation onto the XY cross section, which is the only goal of rotating the formation. In the experiments, unless the formation is rotated in order to

avoid obstacles, the formation shape is parallel to the ground. Only the rotation around the vector between the center of the formation and the goal position, projected onto the XY cross section, is considered. For example, if the center of the formation and the origin is intersecting and the goal position lies on the y axis, only the rotation around y axis is considered. In Figure 3.13, a case with a narrow passage which the team cannot pass without changing its orientation is given. Therefore, they rotate to shrink their area projected onto XY-plane and pass the obstacle. 4 snapshots are given which are shown in XY and XZ cross sections, before and after rotating. In the figure, the green mark depicts the center of the formation and the red mark is the goal position. The axis of rotation is given with the blue arrow.

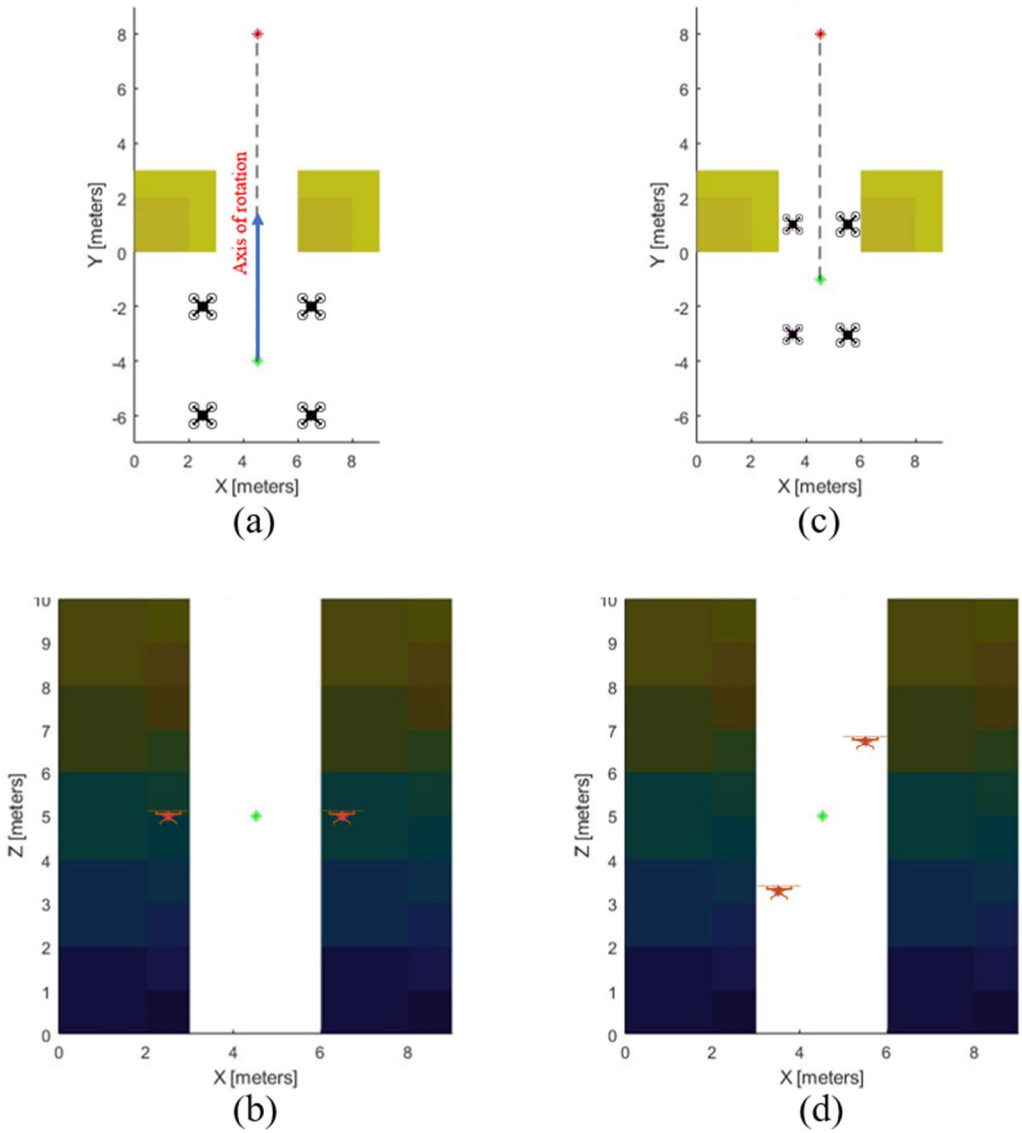


Figure 3.13: A team of four agents (a) before rotating, in XY cross section, (b) before rotating, in XZ cross section, (c) after rotating, in XY cross section and (d) after rotating, in XZ cross section.

To create a lattice-type formation, the polygonal generation algorithm is applied multiple times based on the number of agents in the formation. A sequence of squares is generated, with each square connected to the previous one, resulting in the lattice

shape. After generating each square, the center of the square is shifted by the desired inter-robot distance. This process is repeated for each square to achieve the lattice formation, where the agents are positioned at the centers of the squares while maintaining the desired spacing between them. This approach enables the systematic generation of the lattice formation for multi-agent systems.

3.3 Direction Decision

Robots reach a consensus on an angle which is the direction they will navigate through, at each step that the next formation configuration is computed. Each robot in the formation considers a discrete set of angles, considered to be the same for each robot. This set of angles should be as large as possible in order to consider more angles, with smaller difference between adjacent angles, while navigating; therefore, have better navigation. However, the computational complexity increases with the increasing size of the set. Therefore, we decided on a set of angles of size 360, ranging from $0° - 359°$ with $1°$ difference between adjacent angles. The direction decision depends on the robots' distance to obstacles at each considered angle and the angle between their current formation center and goal position. Note that, this decision is only applicable for the angle on XY-plane, plane parallel to the ground. In Figure 3.14, a team of 5 robots with their center is at the origin can be seen. The goal position, shown as red cross, and the angle between the center of the team and goal position, α, are given in this figure.

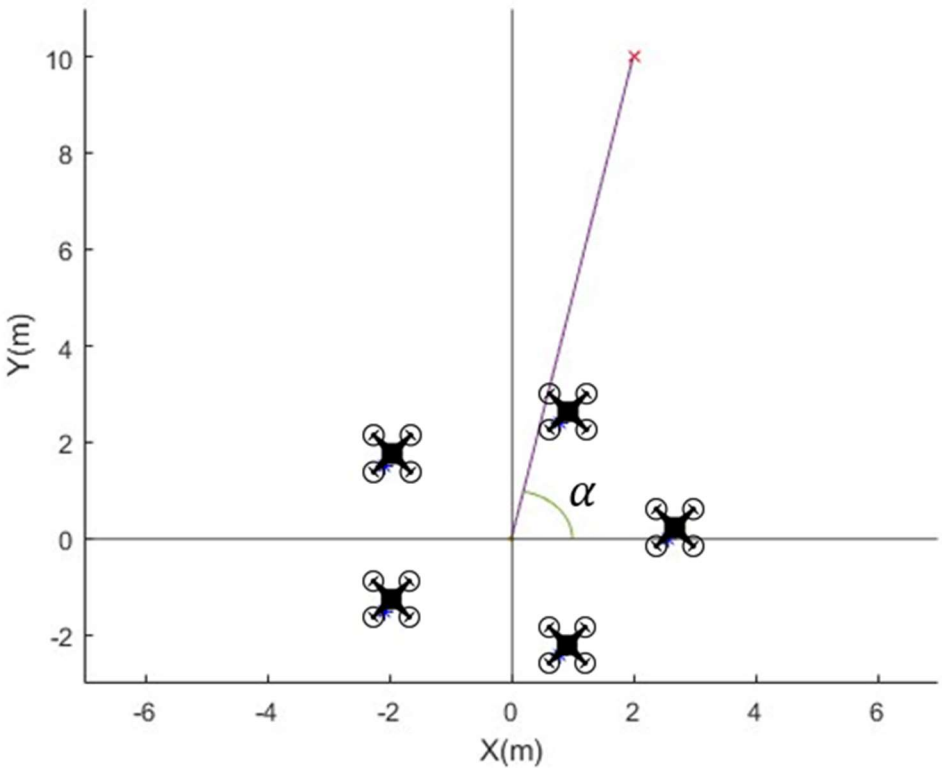


Figure 3.14: The angle to the goal, α, where blue marks are agents and red cross is the goal position.

Consider a 2D case with 3 UAVs given in Figure 3.15 in which areas in black represent obstacles, blue circle represents center of the formation and red mark represents the goal position. Positions of UAVs, the center of the formation and the goal position is given in Table 3.1.

Table 3.1: Positions of UAVs, the formation center and the goal in the example case.

Positions	*X(m)*	*Y(m)*
UAV 1	6	3.73
UAV 2	6	0.27
UAV 3	3	2
Formation Center	5	2
Goal	5	8

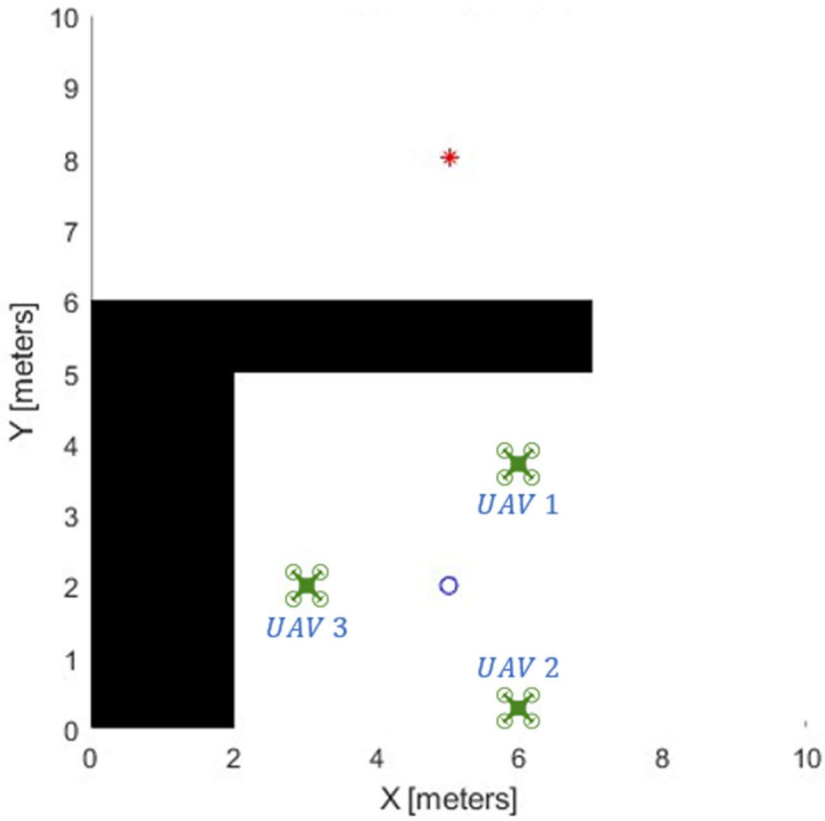


Figure 3.15: An example 2D case with 3 UAVs.

In this example, the angle to the goal, α, is 90° since the line between the center of the formation and the goal position is vertical. For simplicity, we consider the discrete set of angles that we consider in this scenario is between 45° and 135° with 9° difference between adjacent angles. Therefore, the angle set and α in radians are

$$\Theta = \{0.785, 0.942, 1.010, 1.257, 1.414, 1.571, 1.728, 1.885, 2.042, 2.199, 2.356\}$$

$$\alpha = 1.571$$

Robots associate each angle in the set with a cost value, depending on the deviation of the angle from nominal angle and the distance to the obstacle through that angle. We define the cost function as

$$c_i(\theta) = \begin{cases} \infty & if\ d_\theta < d_s \\ |\alpha - \theta| w_a & if\ d_\theta \geq R_{det} \\ |\alpha - \theta| w_a + (R_{det} - d_\theta) w_d & else \end{cases} \tag{3-4}$$

Here, w_d is the coefficient for the distance factor and w_a is the coefficient for the deviation from nominal angle. d_θ is the distance to the obstacles at angle θ. α is the angle between center of the formation and the goal position. d_s is the step size used in the algorithm. c_i is the cost vector of the ith robot. R_{det} is the radius of the detection range of the robot.

In our example, we assume that UAVs have a detection range with a radius of 3 meters and the step size is 1.5 meters. Let us denote distances to obstacles at angles in the set Θ for ith UAV as d_Θ^i, then these distances are, in meters,

$$d_\Theta^1 = \{3, 1.57, 1.43, 1.34, 1.28, 1.27, 1.29, 1.34, 1.43, 1.57, 1.80\}$$

$$d_\Theta^2 = \{3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3\}$$

$$d_\Theta^3 = \{3, 3, 3, 3, 3, 3, 3, 3, 2.20, 1.70, 1.41\}$$

Note that, distances that are equal to 3 meters mean that there are not obstacles detected towards that angle. If coefficients w_a and w_d are both selected to be 1, costs for each angle for each UAV are calculated to be, according to equation (3-4)

$$c_1(\Theta) = \{0.79, 2.06, \infty, \infty, \infty, \infty, \infty, \infty, \infty, 2.06, 1.99\}$$

$$c_2(\Theta) = \{0.79, 0.63, 0.56, 0.31, 0.16, 0, 0.16, 0.31, 0.56, 0.63, 0.79\}$$

$$c_3(\Theta) = \{0.79, 0.63, 0.56, 0.31, 0.16, 0, 0.16, 0.31, 1.36, 1.93, \infty\}$$

Distances to obstacles are obtained from laser scans. In this thesis, robots are assumed to have spherical detection range. Therefore, each robot retrieves a point cloud centered at the current position of the robot, through its sensors. Then, the point cloud is converted to angle-distance table, for a plane parallel to XY-plane. If the distance is equal or nearly equal to the detection range, which means no obstacle is detected through that angle, no cost is added by the distance factor. Also, if the distance to obstacles at the considered angle is smaller than the step size of the algorithm, a very large cost is assumed for that angle.

After each robot calculates costs for all angles, they communicate these costs to other robots, in order to reach a consensus. If there are m discrete angles considered, the cost matrix $C_i \in R^{nxm}$ is the matrix that consist of costs of each angle for each robot, defined as $C_i = [c(1), \dots, c(n)]$ for i^{th} robot, whereas $c(j) \in R^m$ is the cost vector for all angles for j^{th} robot, defined as $c(j) = [c_j(\theta_1), \dots, c_j(\theta_m)]$. After receiving all costs of other agents, sum of costs for each angle is calculated.

$$c_{sum}(\theta_j) = c_1(\theta_j) + \cdots + c_n(\theta_j) \ \forall j \in (1, \dots, m) \tag{3-5}$$

In our example, when costs of three UAVs are summed for each angle in Θ, $c_{sum}(\Theta)$ becomes

$$c_{sum}(\Theta) = \{2.37, 3.32, \infty, \infty, \infty, \infty, \infty, \infty, \infty, 4.32, \infty\}$$

Therefore, the angle with the least cost is found to be the first element of Θ, which is 0.785 radians, or 45°. Since the step size is 1.5 meters and the current formation center is located at [5,2], the intermediate waypoint can be calculated as [5+1.5cos(45°), 2+1.5sin(45°)] which is equal to [6.06, 3.06].

Note that the step length of the algorithm is highly influential in this direction decision. Consider the same case with a different step length of 1 meter, instead of 1.5 meters. Then, costs become

$$c_1(\Theta) = \{0.79, 2.06, 2.13, 1.97, 1.88, 1.73, 1.87, 1.97, 2.04, 2.06, 1.99\}$$

$$c_2(\Theta) = \{0.79, 0.63, 0.56, 0.31, 0.16, 0, 0.16, 0.31, 0.56, 0.63, 0.79\}$$

$$c_3(\Theta) = \{0.79, 0.63, 0.56, 0.31, 0.16, 0, 0.16, 0.31, 1.36, 1.93, 2.38\}$$

$$c_{sum}(\Theta) = \{2.37, 3.32, 3.25, 2.59, 2.2, 1.73, 2.19, 2.59, 3.96, 4.32, 5.16\}$$

In this case, the angle with the least cost is found as 1.571 radians, or 90°. Therefore, in such a case, the formation moves forward and moves to right in the next timestep. The selection of the step length can influence the distance that robots react to the obstacles.

The steps to decide on the direction are given in Algorithm 3.1. In step 1, the cost matrix is initialized. The element corresponding to costs of the robot executing the algorithm is calculated by equation (3-4). At that time, robots only know their own cost values and are unaware of costs of other robots. In steps 4 and 5, robots exchange their own knowledge of cost matrix, and in step 7, these matrices are merged by taking maximum of each element. By executing steps between 3 and 8 for the diameter of communication graph, we guarantee that all costs are known by all robots since the propagation of the data will be complete in steps less than or equal to

diameter d. In step 9, robots calculate the total cost for each angle by summing the costs of each robot for that angle, like shown in equation (3-5). Then, the angle with the minimum cost is selected to be the direction of motion for the team.

Algorithm 3.1. Direction decision for i^{th} robot

1: Initialize $C_i = [c(1), \dots, c(n)]$ as $C_i = 0_{nxm}$
2: Calculate $c(i) = [c_i(\theta_1), \dots, c_i(\theta_m)]$ by using equation (3-4).
3: **for** $k = 1, \dots, d$ **do**
4: Broadcast C_i to all neighboring robots $j \epsilon \mathcal{M}_i$
5: Receive C_j from all neighboring robots $j \epsilon \mathcal{M}_i$
6: **for each** $j \epsilon \mathcal{M}_i$ **do**
7: Compare received costs and take maximum of elements $C_i = \max(C_i, C_j)$
8: **end for**
9: **end for**
10: Take the sum of costs for each angle using equation (3-5).
11: $\theta^* = arg \min_{\theta} c_{sum}(\theta)$

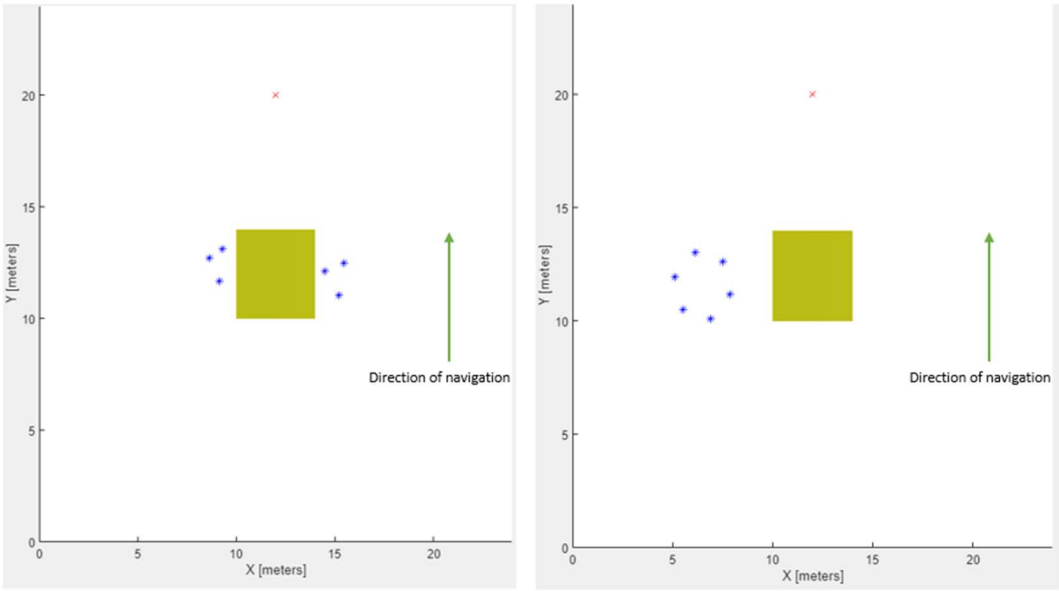
Since all robots have the same cost matrix at the end of communication process, each robot would be able to calculate the same direction of motion.

3.4 Splitting and Merging Conditions

Splitting and merging are quite useful passing around obstacles. For example, when an obstacle aligned with the center of the formation is encountered, it is more optimal in terms of path and mission time to split and go around the obstacle with two groups, instead of moving as a single formation. In such a case, the decision of which robots are going to construct a group is crucial and such a decision should be made intelligently. After robots go around the obstacle, they should be able to merge again,

in order to return to the desired formation shape and maintain the communication structure.

In Figure 3.16, two cases for a team of 6 robots navigating around an obstacle are given. In (a), robots pass the obstacle by splitting into two triangular formations and going from either sides of the obstacle, whereas in (b), the team did not split and they navigate as an hexagonal shape.



(a) (b)

Figure 3.16: An example case where a team of 6 robots navigate around an obstacle (a) with splitting and (b) without splitting.

The splitting conditions take best angles for each agent into account. After executing Algorithm 3.1, each agent know costs for all angles $\theta_j \ \forall \ j \in (1, \dots, m)$ of all robots. Therefore, each robot is able to calculate the angle with the minimum cost for a robot. The best angle for i^{th} robot can be found as

$$\theta_i^* = \underset{\theta}{\operatorname{argmin}}\, c(i) \tag{3-6}$$

Consider a team of 6 UAVs facing an obstacle, as given in Figure 3.17. In the figure, red mark is the goal position and green lines are rough visualizations of best angles of each UAV. Robots' positions, the center of the formation and the goal position are given in Table 3.2. Best angles for UAVs are given in Table 3.3.

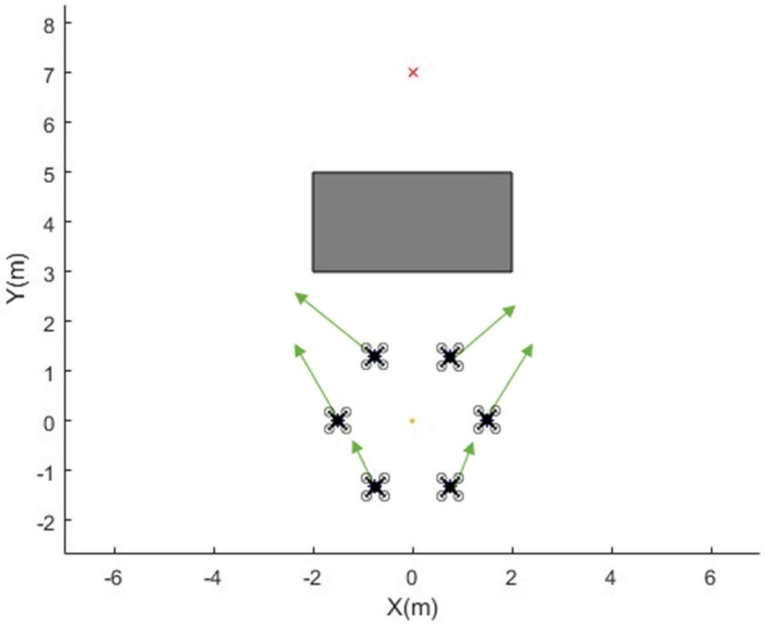


Figure 3.17: A team of 6 UAVs trying to pass an obstacle.

Table 3.2: UAV positions, center and target positions.

Positions	*X(m)*	*Y(m)*
Target	0	7
Center	0	0
UAV 1	1.5	0
UAV 2	0.75	-1.3
UAV 3	0.75	1.3
UAV 4	-0.75	-1.3
UAV 5	-1.5	0
UAV 6	-0.75	1.3

Table 3.3: Best angles of UAVs.

Best angles of	$\theta^*(deg)$
UAV 1	70
UAV 2	75
UAV 3	65
UAV 4	105
UAV 5	110
UAV 6	115

Robots are then grouped into two, considering them as falling to the left or right to the vector between center and the goal position. This is done in XY cross-sectional plane, due to our flight consideration. Let us denote the center of the formation, $\bar{X}_c \in R^2$, goal position, $\bar{X}_G \in R^2$ and position of ith robot in 2D cross-sectional formation as $\bar{X}_i \in R^2$. Then, using vectors between center and robots' positions, $\overrightarrow{\bar{X}_c\bar{X}_i}$ and

between center and goal positions, $\overrightarrow{\bar{X}_c\bar{X}_G}$, it is possible to check whether the robot lies on the left of $\overrightarrow{\bar{X}_c\bar{X}_G}$ or right as

$$\sigma_i = \overrightarrow{\bar{X}_c\bar{X}_G} \times \overrightarrow{\bar{X}_c\bar{X}_i} \tag{3-7}$$

If σ_i is greater than zero, it means that i^{th} robot lies on the left side of the vector between center of the formation and goal position. If σ_i is smaller than zero, i^{th} robot lies on the right side of the same vector. In case σ_i is exactly zero, it implies that robot lies on the vector between center of the formation and goal position.

In our example, these vectors between the center of the formation and robots' positions and the vector between the center and the goal of the formation are calculated as, from Table 3.2:

$$\overrightarrow{\bar{X}_c\bar{X}_1} = \langle 1.5, 0 \rangle$$

$$\overrightarrow{\bar{X}_c\bar{X}_2} = \langle 0.75, -1.3 \rangle$$

$$\overrightarrow{\bar{X}_c\bar{X}_3} = \langle 0.75, 1.3 \rangle$$

$$\overrightarrow{\bar{X}_c\bar{X}_4} = \langle -0.75, -1.3 \rangle$$

$$\overrightarrow{\bar{X}_c\bar{X}_5} = \langle -1.5, 0 \rangle$$

$$\overrightarrow{\bar{X}_c\bar{X}_6} = \langle -0.75, 1.3 \rangle$$

$$\overrightarrow{\bar{X}_c\bar{X}_G} = \langle 0, 7 \rangle$$

By calculating cross products of $\overrightarrow{\bar{X}_c\bar{X}_G}$ and $\overrightarrow{\bar{X}_c\bar{X}_i}$, we can check whether robots are falling to the left or right side of the formation center:

$$\sigma_1 = -10.5$$

$$\sigma_2 = -5.25$$

$$\sigma_3 = -5.25$$

$$\sigma_4 = 5.25$$

$$\sigma_5 = 10.5$$

$$\sigma_6 = 5.25$$

Which means, in our example, the first 3 UAVs belong to the right half of the formation while the other 3 UAVs belong to the left half.

The splitting occurs when the left group decides to go to the left with respect to goal vector, and the right group decides to go to the right. This decision is carried out by comparing the nominal angle, α, which is depicted in Figure 3.14, and the best angle for the corresponding robot.

$$split_i = \begin{cases} true & if\ (\sigma_i > 0\ and\ \theta_i^* > \alpha)\ \boldsymbol{or}\ (\sigma_i < 0\ and\ \theta_i^* < \alpha)\ \boldsymbol{or}\ (\sigma_i = 0\ ) \\ false & otherwise \end{cases} \tag{3-8}$$

If the condition in equation (3-8) holds true for all robots, then splitting occurs and two groups are formed based on being on the left side or the right side. In our example, angle between center of the formation and the goal position is $\alpha = 90°$. For UAVs with $\sigma_i > 0$, it can be seen that $\theta_i^* > \alpha$ as in Table 3.2. Reverse is also true; For UAVs with $\sigma_i < 0$, angular condition also holds true, $\theta_i^* < \alpha$. Therefore, split conditions for all UAVs are true and formation splits into two teams in the next step.

Splitting can also occur when there is an odd number of UAVs. In such a case, two groups will be formed with different numbers of UAVs in each one. Consider a case similar to the previous example but with 5 UAVs, centered again around the origin. Positions of UAVs are given in Table 3.4.

Table 3.4: UAV positions, center and target positions for the case with odd number of UAVs.

Positions	*X(m)*	*Y(m)*
Target	0	7
Center	0	0
UAV 1	1.28	0
UAV 2	0.39	-1.21
UAV 3	-1.03	-0.75
UAV 4	-1.03	0.75
UAV 5	0.39	1.21

Table 3.5: Best angles of UAVs.

Best angles of	$\theta^*(deg)$
UAV 1	75
UAV 2	70
UAV 3	110
UAV 4	105
UAV 5	65

Vectors between the center of the formation and robots' positions and the vector between the center and the goal of the formation are

$$\overrightarrow{\bar{X}_c\bar{X}_1} = \langle 1.28, 0 \rangle$$

$$\overrightarrow{\bar{X}_c\bar{X}_2} = \langle 0.39, -1.21 \rangle$$

$$\overrightarrow{\bar{X}_c\bar{X}_3} = \langle -1.03, -0.75 \rangle$$

$$\overrightarrow{\bar{X}_c\bar{X}_4} = \langle -1.03, 0.75 \rangle$$

$$\overrightarrow{\bar{X}_c\bar{X}_5} = \langle 0.39, 1.21 \rangle$$

$$\overrightarrow{\bar{X}_c\bar{X}_G} = \langle 0, 7 \rangle$$

By calculating cross products of $\overrightarrow{\bar{X}_c\bar{X}_G}$ and $\overrightarrow{\bar{X}_c\bar{X}_\iota}$

$$\sigma_1 = -8.96$$

$$\sigma_2 = -2.73$$

$$\sigma_3 = 7.21$$

$$\sigma_4 = 7.21$$

$$\sigma_5 = -2.73$$

Again, in this case, equation (3-8) holds true for all UAVs. However, in such a case, two unbalanced groups are formed, one containing UAVs 1, 2, 5 and the other containing UAVs 2, 3. When regular polygons are considered, as in this work, groups formed by splitting can have a maximum difference of 1 in terms of number of UAVs. This is because we separate groups by considering UAVs either falling to the left side or the right side of the formation.

Steps of checking whether formation splits or not are given in Algorithm 3.2.

Algorithm 3.2. Checking for splitting.

```
if not already split
    for k = 1, ..., n do
        Calculate best angle θ_k by using equation (3-6).
        Calculate σ_k by using equation (3-7).
        Check for split_k by using equation (3-8).
    end for
    split = split_1 ∩ ... ∩ split_n
```

After splitting, once teams passed an obstacle or a passage, they need to merge again into one team in order to return to desired formation shape. There are two conditions need to be satisfied in order to merge. First, teams should not be seperated too much. Second, teams should want to move closer to each other. To achieve the first condition, distance between centers of groups needs to be lower than the preferred distance between UAVs. For the second condition, again a check to determine whether a group is on the left side or right side of the vector between overall center of the system and goal position. Then, if the group on the left side wants to go to the right and the group on the right wants to go the left; it means that two groups want to move closer to each other. When there are some non-symmetric obstacles present, one group may pass the obstacle faster than the other, which may cause groups to be distant from each other even after passing the obstacle. In such a case, since there is no mechanism that makes the group in the front to wait for the other group, the merging may occur later in the mission, or after reaching the goal ultimately. In Figure 3.18, four different snapshots from a case where a team 6 robots are navigating in an environment with obstacles. The sequence of splitting, passing the obstacle while being split and merging after obstacle is cleared can be seen.

To check whether merging should occur or not, instead of using best angles of each robot, the decided direction of the group is used. The distance between centers of groups is

$$d_G = \|X_{c1} - X_{c2}\| \tag{3-9}$$

where, X_{ck} is the center position of the kth group.

Then, the merging condition becomes

$$merge_{G1} = \begin{cases} true & if\ \left((\sigma_{G1} > 0\ and\ \theta_{G1}^{*} < \alpha)\ or\ (\sigma_{G1} < 0\ and\ \theta_{G1}^{*} > \alpha)\right)\ and\ d_G < s_{pref} \\ false & otherwise \end{cases} \tag{3-10}$$

Algorithm 3.3. Checking for merging.

```
if already split
    Calculate d_G by using equation (3-9)
    for k = 1,2
        Calculate desired angle θ_k by using Algorithm 3.1. for group k
        Calculate σ_Gk by using equation (3-7).
        Check for merge_Gk by using equation (3-10).
    end for
    merge = merge_G1 ∩ merge_G2
```

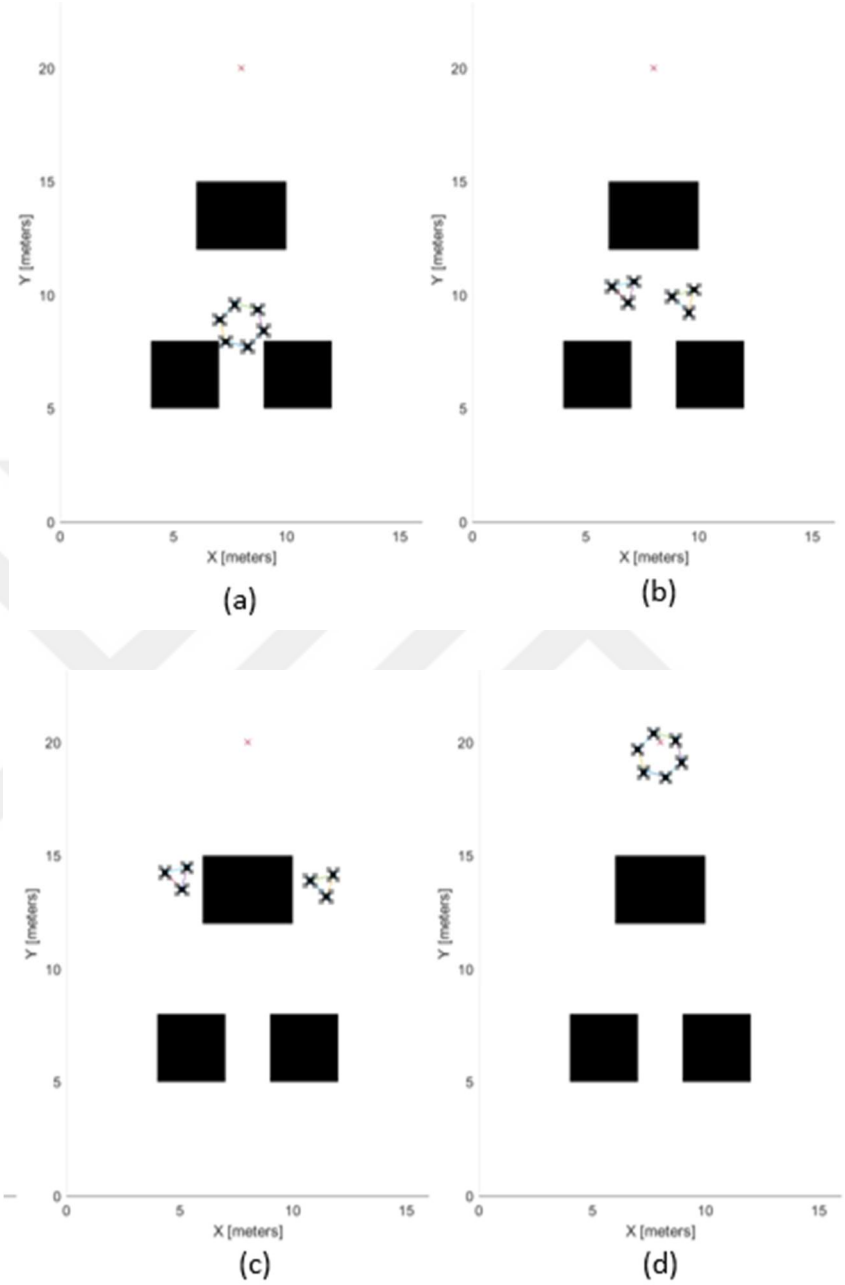


Figure 3.18: A team of robots (a) before splitting, (b) after splitting decision, (c) going around an obstacle while split and (d) merging again after passing the obstacle.

3.5 Obstacle-free Convex Region Generation

After deciding on the direction to move, agents are able to calculate an intermediate waypoint through that direction. An intermediate waypoint is the position where the center of the formation is desired to be at the end of the next step. Therefore, the distance between the current center of the formation and the intermediate waypoint is equal to the step size.

In the optimization step, the algorithm tries to fit the formation in the obstacle-free convex region. Consider the example given in Figure 3.19, in which two obstacle-free convex regions generated around different locations in the same environment are given. As in the figure, when there are sharp edges ahead, generated areas may not include the area past the obstacle. Since we need to make sure the generated area includes the intermediate waypoint as we try to fit the formation inside that area while being centered on that waypoint, convex regions should be generated around on the intermediate waypoint, instead of current positions of the robots. In the case of waypoint not being included in the generated region, the formation would get stuck because of the limiting convex region.

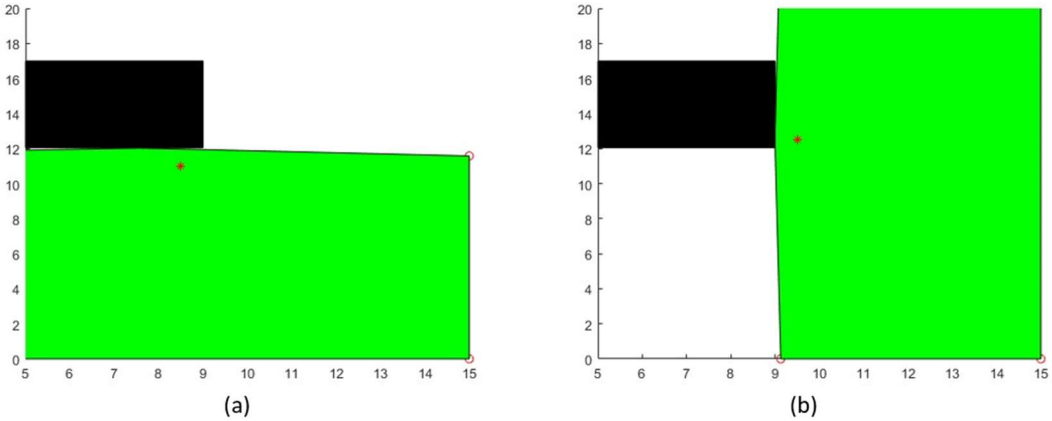


Figure 3.19: Difference of convex regions with respect to starting points, green area is the generated region, black area is the obstacle and red dots are starting points.

Consider a 2D example with 3 UAVs navigating in a cluttered environment. Positions of UAVs and the intermediate waypoint is given below in Table 3.6. There are 3 rectangular obstacles in the environment, whose center locations, lengths and widths are given in Table 3.7. The radius of detection ranges for all UAVs is 1.9m. The example environment can be seen in Figure 3.20 with obstacles are drawn in gray color and the intermediate waypoint is shown as the red mark. Limits of the environment are 0-10m in X axis and 0-7m in Y axis.

Table 3.6: Positions of UAV and intermediate waypoint.

Positions	*X(m)*	*Y(m)*
UAV 1	4	2
UAV 2	6	2
UAV 3	5	3.732
Waypoint	5	4

Table 3.7: Obstacle centers and sizes.

Type of Travel	*X(m)*	*Y(m)*	*Width(m)*	*Length(m)*
Obstacle 1	2.5	3.5	1	5
Obstacle 2	5.5	5.5	1	1
Obstacle 3	7.5	3.5	1	5

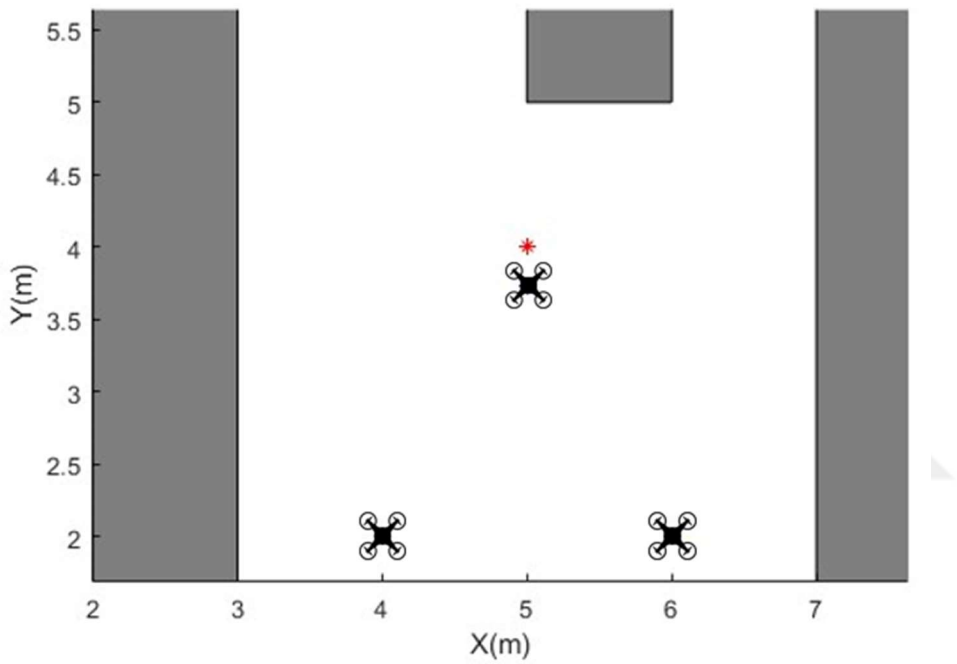


Figure 3.20: Example environment.

To generate convex regions, each agent utilizes IRIS method described in [6]. IRIS algorithm is capable of quickly computing large obstacle-free polytopic and ellipsoidal regions, given a set of obstacles and a starting point. The method is an iterative approach. To generate polytopic regions, a set of hyperplanes that separate obstacles from the convex region are computed by a quadratic program and a

semidefinite program is utilized to find the maximum-volume ellipsoid inside the computed polytopic region.

The result of the IRIS algorithm is a set of hyperplanes, defined as a set of equations. Let $A \in R^{px}$ a matrix where p is the number of hyperplanes generated and $b \in R^{p}$ is a vector. For an arbitrary point x, the condition to be inside the convex region is

$$Ax < b \tag{3-11}$$

In Figure 3.21, obstacle-free convex regions generated by different UAVs are given. Generated regions are depicted in green; detection ranges of UAVs are shown as yellow circles. The red mark is the intermediate waypoint around which regions are generated.

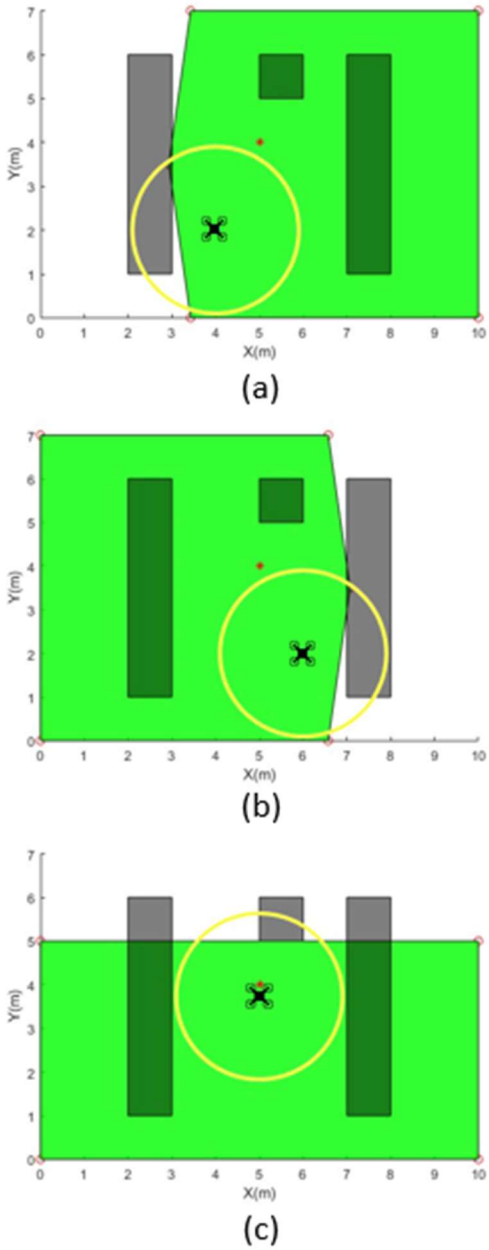


Figure 3.21: Convex regions generated by (a) UAV 1, (b) UAV 2 and (c) UAV 3.

Resulting representations of areas generated by IRIS algorithm are:

$$A_1 = \begin{bmatrix} -0.9899 & -0.1414 \\ -0.9899 & 0.1414 \\ -1 & 0 \\ 0 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$b_1 = \begin{bmatrix} -3.3941 \\ -2.4042 \\ 0 \\ 0 \\ 10 \\ 7 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \\ 0 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$b_2 = \begin{bmatrix} 5 \\ 0 \\ 0 \\ 10 \\ 7 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 0.9899 & -0.1414 \\ 0.9899 & 0.1414 \\ -1 & 0 \\ 0 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$b_3 = \begin{bmatrix} 6.5054 \\ 7.4953 \\ 0 \\ 0 \\ 10 \\ 7 \end{bmatrix}$$

In our approach, each robot generates a convex region based on its knowledge of obstacles. Since robots have limited range of detection, each robot may sense different obstacles. In order to act upon all possible knowledge on obstacles, these regions are communicated between robots and then an intersection of these regions is used to fit the formation.

The region generated by i^{th} agent is defined by A_i and b_i. Then, several regions can be merged into A and b as

$$A = \begin{bmatrix} A_1 \\ A_2 \\ \dots \\ A_n \end{bmatrix}$$

$$b = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix} \tag{3-12}$$

In our example, the intersection of areas generated by UAVs, which is obtained by combining non-duplicate equations are given below. This way, if the chosen step size is smaller than the detection range, robots are guaranteed to generate a safe region for navigation and if the formation is fitted to this region in optimization step, the target configuration would not be intersecting with any obstacles, as in Figure 3.22 which is the visualization of the resulting area of our example.

$$A = \begin{bmatrix} -0.9899 & -0.1414 \\ -0.9899 & 0.1414 \\ 0.9899 & -0.1414 \\ 0.9899 & 0.1414 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$b = \begin{bmatrix} -3.3941 \\ -2.4042 \\ 6.5054 \\ 7.4953 \\ 5 \\ 0 \\ 0 \\ 10 \\ 7 \end{bmatrix}$$

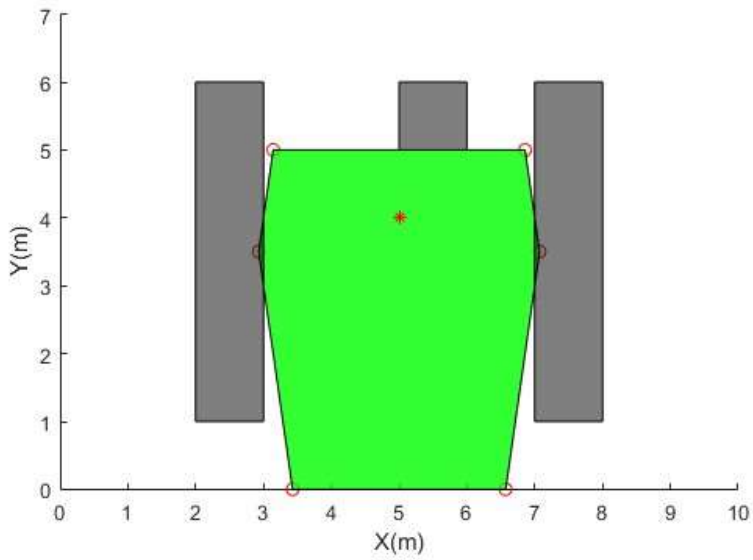


Figure 3.22: The intersection of areas generated by 3 UAVs.

The procedure executed by each agent is described in Algorithm 3.4 below. It starts with the direction found by Algorithm 3.1 to generate an intermediate waypoint to the formation. Then, using that waypoint as the starting point of convex region generation, IRIS [6] algorithm is used to generate convex region as a set of hyperplanes. These hyperplanes are stored in A_i and b_i. Then, steps 4-6 are executed for the diameter of the communication graph in order to guarantee that data from all agents are received by others. In these steps, known hyperplanes are shared with

neighboring robots and also received from them. Then, non-duplicate hyperplanes are appended to A_i and b_i. The resulting matrices are to be used in the optimization step. Note that, although an intersection of independent regions is used, since these regions are generated around the same location which is the intermediate waypoint, the resulting area is no more restrictive than a single region generated with the knowledge of all obstacles combined. Therefore, the resulting region is as restrictive as the IRIS algorithm itself.

Algorithm 3.4. Obstacle-free convex region generation algorithm for i^{th} robot

Using the direction, result of Algorithm 3.1, compute intermediate waypoint
Compute A_i and b_i using IRIS algorithm [6]
for $k = 1, \dots, d$ **do**
    Broadcast A_i and b_i to all neighboring robots $j \in \mathcal{M}_i$
    Receive A_j and b_j from all neighboring robots $j \in \mathcal{M}_i$
    Add non-duplicate entries to $A_i \& b_i$, $A_i = A_i \cup A_j$ $b_i = b_i \cup b_j$ as (3-12).
end for
$A = A_i$ and $b = b_i$

3.6 Formation Optimization

After deciding on a direction to move and calculating an obstacle-free convex region, each UAV solves an optimization problem to determine the optimal formation parameters which are the size, orientation and center of the formation. The solution of the optimization problem gives a set of target points which help to generate the formation configuration in accordance with the environmental features, and the size, rotation and formation center corresponding to that formation configuration. Therefore, the state vector to be optimized consists of positions of UAVs, $X_i \in R^3$,

which sum to $3n$ variables where n is the number of UAVs in the group. The state vector also has the size, the center and the rotation of the formation. Therefore, there are $3n + 5$ variables to be optimized. Note that, size, center and rotation variables can be derived from the positions of the UAVs by considering the formation shape.

Consider an example given in Figure 3.23, where a team of 4 UAVs are planning their configuration for the next step. The obstacle-free region is the red colored area and blue crosses are predicted target positions for UAVs, with the intermediate waypoint given by blue mark which is the center of the predicted formation. The predicted positions are generated around the intermediate waypoint with inter-robot distances being equal to the preferred size. Since there are 4 UAVs, there are 17 variables to be optimized such as

$$X = \{x_1, x_2, \dots, x_{17}\} = \{p_x^1, p_y^1, p_z^1, p_x^2, p_y^2, p_z^2, p_x^3, p_y^3, p_z^3, p_x^4, p_y^4, p_z^4, p_x^c, p_y^c, p_z^c, s, \theta\}$$

where p_x^i, p_y^i, p_z^i are x, y and z positions of ith vertex of the shape, respectively. p_x^c, p_y^c and p_z^c are x, y and z positions of the center of the formation, respectively. s is the size of the formation and θ is the rotation amount of the formation.

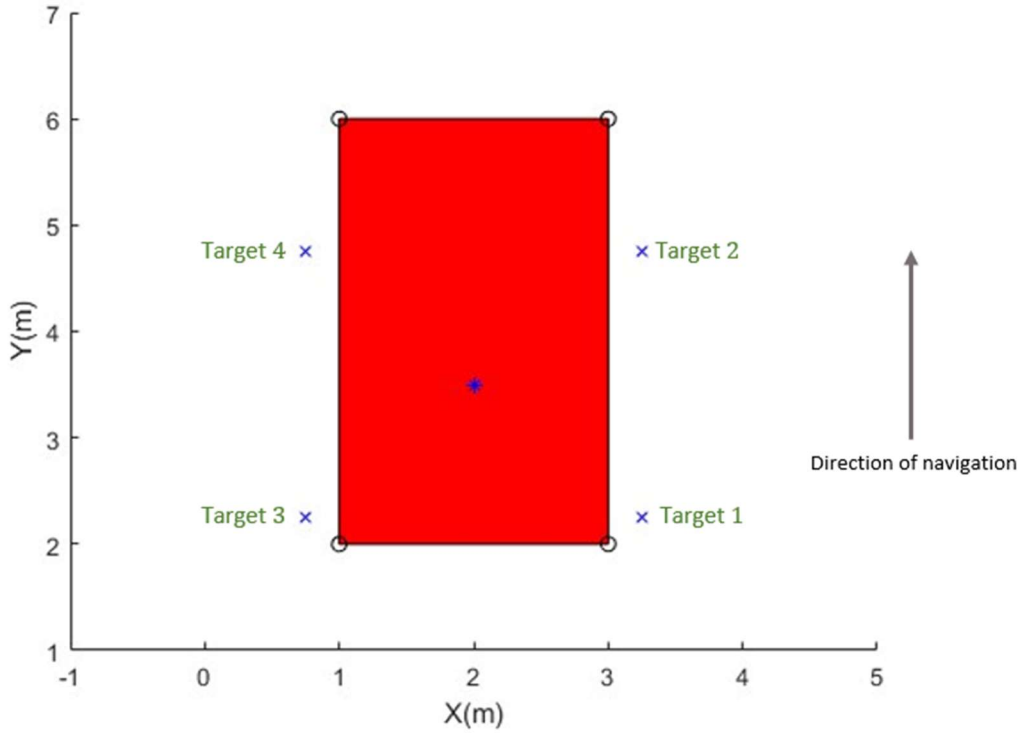


Figure 3.23: An example optimization case for a team of 4 UAVs.

Initial guesses of parameters that are to be optimized are their values when the formation is in its nominal configuration, which is when the formation centered around the intermediate with the preferred size and rotation. In this scenario, the preferred size, s_{pref}, is 2.5 meters and the preferred rotation, θ_{pref}, is zero, such that the formation is preferred to be parallel to the XY cross section. Then, initial guesses of parameters are

$$X_0 = \{3.25, 2.25, 2, 3.25, 4.75, 2, 0.75, 2.25, 2, 0.75, 4.75, 2, 2, 3.5, 2, 2.5, 0\}$$

The optimization problem can be formalized as follows:

$$\begin{aligned} &\min_{X} J(X) \\ &s.t. \quad A_{ofcr}X < b_{ofcr} \\ &\qquad f_{shap}\ (\mathrm{X}) = 0 \\ &\qquad X_{min} < X < X_{max} \end{aligned} \tag{3-13}$$

where the first inequality constraint comes from the obstacle-free convex region generated before, to ensure that the positions at the next step lies inside the convex region. The equality constraint comes from the shape characteristics, which lays the relation between edge positions of the shape and the rotation, center and the size parameters of the formation. In our example, there are 12 equality constraints defining the formation as

$$x_1 - x_{13} - \frac{x_{16}\cos(x_{17})}{2} = 0$$

$$x_2 - x_{14} + \frac{x_{16}}{2} = 0$$

$$x_3 - x_{15} + \frac{x_{16}\sin(x_{17})}{2} = 0$$

$$x_4 - x_{13} - \frac{x_{16}\cos(x_{17})}{2} = 0$$

$$x_5 - x_{14} - \frac{x_{16}}{2} = 0$$

$$x_6 - x_{15} + \frac{x_{16}\sin(x_{17})}{2} = 0$$

$$x_7 - x_{13} + \frac{x_{16}\cos(x_{17})}{2} = 0$$

$$x_8 - x_{14} + \frac{x_{16}}{2} = 0$$

$$x_9 - x_{15} - \frac{x_{16}\sin(x_{17})}{2} = 0$$

$$x_{10} - x_{13} + \frac{x_{16}\cos(x_{17})}{2} = 0$$

$$x_{11} - x_{14} - \frac{x_{16}}{2} = 0$$

$$x_{12} - x_{15} - \frac{x_{16}\sin(x_{17})}{2} = 0$$

In our example, A and b matrices defining the obstacle-free convex region are

$$A = \begin{bmatrix} -1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -1 \end{bmatrix}$$

$$b = \begin{bmatrix} -1 \\ 3 \\ 6 \\ -2 \\ 4 \\ 0 \end{bmatrix}$$

The obstacle-free convex region constraints apply to positions of all robots in order to make sure that they all lay inside obstacle-free region; therefore, the constraint matrices A_{ofcr}, which has 24 rows and 17 columns, and b_{ofcr}, which has 24 rows and 1 column, can be defined as

$$A_{ofcr} = \begin{bmatrix} A & 0 & 0 & 0 & 0 \\ 0 & A & 0 & 0 & 0 \\ 0 & 0 & A & 0 & 0 \\ 0 & 0 & 0 & A & 0 \end{bmatrix}$$

$$b_{ofcr} = \begin{bmatrix} b \\ b \\ b \\ b \end{bmatrix}$$

Note that, the inequality constraint is convex and linear, whereas the equality constraint is nonlinear. Variables to be optimized also have upper and lower bounds for the size and positions, which are determined by the user. Bounds on the positions depend on bounds of the environment. Assume that, in our example, only the formation size parameter has a lower bound, such that,

$$1.5 < x_{16} < 2.5$$

The constrained nonlinear optimization problem given in equation 3-13 can be solved by nonlinear solvers. The nonlinear solver of the MATLAB, fmincon [45], is used and Sequential Quadratic Programming(SQP) algorithm is selected.

There are several components in the objective function, which is the translational cost J_t, the size cost J_s, the rotational cost J_r and a switching cost J_{sw}. Each cost element is described seperately in this chapter. The total objective function is defined as

$$J(X) = w_t J_t(X) + w_s J_s(X) + w_r J_r(X) + J_{sw} \tag{3-14}$$

In our example, switching cost is not considered since there is no other formation shape defined for 4 UAVs. Therefore, the cost function in the example scenario is

$$J(X) = w_t J_t(X) + w_s J_s(X) + w_r J_r(X)$$

3.6.1 Translational cost

Translational cost defines the translational deviation of the formation from the intermediate waypoint. For the translational cost, the norm of the vector between computed center of the formation at the next step and the preferred intermediate waypoint is used with a weight w_t.

$$J_t(X) = \|X_c - X_G\|^2$$

(3-15)

where X_C is the center of the formation at the next step, X_G is the intermediate waypoint calculated by utilizing the preferred direction of motion.

In our example, the translational cost becomes

$$J_t(X) = ((x_{13} - 2)^2 + (x_{14} - 3.5)^2 + (x_{15} - 2)^2)$$

3.6.2 Size cost

Size cost is associated with the size of the formation such that the square of the difference between preferred size and the computed size at a given time step is used with a weight w_s. Note that, the preferred size is determined before the execution of the mission by the user.

$$J_s(X) = \left\|s - s_{pref}\right\|^2$$

(3-16)

where s is the size of the formation at the next step and s_{pref} is the preferred size of the formation.

In our example, since the preferred size is 2.5 meters, the size cost is

$$J_s(X) = (x_{16} - 2.5)^2$$

3.6.3 Rotational Cost

Rotational cost is the cost which penalizes the deviation of the computed rotation amount of the formation at a given step from the preferred orientation. Note that, in

all of our experiments, the preferred orientation is planar with the flight cross section, which means the preferred rotation amount is zero. Rotational cost has a weight w_r.

$$J_r(X) = \left\|\theta - \theta_{pref}\right\|^2 \tag{3-17}$$

where θ is the rotation amount of the formation at the next step, θ_{pref} is the preferred orientation. In our example, since the preferred rotation is zero, the rotation cost becomes

$$J_r(X) = x_{17}^2$$

Therefore, the cost function in the example is

$$J(X) = w_t((x_{13} - 2)^2 + (x_{14} - 3.5)^2 + (x_{15} - 2)^2) + w_s(x_{16} - 2.5)^2 + w_r x_{17}^2$$

If the rotational cost weight is selected to be relatively lower than the size cost, then the formation becomes more inclined towards rotating in order to pass obstacles, rather than changing size. In our example, if weights are selected as $w_t = 1, w_s = 1$ and $w_r = 0.2$, then the optimized formation configuration is as given in Figure 3.24. Note that, the selection of weights is dependent on the behavior that the user want the formation to be inclined towards. In Figure 3.24a, a view from XY cross section is given and in Figure 3.24b, another view from YZ cross section is given. Since the rotational cost weight w_r is relatively lower than other weights, the formation is rotated in order to fit in the obstacle-free region. In Figure 3.25, the value of the size parameter at each iteration of the optimization is given. The optimization process starts at the preferred value of the size which is 2.5 meters. During search of optimal parameter set, it decreases in some iterations while converging back to the preferred size at the end of the process since the rotational cost weight is relatively lower than the size cost weight. The rotation parameter value at each iteration of the optimization process is depicted in Figure 3.26. The search for the optimal rotation value starts at the preferred rotation amount which is 0 radians and converges to

0.644 radians which is the minimum amount of rotation in order to fit in the obstacle-free region.

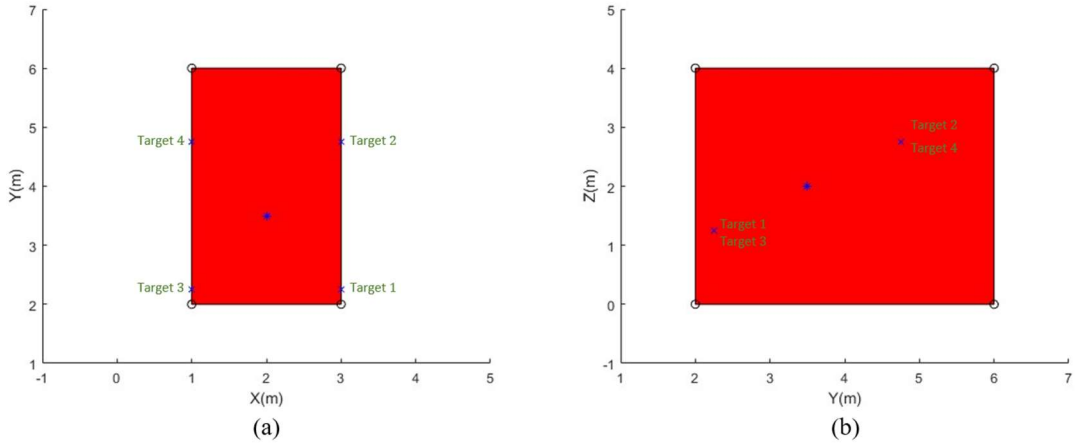


Figure 3.24: The optimized formation configuration for an example weight set from (a) XY cross section and (b) YZ cross section.

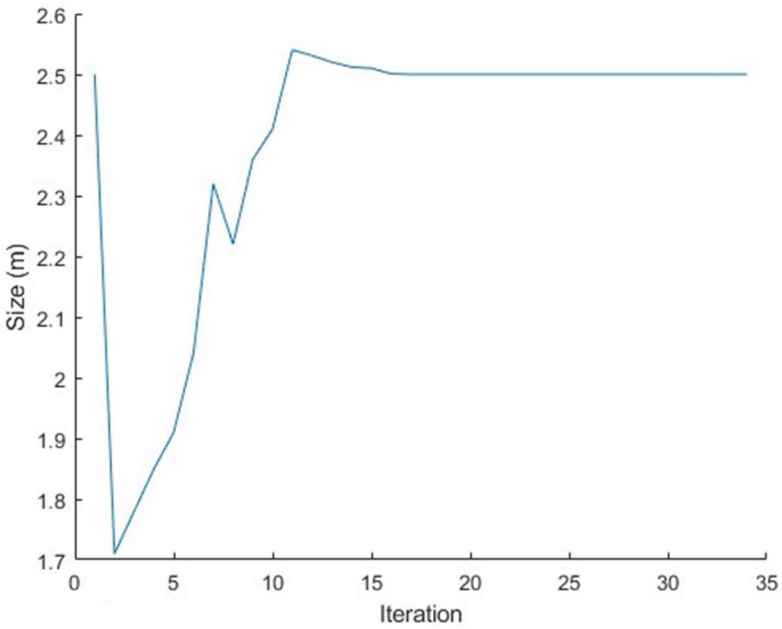


Figure 3.25: The value of size parameter at each iteration for an example weight set.

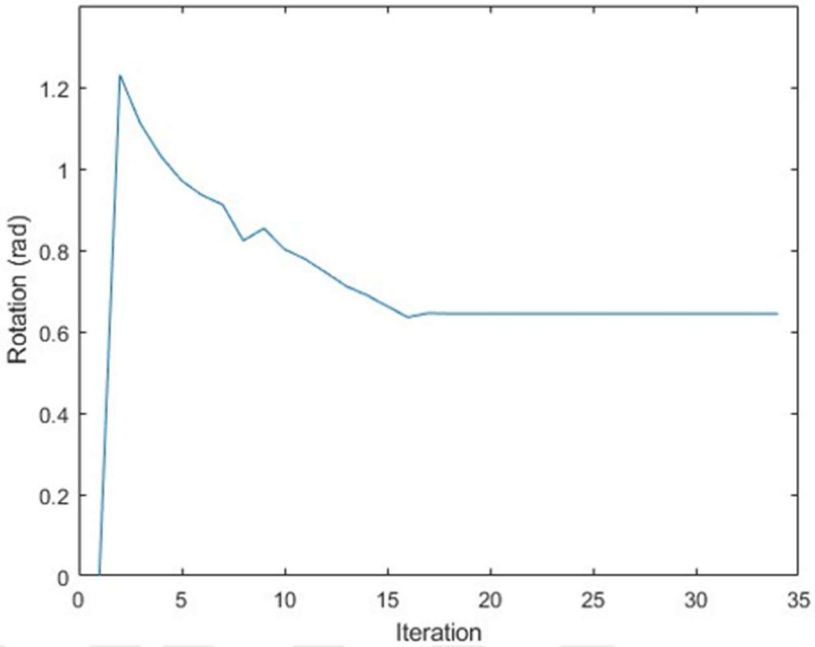


Figure 3.26: The value of rotation parameter at each iteration for an example weight set.

If another weight set which consist of relatively lower size cost weight than other factors is selected, such as $w_t = 1$, $w_s = 0.2$ and $w_r = 1$, the formation would prefer changing its size rather than rotating. The result of such a weight selection is given in Figure 3.27. In the figure, a view from the XY cross section is given and formation is fitted into the obstacle-free region by changing its size. Figure 3.28 provides an illustration of the size parameter values throughout each iteration of the optimization process. The optimization commences with the size parameter set at the preferred value of 2.5 meters and eventually converges to a size of 2 meters which is the maximum size in order to ensure that the formation can fit within the obstacle-free region without rotating. Additionally, Figure 3.29 showcases the rotation parameter values at each iteration of the optimization process. The search for the optimal rotation value initiates at the preferred rotation amount of 0 radians and ultimately converges back to 0 radians. This convergence occurs due to the relatively lower weight assigned to the size cost in comparison to the rotational cost.

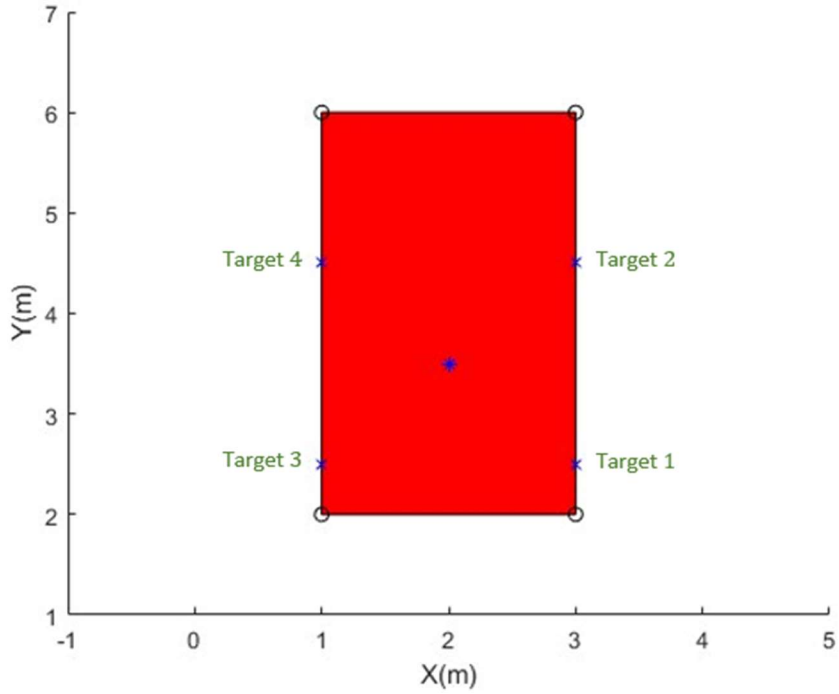


Figure 3.27: The optimized formation configuration for the second example weight set.

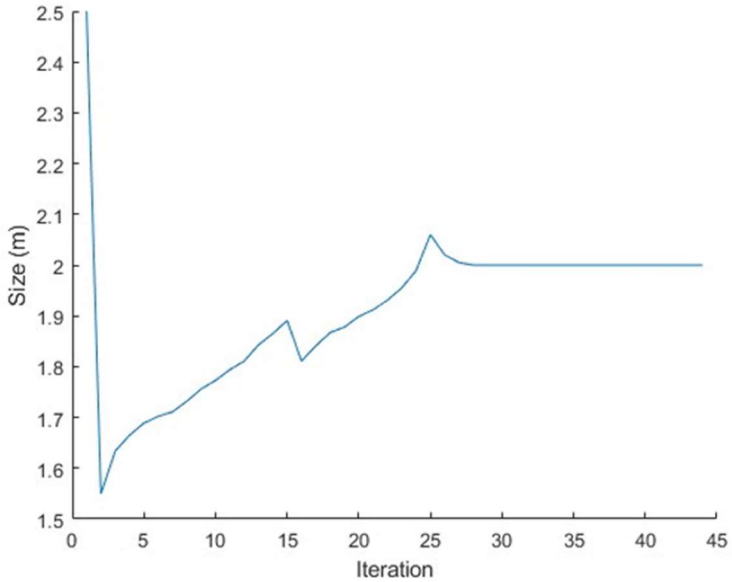


Figure 3.28: The value of size parameter at each iteration for the second example weight set.

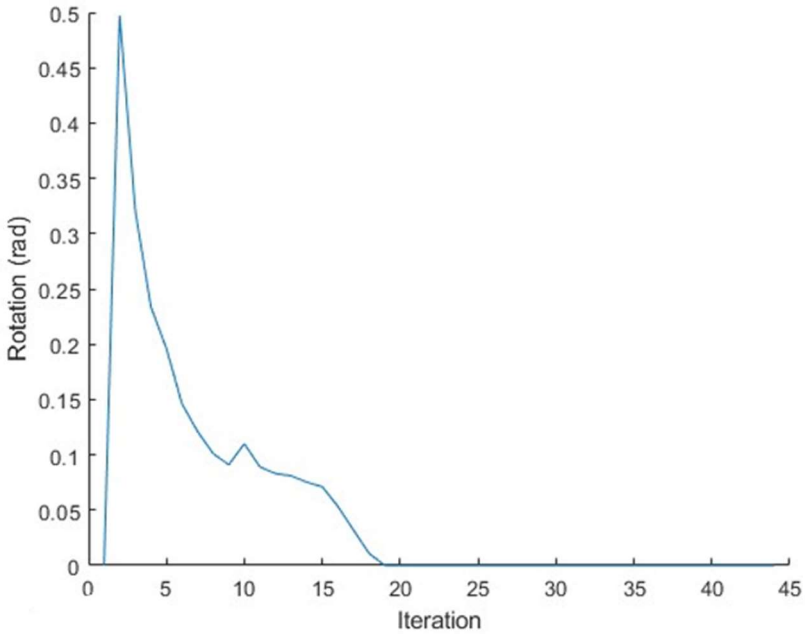


Figure 3.29: The value of rotation parameter at each iteration for the second example weight set.

3.6.4 Switching Cost

The last cost is associated with the switching between different formation shapes. This is an optional cost in our case, since most cases are executed in a way that switching is disabled. Switching cost is zero if the considered shape is the preferred shape. If not, it is a constant and positive cost which is determined by the user.

$$J_{sw} = \begin{cases} 0 & if\ F = F_{pref} \\ c_{sw} & else \end{cases} \tag{3-18}$$

where F is the optimized formation type and F_{pref} is the preferred formation type. c_{sw} is the constant, determined by the user. In Figure 3.30, an example case where size and rotational cost weights high enough such that a formation switching is

preferred is given. In this example, a team of 6 UAVs starts navigating in a polygonal shape, which is the preferred formation shape. In order to pass a corridor, UAVs then switch to the lattice shape which occupies less area for the same interrobot distance. After passing the corridor, UAVs return back to the preferred shape, which is polygon.

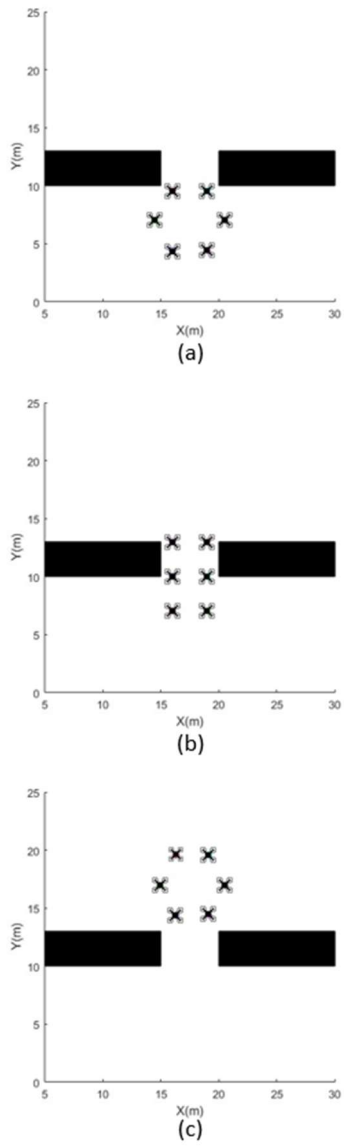


Figure 3.30: A group of 6 UAVs (a) at starting position in polygon shape, (b) in a lattice formation to pass the obstacle (c) in polygonal shape again after passing the corridor.

3.7 Assignment to Target Positions

After solving the optimization problem and deciding on formation parameters defining its shape, a set of target positions are obtained, which are then assigned to the UAVs to form the desired formation shape. These target positions are assigned to UAVs with the objective of minimizing the maximum path length travelled by a UAV. This objective is chosen in order to minimize the mission time. The number of target locations is equal to the number of UAVs and each robot knows all target locations and positions of all other UAVs. This knowledge enables the resolution of the linear bottleneck assignment problem optimally. The linear bottleneck assignment problem refers to the task of assigning the target positions to UAVs in a manner that minimizes the maximum path length traveled by any UAV. By solving this problem optimally, the assignment process ensures that each UAV is assigned a target position that contributes to minimizing their individual travel distances, ultimately leading to an optimized overall mission time. In this work, the method presented in [12] is used.

The assignment is not done at all steps. Whenever the formation is navigating in a shape without changing it, previous assignments are still optimal in terms of mission time. A $reassign$ flag is raised when shape changing decision is made. If that flag is raised, assignment is done after a set of target positions are generated with the shape generation method. As an example, consider a case with 4 UAVs shown in Figure 3.31 with current positions of UAVs are shown in black and computed targets for the next step are shown in orange. X and Y positions of targets are given in Table 3.8. Current assignments of UAVs are given in Table 3.9.

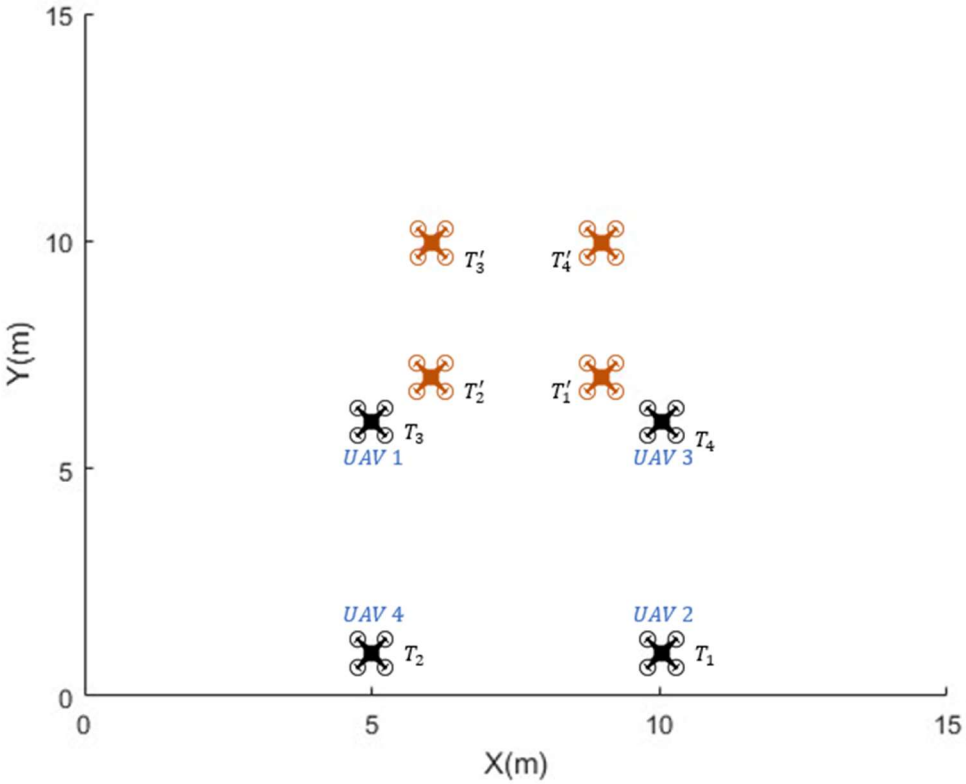


Figure 3.31: A case where a team of 4 UAVs are in formation flight in a rectangular shape, with their current positions shown in black and calculated targets for the next step are colored in orange.

Table 3.8: Target positions.

Targets	*X(m)*	*Y(m)*
T_1	10	1
T_2	5	1
T_3	5	6
T_4	10	6
T_1'	9	7
T_2'	6	7
T_3'	6	10
T_4'	9	10

Table 3.9: Assigned target of each UAV.

UAV #	*Assigned Target*
UAV 1	T_3
UAV 2	T_1
UAV 3	T_4
UAV 4	T_2

Then, the cost matrix with elements c_{ij}, which is the cost of assigning UAV i to T_j' with cost being defined as the Euclidean distance between current position of the UAV and the candidate target position, becomes

$$C = \begin{bmatrix} 4.12 & 1.41 & 4.12 & 5.66 \\ 6.08 & 7.21 & 9.85 & 9.06 \\ 1.41 & 4.12 & 5.66 & 4.12 \\ 7.21 & 6.08 & 9.06 & 9.85 \end{bmatrix}$$

When the cost matrix above is solved with [12], new assignments for each UAV are as in Table 3.10. This assignment results in a maximum path length of 6.08 meters for all UAVs. This result is in accordance with our objective, which is minimizing the maximum path length of all UAVs, since for UAVs 2 and 4, 6.08 meters is already the minimum path length that they can be assigned to. Note that the ordering of assignments is the same as previous assignments given in Table 3.9.

Table 3.10: New assigned target of each UAV.

UAV #	*Assigned Target*
UAV 1	T_3'
UAV 2	T_1'
UAV 3	T_4'
UAV 4	T_2'

Since the order of the assignments does not change as in the given example, we do not need to execute the assignment procedure at all times. This approach offers a substantial reduction in computational requirements since the UAVs typically navigate in a stable manner without switching to a different formation shape for a significant duration. However, if, for example, a hexagonal formation is divided into two triangular formations, then the previous assignments might become meaningless. Therefore, if shape switching, splitting or merging is not present between two steps, previous assignments are used directly. Also note that, when the formation splits, the assignment problem transforms from assigning n UAVs to n targets to assigning n/2 UAVs to n/2 targets. Consider a scenario where there are initially 6 UAVs forming a hexagonal shape. If the formation splits into two triangular formations, each containing 3 UAVs, the assignment problem is no longer about assigning all 6 UAVs to targets. Instead, it involves assigning the UAVs within each triangular formation to their respective targets, resulting in an assignment problem of n/2 UAVs to n/2 targets.

CHAPTER 4

RESULTS AND DISCUSSION

In this section, the detailed description of the simulation environment is given together with an in-depth analysis of performance results and discussions. The selected parameters and results of each scenario are also clearly discussed in this section.

4.1 Simulation Environment

The simulation architecture consists of a server that contains a graphics engine for environment visualizations and retrieving state of the environment and a simulation platform for simulating UAV physics as well as retrieving sensor data and commanding UAVs. Our algorithm, which is implemented in MATLAB, interacts with the simulation platform in order to retrieve states of UAVs and sensor data describing the environment. In Figure 4.1, the diagram of the simulation architecture is given. Microsoft AirSim [7] is a simulation platform that is launched in 2017, targeting simulations for aerial vehicles and autonomy. In addition to aerial vehicles, it is able to simulate ground vehicles too. AirSim is developed as a plugin to Unreal Engine 4, which is a popular 3D graphics engine. This integration enables using highly realistic visuals in simulations. AirSim exposes APIs to publish commands to low-level controller such as control commands, and to listen to data such as sensor data or vehicle states. These APIs are accessible with languages like Python, C++/C# and Java. Python APIs are utilized in this work.

While the flight dynamics model used in the simulator is not open-source, AirSim offers a framework for creating a vehicle representation as a solid object with the potential for a flexible assortment of actuators that produce forces and rotations. This vehicle model encompasses attributes like mass, inertia, coefficients relating to

linear and rotational drag, and coefficients of friction and rebound. These specifications serve as inputs for the physics engine to calculate the dynamics of the rigid body. Coefficients used in the calculation of lift and drag forces are directly retrieved from the empiricial data given by manufacturers. Wind gusts, turbulence, and crosswinds which can affect the vehicle's stability and control are modeled using simplified turbulence models. To simulate sensors like barometer, gyroscope, accelerometer, magnetometer and GPS, the ground truth data computed by the physics engine itself is used.

AirSim has a built-in flight controller called 'simple_flight'. It uses a cascaded PID control system which is able to receive command inputs as angle rate, velocity or position. At the outermost level, a PID controller for position control is present, which drives the linear velocity controller. The linear velocity controller which is also a PID controller drives the innermost controller responsible for controller angle rates. Depending on the level of the input to the flight controller, different PID controllers are activated. In our simulations, we activate all PID controllers since we give position inputs to the flight controller. The flight controller directly uses the ground truth state of the aircraft. Gains of each PID controller are optimized for each quadrotor configuration although it is possible to manually set these gains. Details of the architecture and how controller gains are determined are not available.

Positions of UAVs and sensor data are read through AirSim's Python API. Lidar scans are required in order to detect obstacles around UAVs. Since a spherical detection range is assumed, each UAV is equipped with a lidar that fully covers horizontal field of view and a high vertical field of view, which is $\pm 30°$. Lidar data is updated at a frequency of 10Hz. The range of sensors are configurable and the selected value is given in each simulation scenario. AirSim's Python API allows users to control multiple UAVs from a single script.

Retrieved position and sensor data are sent to MATLAB via MATLAB Engine API for Python and all calculations for a time step are done in MATLAB. Outputs of calculations, which are target positions of UAVs for that time step are sent as

commands by again using AirSim Python API. Actual positions of UAVs are recorded using AirSim's recording feature, which logs position and attitude of each UAV at a frequency of 50Hz.

The proposed algorithm is implemented in a centralized manner, in which it is assumed that each UAV can communicate with all other UAVs. No communication delay is assumed in the simulations. Duplicate operations like formation optimization and assignment are computed for each group only once in each timestep, instead of being computed by each UAV. In simulations, it is also assumed that each UAV knows the positions of all other UAVs accurately, since the simulation environment that we use provide the ground truth directly. Lidar sensors utilized in the simulations also provide ground truth about the environment. Each UAV has a limited detection capabilities whose radius is defined in each scenario. We provide a computation time analysis in this chapter based on centralized computations done on a computer.

The algorithm is simulated and verified with two scenarios. First scenario imitates a forest with many trees such as in Figure 4.2 and the team tries to avoid colliding with them. Second environment contains prismatic obstacles with rectangular faces in different sizes, to imitate an urban environment with buildings like high-rises as in Figure 4.13. For each scenario, the set of selected parameters and results of the simulation are given in this chapter, in detail. In both scenarios, teams consist of 6 UAVs.

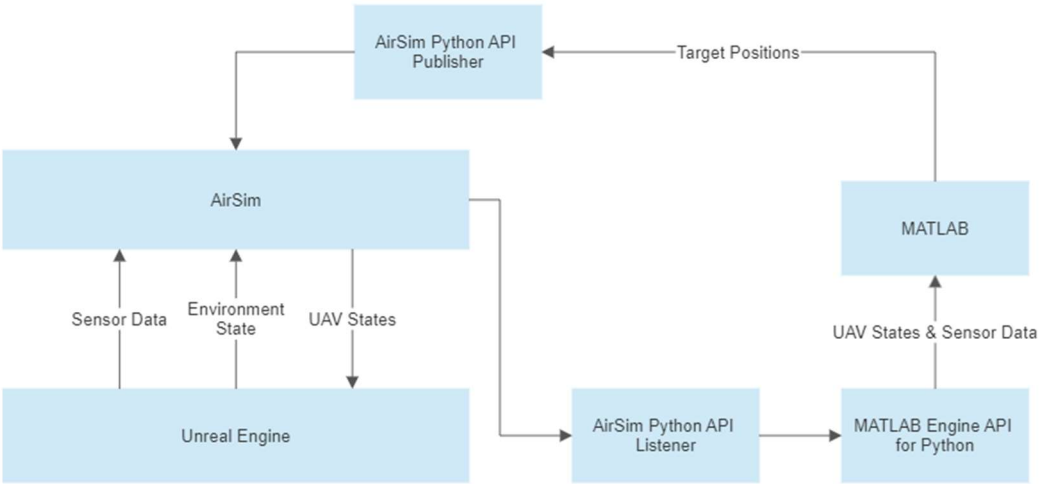


Figure 4.1: Simulation architecture.

4.2 Scenario 1

The forest-like environment used in this simulation consists of many circular obstacles which are tree trunks and branches. Our proposed method can be utilized in forest environments for operations such as fire detection. In such a cluttered environment, our proposed algorithm is able to demonstrate its capabilities such as splitting, merging, shrinking and rotating. An example screenshot of the environment for scenario 1 is given in Figure 4.2, which is retrieved from Unreal Engine. Figure 4.3 represents a 2D section cut at a certain level of the forest. The black full circles represents tree trunks in the 2D section. The green mark in the Figure 4.3 represents the initial center of the formation and the red mark is the target center of the formation.

Figure 4.2: An example screenshot of the environment of scenario 1.

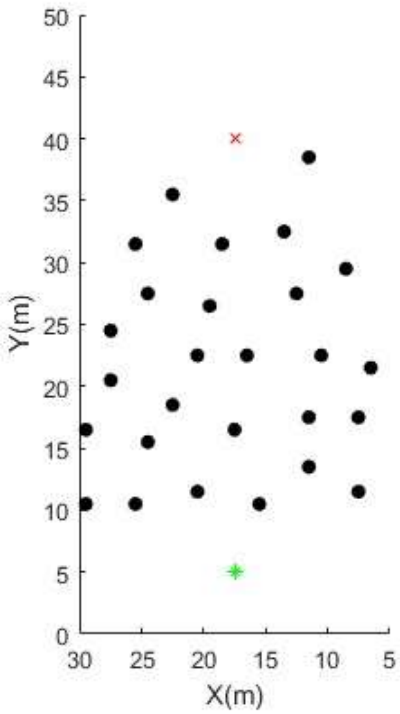


Figure 4.3: Simplified presentation of the environment in scenario 1.

Parameters used in scenario 1 are given in Table 4.1. w_a and w_d are weights used in direction decision step, where w_a represents the weight for the deviation from the nominal angle, and w_d represents the weight for the distance factor. w_t, w_s and w_r are weights used in formation optimization step, where w_t is the weight for the translational cost, w_s is the weight for the size cost and w_r is the weight for the rotational cost. A balanced set of weights is selected in a way that a combination of all features of the proposed method can be demonstrated, without relying on a single behavior to navigate in the environment. For example, selecting a small w_t leads the formation to always go around obstacles instead of shrinking or rotating to move through narrow corridors. s_{pref} is the preferred size, which is the size that the formation restores to, whenever the environmental threats are cleared. s_{lower} represents the lower bound of the formation size, which is the minimum size ensuring the UAVs are as close as possible to each other while maintaining a safe distance. R_{det} is the detection radius of UAVs, which is assumed to be the same for all members of the team in all our applications. v_{pref} is the preferred velocity of UAVs, used by the low-level controller. X_0, Y_0 and Z_0 represent the position of the initial center of the formation, located at the entrance of the forest at one side. Note that, the simulation starts with a formation centered around that point with the preferred size. X_{target}, Y_{target} and Z_{target} represent the position of the target center of the formation. $step_length$ is the length of planning horizon at each step, which is determined in accordance with the computation times in order to allow computations to be completed before target positions are reached by UAVs to have uninterrupted flights. Discussions on computations time are given later in this chapter.

Table 4.1: Parameters for scenario 1.

Parameter	*Value*
w_a	3
w_d	4
w_t	1
w_s	0.5
w_r	0.45
s_{pref}	1.4 m
s_{lower}	1 m
R_{det}	4 m
v_{pref}	1.6 m/s
$step_length$	1.1 m
X_0	17.5 m
Y_0	5 m
Z_0	1.57 m
X_{target}	17.5 m
Y_{target}	40 m
Z_{target}	1.57 m

The simulation results of scenario 1 demonstrate that the proposed formation control algorithm effectively maintains the formation while avoiding colliding with trees. Trajectories of UAVs during the mission shows that the team executes a smooth flight while navigating in the environment. In Table 4.2, total paths covered by each UAV are given. The Euclidean distance between the initial position and the goal position is 35 meters and path lengths are higher than this value because of behaviors exhibited by the formation to avoid obstacles like formation size changes, rotations, translating away from obstacles and low-level control inaccuracies. Time to

complete the mission in scenario 1 is measured at 33.75 seconds which is measured from the start of the flight to the end where the formation center reaches to the target position.

Table 4.2: Path lengths of UAVs for scenario 1.

UAV #	*Path length (m)*
$UAV\ 1$	36.39
$UAV\ 2$	36.87
$UAV\ 3$	39.25
$UAV\ 4$	36.96
$UAV\ 5$	36.52
$UAV\ 6$	37.51

In Figure 4.4, path of each UAV is drawn onto the 2D cross section of the environment. The green dot represents the initial center of the formation and the red cross represents the target center of the formation. Tree trunks or tree branches are represented as black circles and the trajectory of each UAV is given in different colors as shown in the legend on the figure. The team starts in a hexagonal shape around the initial center. Around Y≈14 meters in Figure 4.4, splitting occurs since a tree trunk is encountered nearly aligned with the center of the formation. Around Y≈28 meters, two groups are merged back into one group since the area between two groups is cleared. Note that when the splitting occurs, they divide into two teams of 3 UAVs in a triangular shape. Lastly, they finish the mission in a hexagonal shape around the target center location.

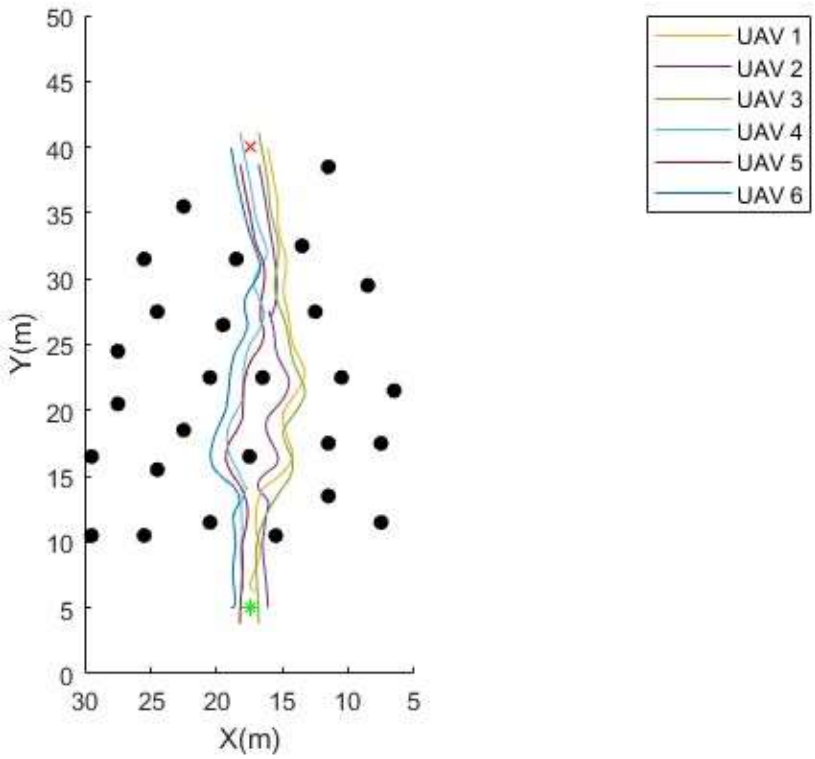


Figure 4.4: Paths covered by UAVs while executing scenario 1.

In Figure 4.5, several snapshots of obstacle-free convex polytopes from 2D cross section are shown at different steps, with resulting formations at these steps fitted into these regions. Red areas represent the computed obstacle-free polytopes, black circles represent tree trunks and branches and dots inside these areas represent target positions of UAVs at these steps. Faces of polytopes are shown as black lines. In Figure 4.5a, the region avoiding two closest tree structures is shown, computed at step 5. As in Figure 4.4 where the team navigates in two separate groups between Y ≈ 14 m and Y ≈ 28 m, in Figure 4.5b, 4.5c and 4.5d, the team splits into two groups because of the incoming tree trunk near the center of the formation and there are two separate polytopes computed by each team, at steps 10, 15 and 20 respectively. In Figure 4.5e which corresponds to the Y ≈ 30 m part of Figure 4.4, the team has to merge again since two groups are close to each other and the area between them is clear and after merging is done, the whole team reaches a consensus on a single obstacle-free region since they reduce to a single group again. Note that, in all these

figures, obstacle-free regions maintain a safe distance from obstacles, since each UAV is treated as points in the algorithm and UAVs might end up colliding with tree structures if distances between obstacles and target positions of UAVs are less than the radius of UAVs.

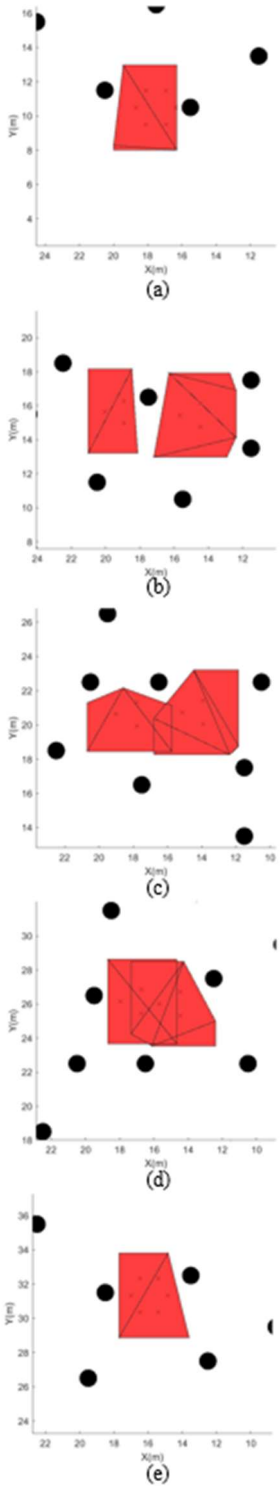


Figure 4.5: Example snapshots of convex regions generated by the team in scenario 1 at step = (a) 5, (b) 10, (c) 15, (d) 20 and (e) 25.

In Figure 4.6, positions of UAVs in the X axis with respect to time are given. These positions are retrieved from AirSim at 50Hz frequency. As UAV 1, 2 and 3 are members of the right group formed by the splitting that occurs at Y ≈ 14 m in Figure 4.4, they navigate in further lower values than other UAVs in X-axis between t ≈ 6 sec and t ≈ 20 sec. The reverse is true for the rest of UAVs, as they are part of the left group formed after splitting at Y ≈ 14 m in Figure 4.4. Figure 4.7 and Figure 4.8 show Y and Z positions of UAVs with respect to time, respectively. As the mission primarily involves advancing in Y-axis since the difference between the initial center position and the goal position is in Y-axis, Y positions of all UAVs increase at a nearly constant rate in Figure 4.7. Note that, while X and Y positions are smooth, Z position is relatively noisy because of low-level controller, rather than the planning algorithm. In scenario 1, since the initial and goal Z positions are kept relatively constant in the 2D cross section and the scenario does not contain any environmental feature that will require deviations in Z-axis, the only reason for our proposed method to command varying target positions to low-level controller in Z-axis is the rotation of the formation, which is computed a positive value around t ≈ 4 sec and t ≈ 21 sec, observed with abrupt changes in Z-axis in Figure 4.8.

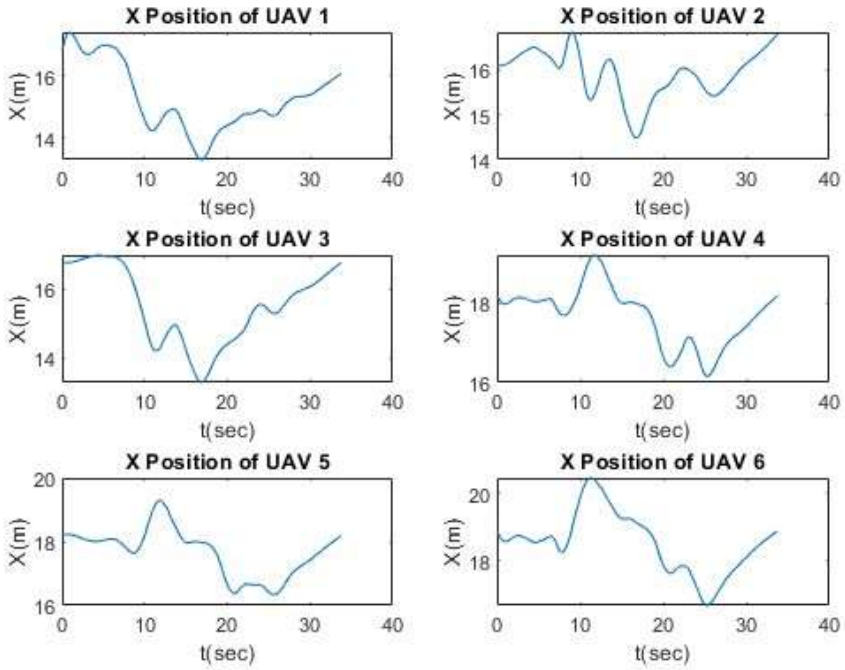


Figure 4.6: X positions of UAVs while executing scenario 1.

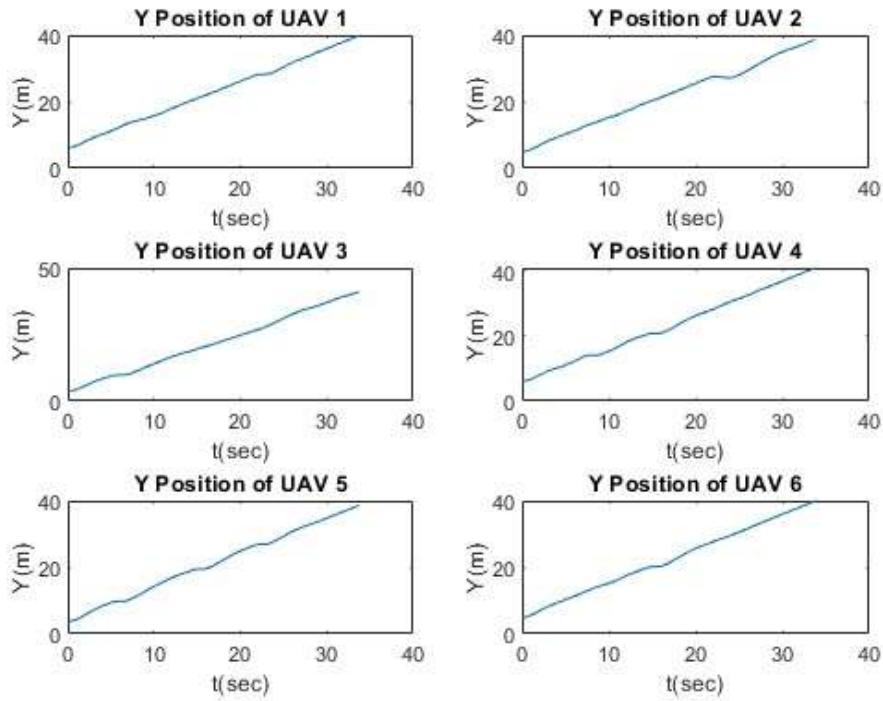


Figure 4.7: Y positions of UAVs while executing scenario 1.

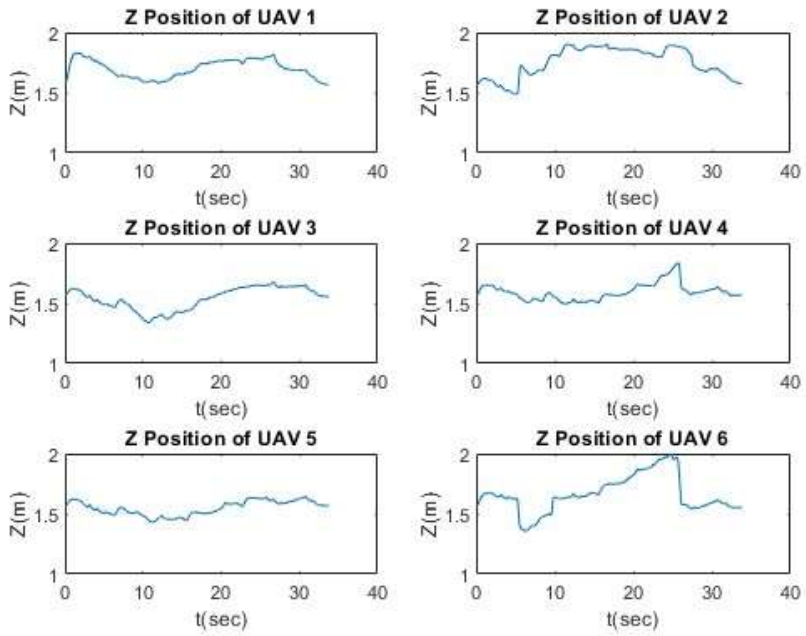


Figure 4.8: Z positions of UAVs while executing scenario 1.

In Figure 4.9, the splitting state of the team is given while executing scenario 1. The value 0 means that the team navigates as a single group while the value 1 means that the team is split into two groups. In scenario 1, splitting occurs at step 8 when the center of the formation is at Y ≈ 14 m where the team is confronted by an obstacle close to the center of the formation as in Figure 4.4 and merging occurs at step 22 where the center of the formation is at Y ≈ 28 m where two groups are close to each and the area between them is obstacle-free as in Figure 4.4.

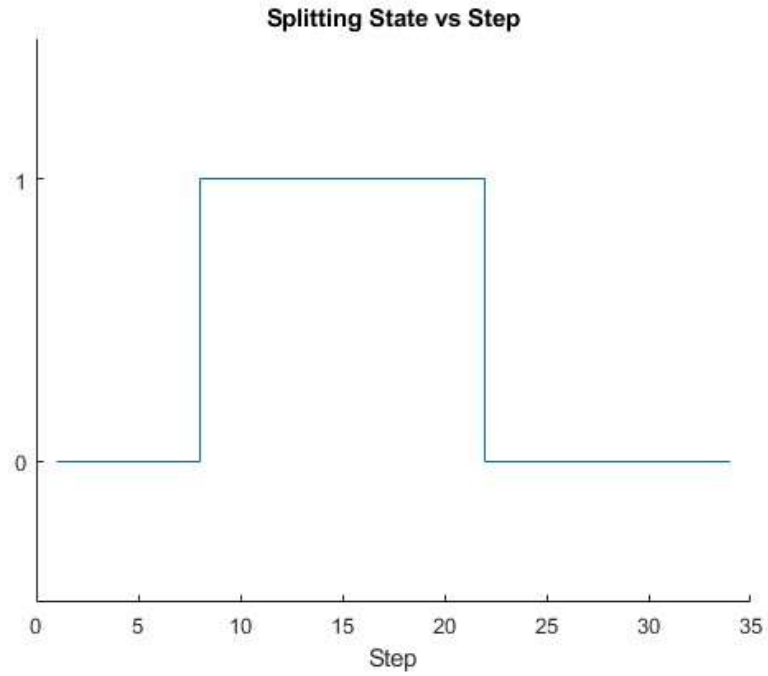


Figure 4.9: Splitting state of the team while executing scenario 1.

In Figure 4.10, the plot of size decision of the algorithm at each step is given. When the team is navigating through narrow corridors, it shrinks and expands back to preferred size when they leave these corridors behind. Note that, when the team splits, which is true between step 8 and 22, different lines for each group are given in the figure, shown as blue line for group 1 and orange line for group 2. In Figure 4.11, the rotation decision of the algorithm at each step is given. The formation rotates when the team is not able to pass through obstacles even at the lower bound of the formation size; so that UAVs can not get any closer to each other but the team needs to decrease the projection of the area they occupy in 2D cross section further because of the environmental restrictions like narrow passages. This behavior results from the selection of size and rotation cost weights; selecting a lower rotation cost weight would lead the team to rotate more frequently rather than changing the size.

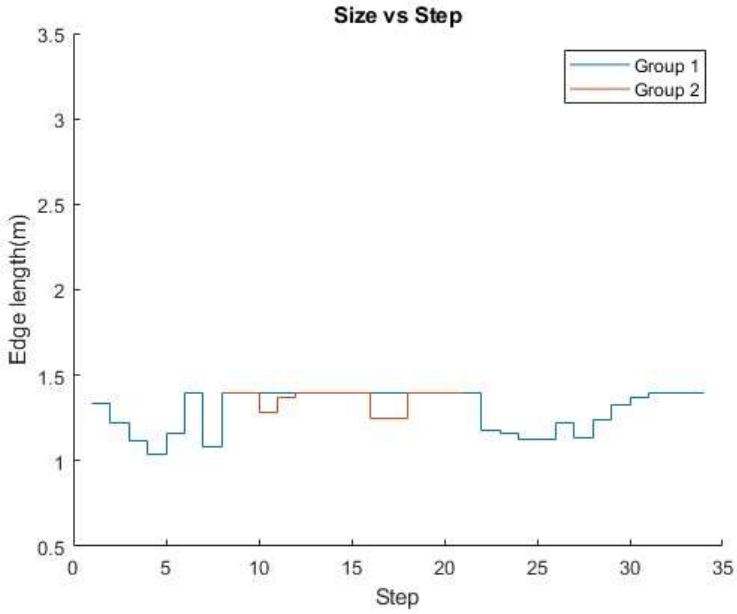


Figure 4.10: Size output of the algorithm at each planning step while executing scenario 1.

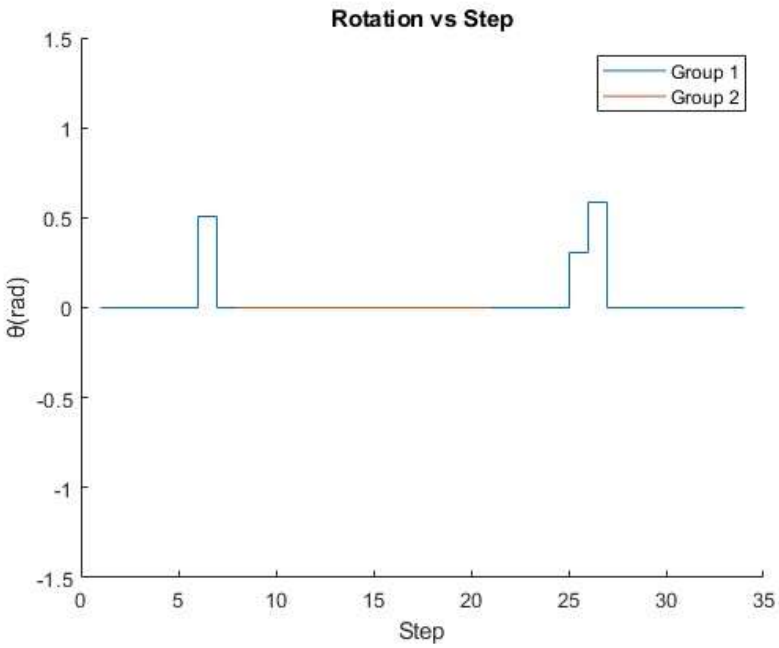


Figure 4.11: Rotation output of the algorithm at each planning step while executing scenario 1.

Simulations are performed with a computer having 6-core Intel i7 CPU@3.2GHz. The mean and maximum computation times for a step of the proposed method for scenario 1 are 268.4 milliseconds and 476.5 milliseconds, respectively. Therefore, for a case with 6 UAVs in the team and computational capabilities similar to ours, it is appropriate to take 476.5 milliseconds computation time as the maximum computation time. Taking it as reference, Figure 4.12 shows the minimum step length that should be taken with respect to the preferred velocity of UAVs. This limitation roots from the fact computations need to be done before completing the previous step, in order to obtain the next target position before reaching the current target position and have a smooth, uninterrupted flight. In performed simulations, preferred velocities are taken as $1.6\ m/s$ which corresponds to $0.76\ m$ minimum step length. Step length is taken as 1.1 meters in scenarios 1, respectively, which is appropriate when considering this computational limitation. Note that, in simulations, the algorithm was executed in a centralized manner; in the case of distributed employment of the algorithm, computation needs can be slightly lower, which enables the proposed method to be employed in distributed manner in a real-world scenario.

The minimum step length parameter also serves as a measure of maximum frequency of emerging environmental features which the formation can react to. For example, in our case, if the formation is confronted by unknown obstacles, that the formation cannot detect beforehand, at every $0.5\ m$, they might not be able to react to these environmental features. Note that, in addition to computational limitation, the detection range of UAVs also restricts the reaction capabilities of the formation. If the step length is chosen to be higher than the detection radius of UAVs, they are not going to be able to sense and react to environmental elements beyond their detection ranges at that step and may end up in collision.

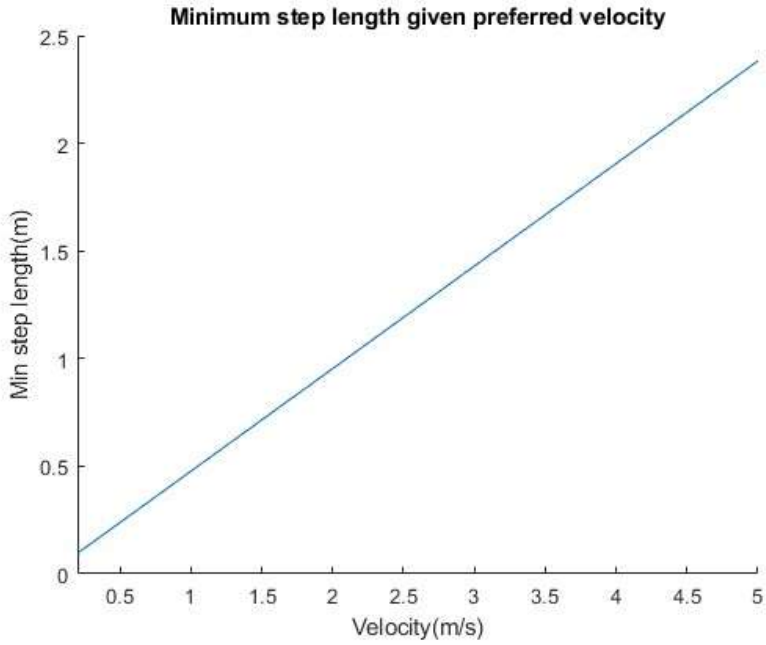


Figure 4.12: Minimum step length with respect to preferred velocity.

4.3 Scenario 2

Figure 4.13 displays a screenshot of the environment for scenario 2, taken from Unreal Engine depicting a crowded urban environment. Additionally, a 2D cross section of the same environment is given taken across any predefined heights in Figure 4.14. The 2D cross section defines a horizontal plane taken at the altitude at which UAVs conduct their formation flight. In Figure 4.14, the green mark indicates the initial center of the formation, while the red mark represents the target center of the formation.

Figure 4.13: An example screenshot of the environment of scenario 2.

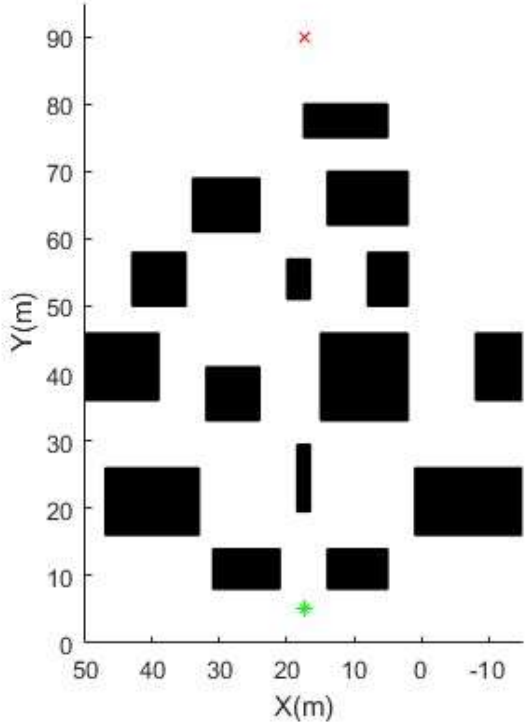


Figure 4.14: Simplified presentation of the environment in scenario 2.

Parameters used in scenario 2 are given in Table 4.3. In order to demonstrate a combination of all features of the proposed method without relying on a single behavior to navigate in the environment, a set of weights, which is the same set used in scenario 1, is selected to maintain balance, which are w_a, w_d, w_t, w_s and w_r. Since the size of the area used in this scenario is higher than the area of scenario 1 and in order to demonstrate capabilities of our proposed method with different sensing capabilities, a higher detection radius R_{det} is selected with a combination of higher preferred size s_{pref}, lower bound of the size s_{lower} and the length of the planning horizon $step_length$. A larger detection range allows UAVs to take obstacles further away from them into account; therefore, a higher planning horizon length can be selected to act upon incoming environmental features.

Table 4.3: Parameters for scenario 2.

Parameter	*Value*
w_a	3
w_d	4
w_t	1
w_s	0.5
w_r	0.45
s_{pref}	1.5 m
s_{lower}	1.2 m
R_{det}	7 m
v_{pref}	1.6 m/s
$step_length$	1.5 m
X_0	17.5 m
Y_0	5 m
Z_0	1.57 m
X_{target}	17.5 m
Y_{target}	90 m
Z_{target}	1.57 m

The findings from the simulation of scenario 2 indicate that the suggested algorithm for formation control successfully maintains the desired formation of UAVs while avoiding collisions with rectangular obstacles. Table 4.4 provides information on the total paths covered by each UAV. The path lengths between the initial and goal positions exceeded 85 meters, which is the Euclidean distance between the initial center and the goal positions, due to various behaviors exhibited by the formation to avoid obstacles. These behaviors include formation size changes, rotations, translating away from obstacles, as well as low-level control inaccuracies. The time taken to complete the mission in scenario 2 was 75.75 seconds, which is measured from the start of the flight until the formation center reaches the target position.

Table 4.4: Path lengths of UAVs for scenario 2.

UAV #	*Path length (m)*
$UAV\ 1$	88.25
$UAV\ 2$	91.45
$UAV\ 3$	90.78
$UAV\ 4$	87.46
$UAV\ 5$	87.80
$UAV\ 6$	87.90

Figure 4.15 displays the paths of each UAV drawn onto the 2D cross section of the environment. The initial center of the formation is marked by a green dot, while the target center is represented by a red cross. Obstacles are depicted as black rectangles, and the trajectory of each UAV is distinguished by different colors, as indicated in the figure's legend. The team's starting formation is hexagonal, centered around the initial center. At Y≈15, the team splits into two triangle-shaped groups to go around the incoming obstacle that is aligned with the center of the formation. After the obstacle is passed and two groups are close enough to each other, at Y≈43, they merge into a single group again, forming a hexagonal shape. At Y≈48, the team again encounters an obstacle and splits into two teams. After the obstacle is passed, groups start to get closer to each other to get aligned with the goal positions. At Y≈73, groups merge again and navigates as a single group afterwards. Lastly, the mission is completed with the UAVs forming a hexagonal shape around the target center location.

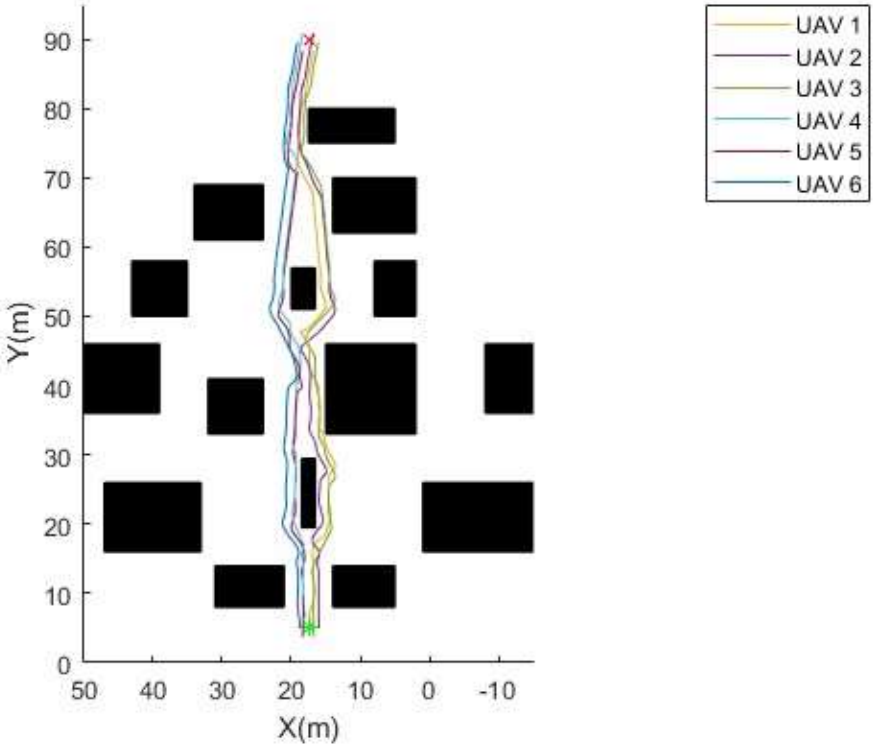


Figure 4.15: Paths covered by UAVs while executing scenario 2.

In Figure 4.16, several snapshots at different steps are shown to demonstrate obstacle-free convex polytopes, similar to Figure 4.5 of scenario 1. Red areas represent the computed obstacle-free polytopes, black circles represent trees and dots inside these areas represent target positions of UAVs at these steps. Faces of polytopes are shown as black lines. In Figure 4.16a, the obstacle-free region in a narrow corridor just before splitting is shown, computed at step 6, which corresponds to Y≈13 part of Figure 4.15. As the team navigates in two groups between Y≈15 and Y≈43, in Figure 4.16b, two separate obstacle-free regions computed by two groups are shown at step 16, around Y≈28. Merging occurs at Y≈43 and the team is in hexagonal shape again until another splitting occurs at Y≈48. In Figure 4.16c, an obstacle-free region computed by the team while being in a single group between two split states is shown at step 26, around Y≈44 part of Figure 4.15. In Figure 4.16d, another snapshot of two obstacle-free regions computed by groups navigating separately is given after second splitting occurs, at step 36. In Figure 4.16e, the team

has merged and they collectively reach a consensus on a single obstacle-free region since they reduce to a single group again, shown at step 46. Again, obstacle-free regions maintain a safe distance from obstacles, as in scenario 1.

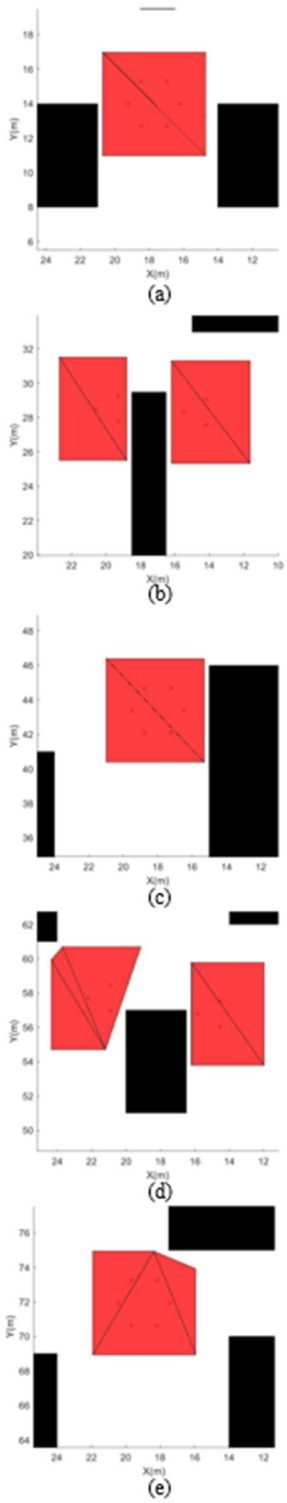


Figure 4.16: Example snapshots of convex regions generated by the team in scenario 2 at step = (a) 6, (b) 16, (c) 26, (d) 36 and (e) 46.

Figure 4.17 illustrates the positions of UAVs along the X-axis over time. UAVs 1, 2, and 3 belong to the right group, which formed after the splitting event at approximately Y ≈ 14 m in Figure 4.15. Consequently, at time t ≈ 7 sec, they move to lower X positions than other 3 UAVs. Conversely, the remaining UAVs are part of the left group formed after the same splitting event, causing them to navigate towards higher X-axis values during this time period. After the obstacle is passed, they recover to their previous X positions since two groups get closer to each other until merging back again at time t ≈ 25 sec. The same sequence is also true for the second splitting case, occurring at t ≈ 33 sec, and Y ≈ 48 m part of Figure 4.15. Figure 4.18 displays the Y positions of the UAVs over time. Since the mission primarily involves advancing along the Y-axis, all UAVs experience a nearly constant increase in their Y positions. This is due to the difference between the initial center position and the goal position being solely along the Y-axis. It is worth noting that while the X and Y positions exhibit smooth trajectories, the Z position appears relatively noisy in Figure 4.19. This noise is primarily attributed to the low-level controller rather than the planning algorithm. In scenario 2, as the initial and goal Z positions remain relatively constant within the 2D cross-section, and there are no environmental features requiring deviations along the Z-axis.

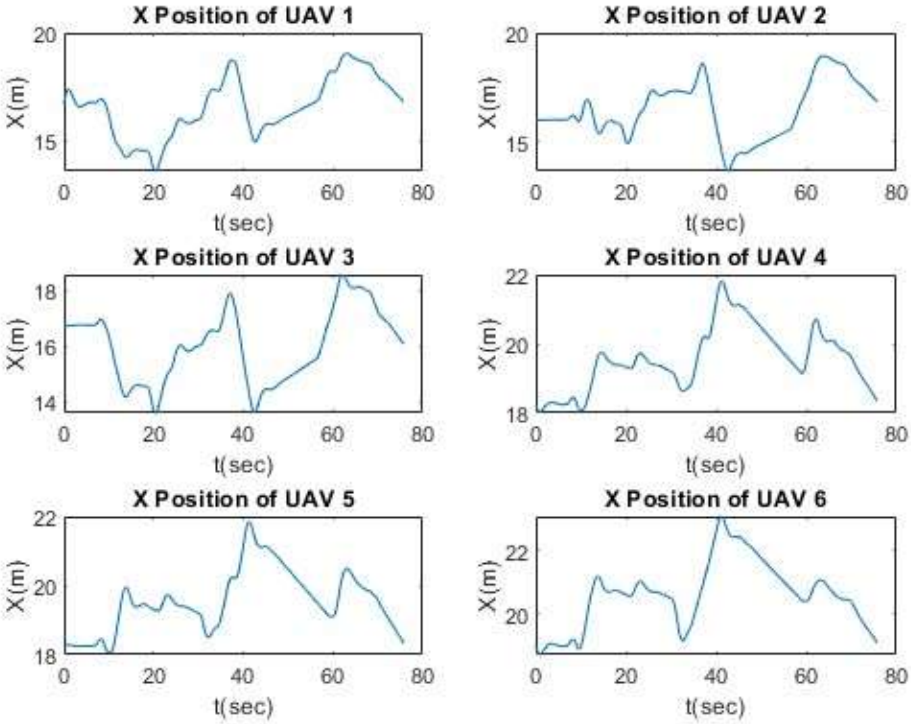


Figure 4.17: X positions of UAVs while executing scenario 2.

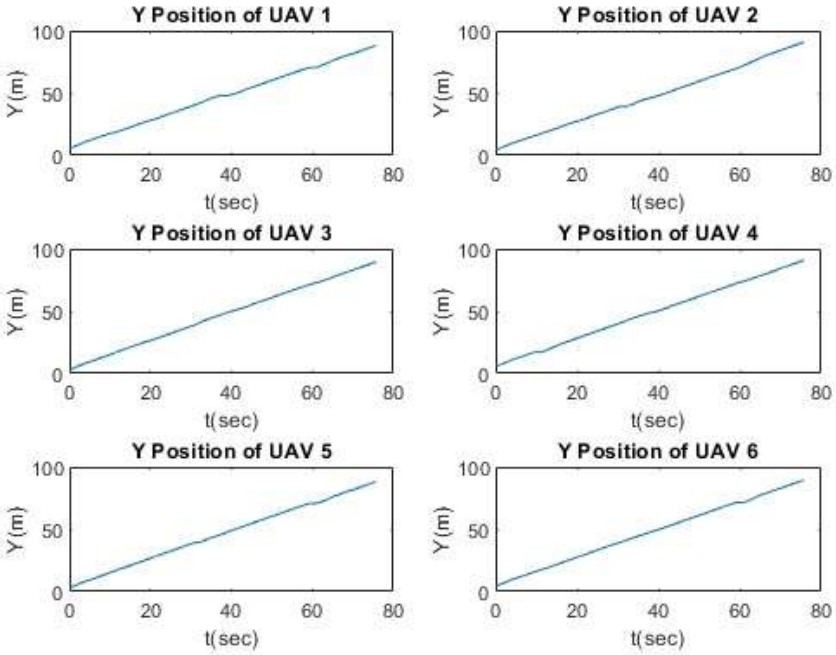


Figure 4.18: Y positions of UAVs while executing scenario 2.

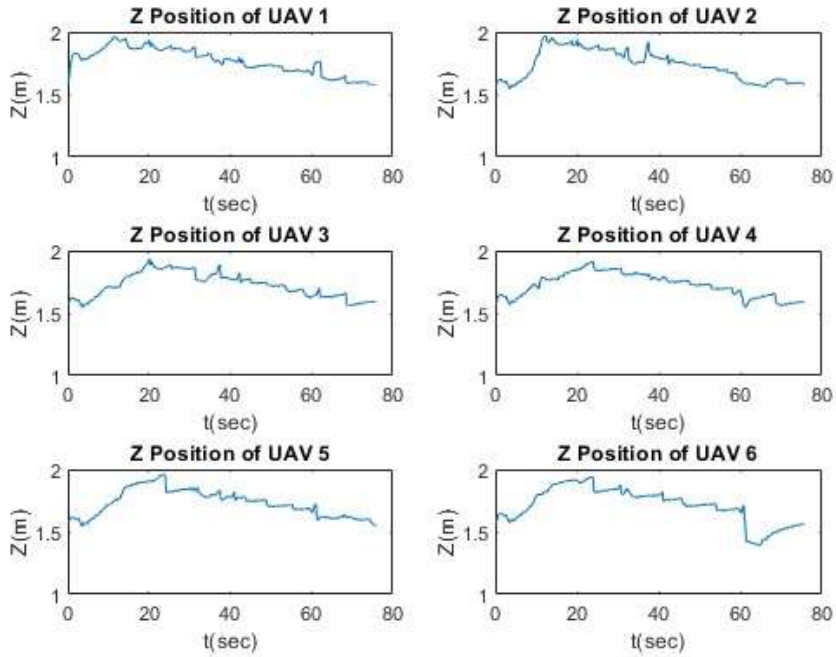


Figure 4.19: Z positions of UAVs while executing scenario 2.

Figure 4.20 presents the depiction of the team's splitting state during the execution of scenario 2. A value of 0 indicates that the team is navigating as a single group, while a value of 1 indicates that the team is split into two groups. In scenario 2, the first instance of splitting occurs at step 8 when the team encounters an obstacle, followed by merging at step 22. The second splitting occurs at step 29, with merging occurring again at step 45.

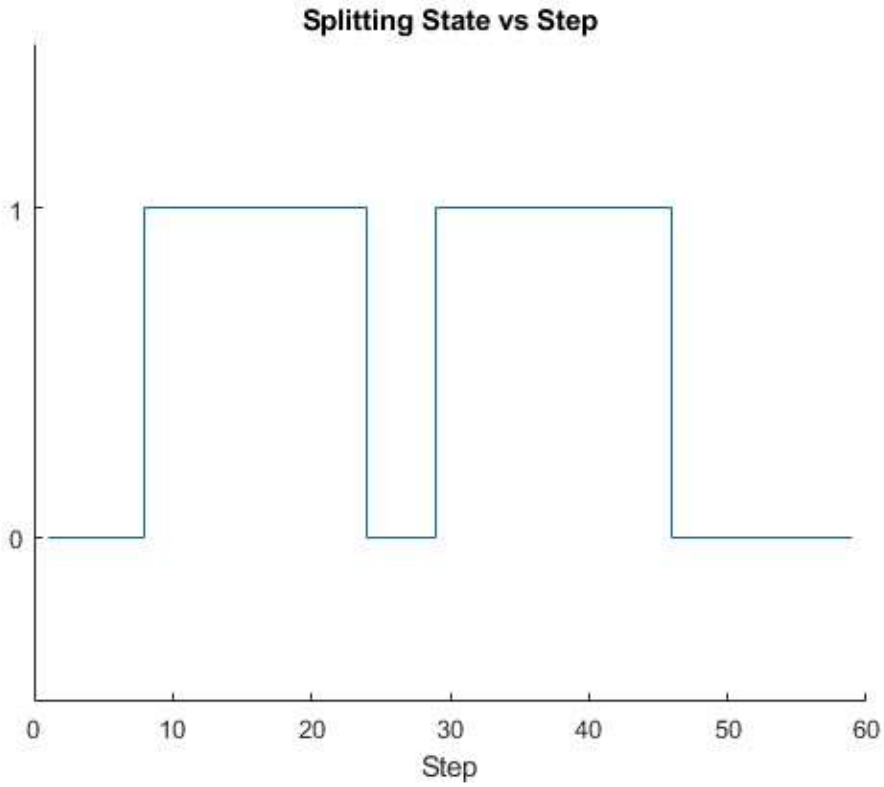


Figure 4.20: Splitting state of the team while executing scenario 2.

In Figure 4.21, the plot of size decision of the algorithm at each step is given. When the team is navigating through narrow corridors, it shrinks and expands back to preferred size when they leave these corridors behind. This size reduction happens at steps between 48 and 53. Note that, when the team splits, which is true between steps 8-22 and 29-45, different lines for each group are given in the figure, shown as blue line for group 1 and orange line for group 2. In Figure 4.22, the rotation decision of the algorithm at each step is given. In situations where the team encounters

obstacles that cannot be traversed even at the lower bound of the formation size, the formation resorts to rotating, which is only true for step 47 in this scenario. This rotation allows the UAVs to maintain their current spacing as they are unable to get any closer to each other. However, the team still needs to reduce the projection of the area they occupy in the 2D cross section due to environmental restrictions such as narrow passages.

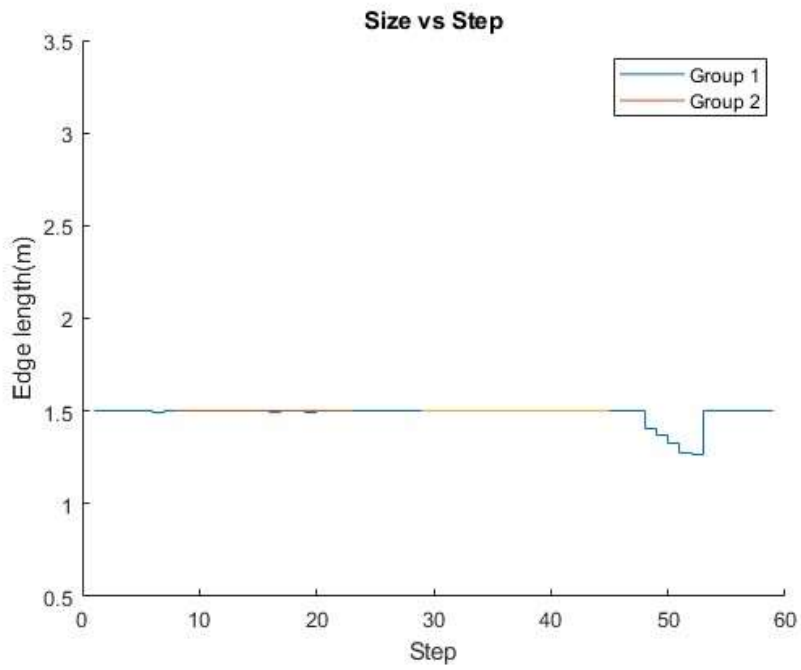


Figure 4.21: Size output of the algorithm at each planning step while executing scenario 2.

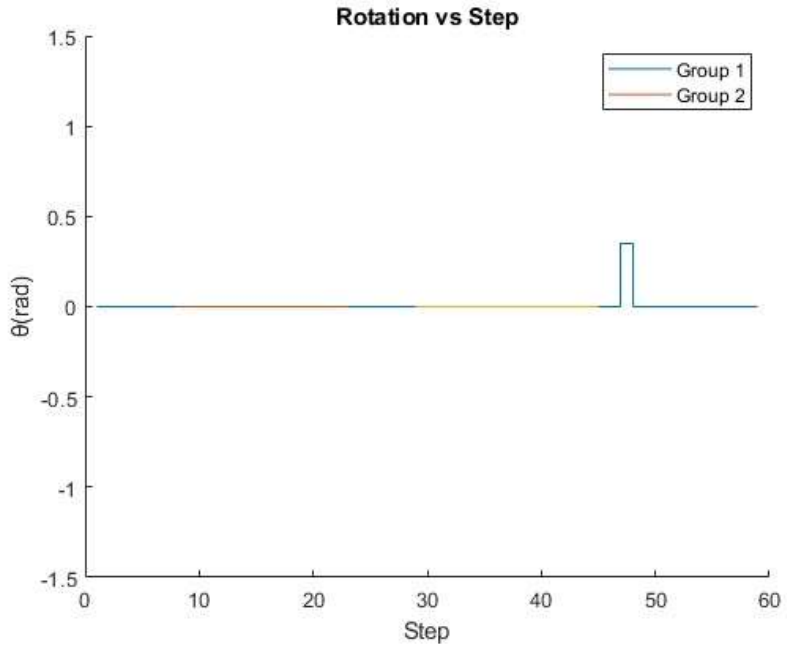


Figure 4.22: Rotation output of the algorithm at each planning step while executing scenario 2.

The mean and maximum computation time per step for scenario 2 is 268.4 milliseconds and 476.5 milliseconds, respectively. Computation times for scenario 1 and scenario 2 are quite similar since the computational complexity of the algorithm is independent of parameters or step length; instead, it is mostly dependent on the number of agents in the team and slightly dependent on complexity of the environment only at the obstacle-free region generation step, IRIS algorithm. However, the most computationally expensive process is the optimization step and IRIS algorithm occupies a small percentage of the runtime. Therefore, limitations on the minimum step length given in scenario 1 hold for scenario 2, too.

In both scenarios, results show that the proposed method is able to navigate a team of UAVs in a cluttered environment while avoiding collisions. Different types of environments, the first one being a forest and the second one being similar to an urban area, simulated in order to verify the capabilities of the proposed method. In both scenarios, UAVs navigate in unknown environments while environmental

features, obstacles, were emerging to them frequently, and they were able to react to these features quickly enough. This is achieved by manipulating the formation shape in terms of size and orientation as shown in Figures 10, 11, 21 and 22. After they pass the environmental threats, they recovered to their original formation shape as desired. In addition to manipulating the formation shape, simulations show that the team was able to successfully split when confronted by an environmental feature and merge again when it is safe. This ability results in more optimal navigation of UAVs while still being able to navigate as a team instead of making decision on their own. It also shows that the proposed method is able to handle different formation shapes during flight if these shapes are defined beforehand. If more formation shapes are utilized, more complex behaviors and enhanced navigation capabilities can be observed.

CHAPTER 5

SENSITIVITY ANALYSIS

In this chapter, we present a sensitivity analysis of our proposed multi-UAV formation control algorithm. Our algorithm utilizes an optimization process to find the optimal formation configuration at each step and involves the use of weights for costs in the objective function. These weights play a crucial role in determining the trade-offs between different behaviors like size change, rotation and translation of the formation in order to avoid sudden emergence of obstacles. In this sensitivity analysis, we investigate the effects of varying these weights on the performance of our algorithm. We analyze the results obtained from simulations and discuss the implications of different weight values on the performance metrics of interest, including average perpendicular distance of the formation from the line between initial center position and target position, average size deviation and average rotation of the formation.

We conduct a series of simulations of the proposed method with varying weight values for the different costs in the objective function. We select a range of weight values for each cost, and systematically vary them while keeping other parameters and conditions constant. For each set of weight values, we execute the algorithm in the simulation environment of scenario 1 of Chapter 4.

In the first part of our sensitivity analysis, we investigate the effects of varying weights in the optimization step of our algorithm. We analyze the performance metrics of average lateral deviation from the nominal line per step, average size deviation per step, and average rotation per step. Considering the environment same as in scenario 1 of simulations and keeping the other parameters constant, effect of changing w_t, weight of translational cost, is analyzed first. The results are shown in Table 5.1. δ_t is the average perpendicular distance of the center of the formation

from the line between initial center and the target position per step, δ_s is the average absolute deviation of the size of the formation from the preferred size per step and δ_r is the average rotation of the formation per step. Note that in Chapter 4, the weight was selected as $w_t = 1$ in scenario 1. It can be seen that, decreasing w_t, the average lateral deviation per step, δ_t, is increased since lower cost is associated with the translation of the formation. With lower translational cost, the behavior of the system shifts from changing size or orientation to finding a way around obstacles by translating the formation. Such behavior increases the total path of UAVs as well. Therefore, average size deviation per step, δ_s, and average rotation of the formation per step, δ_r, decrease with decreasing w_t. On the other hand, increasing translational cost results in lower lateral deviation δ_t, higher average size deviation δ_s, and higher average rotation δ_r. Increasing w_t also results in lower total path of UAVs.

Table 5.1: Results for different translational cost weights.

w_t	$\delta_t(m)$	$\delta_s(m)$	$\delta_r(rad)$
0.5	0.674	0.067	0.035
0.75	0.618	0.118	0.041
1	0.575	0.142	0.044
1.25	0.552	0.159	0.047
1.5	0.495	0.182	0.054

Another set of simulations have been done for the rotational cost weight, w_r. With decreasing w_r, the team intends to change the orientation of the formation on the XY plane in order to avoid obstacles, rather than changing its size or translation. In Table 5.2, by keeping other parameters same as in scenario 1 of Chapter 4, results for different w_r values are given. Note that in Chapter 4, the weight was selected as $w_r = 0.45$ in scenario 1. It can be seen that average size deviation δ_s and lateral

deviation per step δ_t decrease with decreasing w_r and average rotation of the formation per step increases.

Table 5.2: Results for different rotational cost weights.

w_r	$\delta_t(m)$	$\delta_s(m)$	$\delta_r(rad)$
0.25	0.498	0	0.092
0.35	0.512	0.089	0.068
0.45	0.575	0.142	0.044
0.55	0.593	0.161	0.032
0.65	0.613	0.168	0.023

A similar analysis is done for the last cost factor as well, which is the size cost with weight w_s. The size cost punishes the deviation of the selected size from the preferred one through a multiplication with the weight w_s. Similar to previous analyses, simulations with different w_s values while keeping other parameters constant as in scenario 1 of Chapter 4 are executed and results are given in Table 5.3. Note that in Chapter 4, the weight w_s was selected as 0.5 in scenario 1. It can be seen that decreasing w_s, the algorithm becomes more likely to modify the size of the formation rather than rotating or translating it in order to avoid obstacles, which can be inferred from increasing average size deviation δ_s and decreasing average lateral deviation per step δ_t and average rotation per step δ_r. In an overall summary, it can be concluded that three weights of the optimization process have coupled effects with each other since each weight represents a separate behavior. An increase in one of weights decreases the intention of the UAV team to exhibit the corresponding behavior while increasing it for exhibiting the other two behaviors. For example, with higher size cost weight w_s, average size deviation δ_s decreases and average rotation δ_r and lateral deviation δ_t increases.

Table 5.3: Results for different size cost weights.

w_s	$\delta_t(m)$	$\delta_s(m)$	$\delta_r(rad)$
0.3	0.502	0.202	0.014
0.4	0.534	0.181	0.029
0.5	0.575	0.142	0.044
0.6	0.589	0.111	0.049
0.7	0.608	0.087	0.061

In the second part of our sensitivity analysis, we investigated the effects of varying the weights in the direction decision step of our algorithm. For each angle θ in a discrete set Θ, a cost $c_i(\theta)$ is calculated and UAVs reach a consensus by exchanging these costs with each other. There are two factors affecting the cost of the angle, one of which punishes the difference between the target angle α and the angle θ that the cost $c_i(\theta)$ is calculated for, with the associated weight w_a. The second one punishes the distance to obstacles through that angle, θ, with the associated weight w_d. Since the cost is dependent only on these two factors, instead of analyzing each weight separately, the ratio of these two weights can be analyzed. In Table 5.4, results for different ratios $\beta = w_a/w_d$ are given. Note that in Chapter 4, the weight was selected as $\alpha = 0.75$ in scenario 1. With increasing β, the UAV team intends to be closer to the nominal line between the initial center and the goal instead of deviating much in order to avoid obstacles. Therefore, an increase in average size deviation per step and average rotation per step can be seen with increasing β. Similarly, with decreasing β, reverse effects can be seen with higher average lateral deviation δ_t and lower average size deviation δ_s and rotation per step δ_r. Note that, with increasing β, since the intention to get separated decreases, robots are less inclined to split into two teams in order to pass an obstacle.

Table 5.4: Results for different $\beta = w_a/w_d$.

β	$\delta_t(m)$	$\delta_s(m)$	$\delta_r(rad)$
0.25	0.695	0.120	0.037
0.5	0.631	0.134	0.041
0.75	0.575	0.142	0.044
1	0.522	0.154	0.047
1.25	0.473	0.159	0.054

The results of the sensitivity analysis of the proposed method have several important implications for the field of UAV formation control and related applications. Firstly, the analysis provides insights about robustness and stability of the algorithm with respect to changes in the weight values. By systematically varying the weight values and studying their effects on the performance metrics, a smooth shift between different behaviors is observed and stability of the algorithm with different weight selections is shown with results. This information can be useful for real-world applications where UAV formations may encounter varying environmental conditions, uncertainties, and disturbances.

Secondly, the sensitivity analysis allows us to understand the trade-offs between different cost elements in the objective function and the direction decision step. By varying the weights associated with different costs, we can study how changes in these weights impact the performance of the formation control algorithm. For example, we can investigate the effect of prioritizing size deviation over lateral deviation, or the effect of giving more importance to obstacle avoidance over orientation alignment. This information can be used to customize the formation control behavior to suit specific mission requirements or operational constraints.

Furthermore, the sensitivity analysis can provide insights into the adaptability of the algorithm to different mission scenarios and objectives. By studying the performance of the formation control algorithm under different weight values, we can identify

weight configurations that are most suitable for specific mission scenarios or objectives. For example, in missions where obstacle avoidance is critical, weight values that prioritize obstacle avoidance can be chosen, while in missions where maintaining a specific formation size is more important, weight values that prioritize size deviation can be selected. This adaptability of the algorithm can enhance the formation versatility and applicability in different real-world scenarios.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

In this work, we proposed a formation control system tailored for quadrotors, allowing UAVs in the formation to dynamically adjust formation parameters within user-defined bounds in response to changing environmental conditions. The environment plays a crucial role in shaping the formation in environments like forest or urban areas. This presents several challenges for the team, as they may need to modify the path of the formation to navigate around obstacles, split to avoid obstacles, or shrink and rotate to pass through narrow passages. However, once the challenge is resolved, the formation has to return to its original state, ensuring robots to remain connected.

The proposed formation control method consists of several steps. During the navigation of the robots, the first step is direction finding, where each robot calculates a cost for a discrete set of angles, measuring the risk of collision with obstacles in the environment. Through communication and consensus among the robots, a direction to move is determined. This information also facilitates splitting and merging of the formation, which is a key feature of the method. If two groups of robots need to move in separate directions, a splitting decision is made, and a merging decision is executed when the groups desire to come closer to each other. Conditions of splitting and merging has been laid out in this work. The proposed direction decision method, and splitting and merging conditions are novel. The proposed direction decision method acts as an additional guarantee to avoid obstacles in a cluttered environment. With splitting and merging conditions, robots are enabled to exhibiting these behaviors based on environmental conditions.

To avoid collisions with obstacles, the robots need to determine the area where they can move freely. For this purpose, each robot calculates an obstacle-free convex region using the Iterative Regional Inflation by Semidefinite Programming (IRIS) algorithm. IRIS is a method that quickly computes large obstacle-free polytopic regions using convex optimizations. Since each robot may have different sensing capabilities and knowledge about obstacles, they exchange their calculated regions and reach a consensus on the safest area. The consensus algorithm to compute that area is given in this work. This way, robots are able to obtain the area that is as safe as possible with their total sensing capabilities and combined with the proposed direction decision method, the region is guaranteed to include the intermediate waypoint of the team.

To enable the robots to optimize their formation configuration based on parameters such as size and rotation, a formation optimization process is defined. This is a nonlinear optimization problem with both equality and inequality constraints, which is solved using Sequential Quadratic Programming (SQP). This optimization process allows the robots to adapt their formation to environmental conditions, such as shrinking or rotating in narrow corridors, translating waypoints to avoid obstacles, switching to different formation shapes, or restoring to the original formation configuration once the risk ahead is cleared. The objective function, constraints and parameters are explicitly defined in this work. This formation optimization process enable robots to exhibit a novel combination of these behaviors like rotating, shrinking and translating with user-controllable intention to execute these behaviors by adjusting weights in the objective function.

An assignment process is also executed to minimize the mission time, with the aim of minimizing the maximum path length of the robots. For quadrotors, hovering, which means staying still in the air, is not an energy-efficient state. Therefore, optimal mission time often results in optimal energy consumption in such systems. This problem is solved using the framework of strong spanning trees. In contradiction to usual approach to this problem in the literature, which is minimizing

the total path of the team, a superior performance for UAVs are obtained in this thesis.

The proposed method is verified with two simulation scenarios. The environment in the first scenario is similar to a dense forest while the second one being similar to an urban area with prismatic obstacles of different sizes. In both scenarios, it is observed that the proposed method successfully navigated the team in cluttered environments where they confront features abruptly. The emphasized means of manipulating the formation shape, such as navigating around obstacles, shrinking and rotating in order to navigate through narrow corridors, splitting to go through both sides of the confronted obstacle and merging after passing it, are all observed during these simulations, which verifies the capabilities of the proposed method. However, it is important to note that our method operates as a local path planner, and it is possible to end up in deadlocks in some scenarios.

6.2 Future Work

The proposed method is a local path planner for a team of agents, which means that it does not take into account for previously travelled paths and does not track a global plan. Therefore, it is possible to end up in a deadlock where the team cannot advance through the target position. For example, if the team enters a corridor with dead end, they will not turn back to the junction and try out the other way. To overcome this, the method can either be enhanced with a global planner, which limits the applicability since it will require the knowledge of the environment map, or be equipped with a graph search algorithm.

The method is implemented in a centralized manner in the simulation environment, because of its low computational needs, it is possible to implement it in distributed manner in a real-world scenario. However, several assumptions done in simulation may not be true in a real-world scenario and make it harder to implement. For example, it is assumed that each robot knows positions of other robots at any time.

However, in a real-world scenario, especially in cases where UAVs are seperated by far, this information might have some latency; therefore, each robot might execute the algorithm with different information. Also, especially in systems with high number of agents, the step of reaching consensus on obstacle-free polytopes might require high communication bandwidths.

Although it is mentioned that the method can be implemented in a distributed manner, some computations still need to be done by each agent in the team seperately, like formation optimization and assignment. Formation optimization is the main computational expense of the proposed method; therefore, it would be highly advantageous if it can be distributed among agents where each agent needs to handle a less complex computation. Note that, while it would be advantageous in terms of computation time, communication overhead of such modification can be high.

In both simulation scenarios, we control a team of 6 UAVs. Simulations with higher number of agents should be analyzed too in order to observe more complex behaviors such as multiple consequent splitting and merging.

The proposed method should be tested in a real-world scenario in order to fully verify its applicability. In our simulations, we use ground truth positions of each UAV. In order to isolate the analysis of the performance of the proposed method, an accurate localization system for each UAV is needed. Also, simulations are done with a framework that utilizes simple flight dynamics with no external disturbances. Therefore, UAVs easily track the waypoints assigned to them. In an hardware implementation, an accurate flight controller is again needed to eliminate low-level controller errors as much as possible. Lastly, in simulations, it is assumed that each UAV communicate with all other UAVs with no communication delay. In an hardware implementation, a realiable communication scheme between UAVs is needed in order to eliminate failures resulting from communition delays.

REFERENCES

[1] Petersen, K., Nagpal, R., and Werfel, J. (2011). Termes: An autonomous robotic system for three-dimensional collective construction. *Robotics: Science and Systems VII*.

[2] Ang, K. Z., Dong, X., Liu, W., Qin, G., Lai, S., Wang, K., Wei, D., Zhang, S., Phang, S. K., Chen, X., Lao, M., Yang, Z., Jia, D., Lin, F., Xie, L., and Chen, B. M. (2018). High-Precision Multi-UAV Teaming for the First Outdoor Night Show in Singapore. *Unmanned Systems*, 06(01), 39–65.

[3] Bajec, I. L., and Heppner, F. H. (2009). Organized flight in birds. *Animal Behaviour*, 78(4), 777–789.

[4] Chen Wang, Guangming Xie, and Ming Cao. (2013). Forming circle formations of anonymous mobile agents with order preservation. *IEEE Transactions on Automatic Control*, 58(12), 3248–3254.

[5] Song, Y., and O'Kane, J. M. (2014). Decentralized formation of arbitrary multi-robot lattices. Proceedings. *IEEE International Conference on Robotics and Automation*, 1118–1125.

[6] Deits, R., and Tedrake, R. (2015). Computing large convex regions of obstacle-free space through semidefinite programming. *Springer Tracts in Advanced Robotics*, 109–124.

[7] Shah, S., Dey, D., Lovett, C., and Kapoor, A. (2017). AirSim: High-fidelity visual and physical simulation for Autonomous Vehicles. *Field and Service Robotics*, 621–635.

[8] Turpin, M., Michael, N., and Kumar, V. (2012). Decentralized formation control with variable shapes for aerial robots. *2012 IEEE International Conference on Robotics and Automation*.

[9] Li, G., St-Onge, D., Pinciroli, C., Gasparri, A., Garone, E., and Beltrame, G. (2018). Decentralized progressive shape formation with robot swarms. *Autonomous Robots*, 43(6), 1505–1521.

[10] Bonnans, J. F., Gilbert, J. C., Lemaréchal, C., and Sagastizabal, C. (2006). Numerical Optimization: Theoretical and Practical Aspects.

[11] Burkard, R. E., and Çela, E. (1999). Linear assignment problems and extensions. *Handbook of Combinatorial Optimization*, 75–149.

[12] Armstrong, R. D., and Zhiying Jin. (1992). Solving linear bottleneck assignment problems via strong spanning trees. *Operations Research Letters*, 12(3), 179–180.

[13] Esin, Y. H., and Ünel, M. (2010). Formation control of nonholonomic mobile robots using implicit polynomials and elliptic Fourier descriptors. *Turkish Journal of Electrical Engineering and Computer Sciences*.

[14] Xiao, H., Li, Z., and Philip Chen, C. L. (2016). Formation control of Leader–Follower Mobile Robots' systems using model predictive control based on neural-dynamic optimization. *IEEE Transactions on Industrial Electronics*, 63(9), 5752–5762.

[15] Krishna S. Raghuwaiya, Shonal Singh, and Jito Vanualailai. (2011). Formation Control of Mobile Robots.

[16] Do, H., Hua, H., Nguyen, M., Nguyen, C., Nguyen, H., Nguyen, H., and Nguyen, N. (2021). Formation control algorithms for multiple-UAVs: A comprehensive survey. *EAI Endorsed Transactions on Industrial Networks and Intelligent Systems*, 8(27).

[17] Desai, J. P., Ostrowski, J., and Kumar, V. (1998). Controlling formations of multiple Mobile Robots. *Proceedings of 1998 IEEE International Conference on Robotics and Automation*.

[18] Xiaohai Li, Jizong Xiao, and Zijun Cai. (2005). Backstepping based multiple Mobile Robots Formation Control. *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*.

[19] Lewis, M. A., and Tan, K.-H. (1997). High Precision Formation Control of Mobile Robots Using Virtual Structures. In *Autonomous Robots (Vol. 4)*. Kluwer Academic Publishers.

[20] Li, N. H., and Liu, H. H. (2008). Formation UAV flight control using virtual structure and motion synchronization. *2008 American Control Conference*.

[21] Antonelli, G., Arrichiello, F., and Chiaverini, S. (2008). Flocking for multi-robot systems via the null-space-based behavioral control. *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1409–1414.

[22] Park, C.-S., Tahk, M.-J., and Bang, H. (2003). Multiple Aerial Vehicle Formation using swarm intelligence. *AIAA Guidance, Navigation, and Control Conference and Exhibit*.

[23] Wang, J., Wu, X., and Xu, Z. (2008). Potential-based obstacle avoidance in formation control. *Journal of Control Theory and Applications*, 6(3), 311–316.

[24] Paul, T., Krogstad, T. R., and Gravdahl, J. T. (2008). Modelling of UAV formation flight using 3D potential field. *Simulation Modelling Practice and Theory*, 16(9), 1453–1462.

[25] Listmann, K. D., Masalawala, M. V., and Adamy, J. (2009). Consensus for formation control of nonholonomic Mobile Robots. *2009 IEEE International Conference on Robotics and Automation*.

[26] Zhou, Y., Dong, X., Lu, G., and Zhong, Y. (2014). Time-varying formation control for unmanned aerial vehicles with switching interaction topologies. *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*.

[27] Burkard, R., Dell'Amico, M., and Martello, S. (2009). *Assignment Problems*.

[28] Ramshaw, L., and Tarjan, R.E. (2012). On Minimum-Cost Assignments in Unbalanced Bipartite Graphs.

[29] Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1–2), 83–97.

[30] Dantzig, G. (1963). *Linear Programming and Extensions*.

[31] Srinivasan, V., and Thompson, G. L. (1977). Cost operator algorithms for the Transportation Problem. *Mathematical Programming*, 12(1), 372–391.

[32] Liu, L., and Shell, D. A. (2014). Multi-robot formation morphing through a graph matching problem. *Springer Tracts in Advanced Robotics*, 291–306.

[33] Alonso-Mora, J., Breitenmoser, A., Rufli, M., Siegwart, R., and Beardsley, P. (2011). Multi-robot system for Artistic Pattern Formation. *2011 IEEE International Conference on Robotics and Automation*.

[34] Garfinkel, R. S. (1971). Technical note—an improved algorithm for the bottleneck assignment problem. *Operations Research*, 19(7), 1747–1751.

[35] Derigs, U., and Zimmermann, U. (1978). An augmenting path method for solving linear bottleneck assignment problems. *Computing*, 19(4), 285–295.

[36] Gabow, H. N., and Tarjan, R. E. (1988). Algorithms for two bottleneck optimization problems. *Journal of Algorithms*, 9(3), 411–417.

[37] Fischer, P. (1993). Finding maximum convex polygons. *Fundamentals of Computation Theory*, 234–243.

[38] Savin, S. (2017). An algorithm for generating convex obstacle-free regions based on stereographic projection. *2017 International Siberian Conference on Control and Communications (SIBCON)*.

[39] Zhong, X., Wu, Y., Wang, D., Wang, Q., Xu, C., and Gao, F. (2020). Generating Large Convex Polytopes Directly on Point Clouds.

[40] Goldstein, A. A. (1964). Convex programming in Hilbert space. *Bulletin of the American Mathematical Society*, 70(5), 709–710.

[41] Fletcher, R. (1973). An exact penalty function for nonlinear programming with inequalities. *Mathematical Programming*, 5(1), 129–150.

[42] Pantoja, J. F., and Mayne, D. Q. (1991). Exact penalty function algorithm with simple updating of the penalty parameter. *Journal of Optimization Theory and Applications*, 69(3), 441–467.

[43] Schittkowski, K. (1986). NLPQL: A Fortran subroutine solving constrained nonlinear programming problems. *Annals of Operations Research*, 5(2), 485–500.

[44] Nocedal, J., and Wright, S. J. (2006). *Numerical optimization*. Springer.

[45] MathWorks, (2022). Constrained Nonlinear Optimization Algorithms (R2022b). Retrieved January 15, 2022 from www.mathworks.com/help/optim/ug/constrained-nonlinear-optimization-algorithms.html

[46] Pitcher, T. J. (2001). Fish schooling. *Encyclopedia of Ocean Sciences*, 975–987.

[47] No, T. S., Kim, Y., Tahk, M.-J., and Jeon, G.-E. (2011). Cascade-type guidance law design for multiple-UAV formation keeping. *Aerospace Science and Technology*, 15(6), 431–439.

[48] Alonso-Mora, J., Montijano, E., Schwager, M., and Rus, D. (2016). Distributed multi-robot formation control among obstacles: A geometric and Optimization Approach with consensus. *2016 IEEE International Conference on Robotics and Automation (ICRA)*.

[49] Zhu, H., Juhl, J., Ferranti, L., and Alonso-Mora, J. (2019). Distributed multi-robot formation splitting and merging in Dynamic Environments. *2019 International Conference on Robotics and Automation (ICRA).*

[50] Novoth, S., Zhang, Q., Ji, K., and Yu, D. (2020). Distributed formation control for multi-vehicle systems with splitting and merging capability. *IEEE Control Systems Letters*.

[51] Qi, J., Guo, J., Wang, M., Wu, C., and Ma, Z. (2022). Formation tracking and obstacle avoidance for multiple quadrotors with static and dynamic obstacles. *IEEE Robotics and Automation Letters, 7(2).*