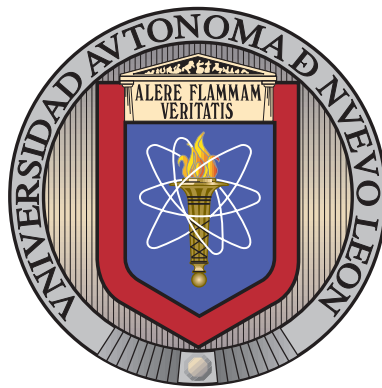


UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

SUBDIRECCIÓN DE ESTUDIOS DE POSGRADO



DESIGN OF INTELLIGENT STRATEGIES FOR THE
FORMATION FLIGHT OF MULTIPLE UAVS BASED
ON GENETIC ALGORITHMS

BY

MANUEL ALEJANDRO GUTIERREZ MARTINEZ

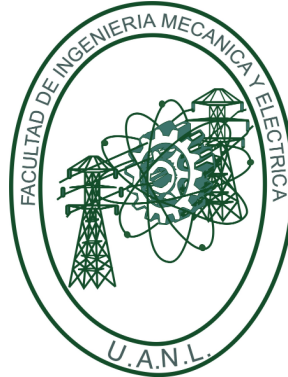
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF SCIENCE IN AERONAUTICAL ENGINEERING

FEBRUARY 2026

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

SUBDIRECCIÓN DE ESTUDIOS DE POSGRADO



DESIGN OF INTELLIGENT STRATEGIES FOR THE
FORMATION FLIGHT OF MULTIPLE UAVS BASED
ON GENETIC ALGORITHMS

BY

MANUEL ALEJANDRO GUTIERREZ MARTINEZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF SCIENCE IN AERONAUTICAL ENGINEERING

FEBRUARY 2026

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN
Facultad de Ingeniería Mecánica y Eléctrica
Posgrado

Los miembros del Comité de Evaluación de Tesis recomendamos que la tesis "Design of intelligent strategies for the formation flight of multiple UAVs based on genetic algorithms", realizada por el estudiante Manuel Alejandro Gutiérrez Martínez, con número de matrícula 1517068, sea aceptada para su defensa como requisito parcial para obtener el grado de Doctorado en Ciencias de la Ingeniería Aeronáutica.

El Comité de Evaluación de Tesis

Dr. Octavio García Salazar
Director

Dr. Edmundo Javier Ollervides Vázquez
Co-director

Dr. Luis Antonio Amézquita Brooks
Revisor

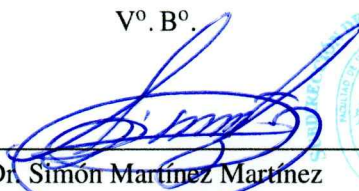
Dr. Efraín Alcorta García
Revisor

Dr. Anand Eleazar Sánchez Orta
Revisor

Dr. Ricardo Chapa García
Revisor

Dr. Efraín Ibarra Jiménez
Revisor

Vº. Bº.


Dr. Simón Martínez Martínez
Subdirector de Estudios de Posgrado

Institución 190001

Programa 557603

Acta núm. 369

Ciudad Universitaria, a 15 de enero de 2026.

*Fairy tales are more than true: not because they tell us that dragons exist, but
because they tell us that dragons can be beaten -G.K. Chesterton.*

CONTENTS

Acknowledgements	xiv
Abstract	xvi
1 Introduction	1
1.1 State of art	1
1.2 Justification	4
1.3 Hypothesis	5
1.4 Objectives	5
1.4.1 General objective	5
1.4.2 Specifics objectives	5
1.5 Methodology	6
1.6 Contributions	7
2 Genetic Algorithm Fundamentals	10
2.1 Biological Inspiration	10
2.2 Genetic algorithm	12

2.3	Parts of the genetic algorithm	15
2.3.1	Objective function	15
2.3.2	Population	16
2.3.3	Encoding and decoding	19
2.3.4	Genetics operators	20
2.4	Consensus theory	24
2.4.1	Second-order consensus	28
2.4.2	Formation control second-order protocol	29
3	Genetic algorithm for maze environment	31
3.1	Objective function	34
3.2	Genetic operators for maze solving	36
3.2.1	Selection-Crossover-Mutation	36
4	Genetic algorithm for path planning and flight formation.	40
4.1	Genetic Algorithm Initialization	40
4.1.1	Objective functions	42
4.1.2	length of path	43
4.1.3	Distance between waypoint and obstacles	44
4.1.4	Sum of angles of path	46
4.2	Genetic operators	48
4.2.1	Selection operator	48

4.2.2	Crossover operator	49
4.2.3	Mutation operator	49
4.3	Criteria for dynamic environment	52
4.4	Simulation and experimental platform	54
4.4.1	A case study: Crazyflie quadrotor	55
4.4.2	Experimental platform	56
5	Formation flight based on Consensus	57
5.1	Three UAVs in flight	58
5.2	Five UAVs in flight	60
5.3	Consensus with tracking trajectory	64
6	Results and discussion	66
6.1	Validation of Genetic Algorithm for Maze Environment	66
6.1.1	Sequence generation for maze case 1 using genetic algorithm .	67
6.1.2	Sequence generation for maze case 2 using genetic algorithm .	68
6.1.3	Sequence generation for maze case 3 using genetic algorithm .	69
6.1.4	Sequence generation for maze case 4 using genetic algorithm .	69
6.1.5	Sequence generation for maze case 5 using genetic algorithm .	70
6.1.6	Sequence generation for maze case 6 using genetic algorithm .	71
6.1.7	Fitness values for maze environment	72
6.2	Trajectory Simulation for the Maze Environment	73

6.2.1	Simulation of paths	74
6.2.2	Real-Time Maze Experiments	75
6.3	Algorithm characterization in static environment	78
6.4	Algorithm Evaluation in Dynamic Environment.	80
6.4.1	Simulation of path planning Case 1	80
6.4.2	Simulation of path planning case 2	83
6.4.3	Simulation of path planning case 3	84
6.4.4	Simulation of path planning case 2	86
6.5	Experimental results	87
6.5.1	Experimental path planning case 1	88
6.5.2	Experimental path planning case 2	89
6.5.3	Experimental path planning case 3	90
6.5.4	Experimental path planning case 4	94
6.6	Results of consensus	96
6.6.1	Formation Results with Three UAVs	96
6.6.2	Formation Results with Five UAVs	99
6.6.3	Formation results with five UAVs line-Shape	102
6.6.4	Results consensus 3 UAVs with tracking trajectory and obstacle avoidance	105
6.6.5	Results consensus 5 UAVs with tracking trajectory and obstacle avoidance	109

7	Conclusions	113
A	Equations of motion of a quadrotor	115

LIST OF FIGURES

2.1	Example of evolutionary processes using giraffes.	13
2.2	Analogy between nature and evolutionary computation.	14
2.3	Binary representation of chromosomes and corresponding phenotypes.	18
2.4	Selection methods in genetic algorithms.	22
2.5	Types of crossover operators in genetic algorithms.	24
2.6	Mutation operator.	24
2.7	Flock [1].	26
2.8	Connection graph.	27
3.1	Schematic representation of movement sequences.	32
3.2	Change of direction by the intersection point.	34
3.3	Change of direction by decision point.	34
3.4	Double crossover point.	36
4.1	Path length evaluation.	43
4.2	Evaluation waypoint.	44

4.3	Evaluation planes 2D.	45
4.4	Evaluation of path crossing by obstacles.	46
4.5	Crossover genetic operator.	49
4.6	Block Diagram of the UAV Guidance System Using Genetic Algorithms.	56
5.1	Graph of Connections for 3 vehicles.	58
5.2	Graph of Connections for 5 vehicles.	61
5.3	Formation with 5 vehicles for delta.	62
5.4	Formation with 5 vehicles for line.	63
6.1	Path generated by genetic algorithm for case 1.	67
6.2	Path generated by genetic algorithm for Case 2.	68
6.3	Path generated by genetic algorithm for case 3.	70
6.4	Path generated by genetic algorithm for case 4.	71
6.5	Path generated by genetic algorithm for case 5.	72
6.6	Path generated by genetic algorithm for case 6.	73
6.7	Fitness value graphs for cases 1 to 6.	74
6.8	Trajectories in the XY -axis for different cases.	75
6.9	Quadrotor used for experimental tests.	76
6.10	Trajectory in the XY -axis for first experimental scenario.	77
6.11	Trajectory in the XY -axis for second experimental scenario.	77
6.12	Fitness value for GaCPo, GaHM, GaLM.	79

6.13 Iterations of convergence for GaCPo,GaHM,GaLM.	80
6.14 Avarege for GaCPo,GaHM,GaLM.	81
6.15 Simulation: 3D path static obstacles case 1 view 1.	82
6.16 Simulation: 3D path static obstacles case 1 view 2.	82
6.17 Simulation: single axis X case 1.	82
6.18 Simulation: single axis Y case 1.	82
6.19 Simulation: single axis Z case 1.	83
6.20 Simulation: Fitness value static case 1.	83
6.21 Simulations: 3D path dynamics obstacles case 2 view 1.	84
6.22 Simulations: 3D path dynamics obstacles case 2 view 2.	84
6.23 Simulation: single axis X case 2.	84
6.24 Simulation: single axis Y case 2.	84
6.25 Simulation: single axis Z case 2.	85
6.26 Simulation: fitness value dynamic case 2.	85
6.27 Simulation: 3D path dynamics obstacles case 3 view 1.	86
6.28 Simulation: 3D path dynamics obstacles case 3 view 2.	86
6.29 Simulation: single axis X case 3.	87
6.30 Simulation: single axis Y case 3.	87
6.31 Simulation: single axis Z case 3.	87
6.32 Simulation: fitness value dynamic case 3	87

6.33 Simulation: 3D path dynamics obstacles case 4 view 1.	88
6.34 Simulation: 3D path dynamics obstacles case 4 view 1.	88
6.35 Simulation: single axis X case 4.	88
6.36 Simulation: single axis Y case 4.	88
6.37 Simulation: single axis Z case 4.	89
6.38 Simulation: fitness value dynamic case 4.	89
6.39 Experiment: 3D path dynamic obstacles case 1 view 1.	90
6.40 Experiment: 3D path static obstacles case 1 view 2.	90
6.41 Experiment: single axis X case 1.	91
6.42 Experiment: single axis Y case 1.	91
6.43 Experiment: single axis Z case 1.	91
6.44 Experiment: fitness value dynamic case 1.	91
6.45 Experiment: 3D path dynamics obstacles case 2 view 1.	92
6.46 Experiment: 3D path dynamics obstacles case 2 view 2.	92
6.47 Experiment: single axis X components case 2.	93
6.48 Experiment: single axis Y components case 2.	93
6.49 Experiment: single axis Z case 2.	93
6.50 Experiment: fitness value dynamic case 2.	93
6.51 Experiment: 3D path dynamics obstacles case 3 view 1.	93
6.52 Experiment: 3D path dynamics obstacles case 3 view 2.	93

6.53 Experiment: single axis X case 3.	94
6.54 Experiment: single axis Y case 3.	94
6.55 Experiment: single axis Z case 3.	94
6.56 Experiment: Fitness value dynamic case 3.	94
6.57 Experiment: 3D path dynamics obstacles case 4 view 1.	95
6.58 Experiment: 3D path dynamics obstacles case 4 view 2.	95
6.59 Experiment: single axis X case 4.	96
6.60 Experiment: single axis Y case 4.	96
6.61 Experiment: single axis Z case 4.	96
6.62 Experiment: fitness value dynamic case 4.	96
6.63 Axis of movement for 3 UAVs: a) axis of movement X, b) axis of movement Y, c) axis of movement Z.	97
6.64 Distance between 3 UAVs for delta formation.	98
6.65 Distance between 3 UAVs for delta formation axis Z	98
6.66 3D Trajectory View of 3 UAVs.	99
6.67 Axis of movement for 5 UAVs delta formation: a) axis of movement X, b) axis of movement Y, c) axis of movement Z.	100
6.68 Distance between 5 UAVs for delta formation.	101
6.69 Distance between 5 UAVs for delta formation along the Z-axis.	101
6.70 3D Trajectory View of 5 UAVs delta formation.	102

6.71	Axis of movement for 5 UAVs line formation: a) axis of movement X, b) axis of movement Y, c) axis of movement Z.	103
6.72	Distance between 5 UAVs for line formation.	104
6.73	Distance between 5 UAVs for line formation along Z-axis.	104
6.74	3D Trajectory View of 5 UAVs line formation.	105
6.75	Axis of movement for 3 UAVs formation and genetic path: a) axis of movement X, b) axis of movement Y, c) axis of movement Z.	106
6.76	Distance between 3 UAVs for delta formation with tracking trajectory.	107
6.77	Distance between 3 UAVs delta for formation along Z-axis with track- ing trajectory.	107
6.78	Distance between 3 UAVs for delta formation along Z-axis with track- ing trajectory.	108
6.79	Fitness values for delta formation 3 UAVs for genetic path.	108
6.80	Axis of movement for 5 UAVs formation and genetic path: a) axis of movement X, b) axis of movement Y, c) axis of movement Z.	109
6.81	Distance between 5 UAVs for delta formation with tracking trajectory.	110
6.82	Distance between 5 UAVs for delta formation along Z-axis with track- ing trajectory	111
6.83	3D Trajectory view of 5 UAVs for delta formation with tracking tra- jectory	112
6.84	Fitness values for delta formation 5 UAVs for genetic path	112

LIST OF TABLES

5.1	Initial positions of 5 vehicles	59
5.2	Initial positions of 5 vehicles	61
5.3	Initial positions of 5 vehicles	64
5.4	Initial positions of 5 vehicles	65
6.1	Quadrotor parameters.	73
6.2	Case static environment.	79

ACKNOWLEDGEMENTS

I would like to thank the members of my Thesis Committee for their guidance, support, and patience throughout this process.

Similarly, I express my deepest gratitude to my professors, whose teachings and guidance have profoundly shaped my academic journey. I extend my appreciation to the Universidad Autónoma de Nuevo León (UANL), the Faculty of Mechanical and Electrical Engineering (FIME), and the Center for Research and Innovation in Aeronautical Engineering (CIHA) for providing the facilities and an enriching environment that have been essential for my academic and professional development.

I also wish to express my gratitude to CONACYT for granting me a scholarship No. 794679, which was crucial in supporting my research and studies. Without this financial assistance, the completion of this thesis would not have been possible.

This research work is funded by the U.S. Office of Naval Research Global under Grant N629092512014.

I express my deepest gratitude to my Thesis Advisor, Dr. Octavio García Salazar, for his unwavering support, insightful guidance, and valuable feedback throughout the development of this research. His expertise and encouragement were fundamental to the successful completion of this work. I extend my sincere thanks to my Co-Advisor, Dr. Edmundo Javier Ollervides Vázquez, whose advice, experience, and critical perspective enriched the quality of this project. I also wish to thank the distinguished members of my Thesis Committee: Dr. Luis Antonio

Amézquita Brook, Committee Chair, for his leadership and direction; Dr. Efraín Alcorta García, Secretary, for his thoughtful reviews and observations; and Dr. Anand Eleazar Sánchez Orta, Dr. Ricardo Chapa García, and Dr. Efraín Ibarra Jiménez, Committee Members, for their rigorous evaluation, constructive feedback, and valuable contributions during the defense process.

It has been an honor to work under the guidance of such a knowledgeable and supportive committee. I am truly grateful for their time, effort, and dedication. Finally, I want to thank my family and friends for their unwavering support, encouragement, and understanding throughout this demanding but rewarding journey. To my family, thank you for always believing in me, even in moments when I doubted myself. Your love, patience, and sacrifices have been the foundation upon which I have built this achievement. To my friends, thank you for being a constant source of motivation, for thoughtful conversations, and for reminding me to take breaks and enjoy the process. Your presence, either near or far, has made this journey not only possible but also meaningful. I am deeply grateful to have shared this chapter of my life with you.

ABSTRACT

MANUEL ALEJANDRO GUTIERREZ MARTINEZ.

Candidate for the degree of Doctor of Science in Aeronautical Engineering .

Universidad Autónoma de Nuevo León.

Facultad de Ingeniería Mecánica y Eléctrica.

Title of the study: DESIGN OF INTELLIGENT STRATEGIES FOR THE FORMATION
FLIGHT OF MULTIPLE UAVS BASED ON GENETIC ALGORITHMS.

Number of pages: 124.

OBJECTIVES AND STUDY METHOD: The main objective of this research is to design an intelligent formation flight strategy for the coordination of multiple agents using genetic algorithms.

To develop a static path planning and obstacle avoidance algorithm.

To develop and implement an online dynamic path planning algorithm for path planning and obstacle avoidance.

To develop a cooperative flight strategy for vehicle coordination using genetic algorithms.

The first stage of this project will focus on the development of a static trajectory planning algorithm. In this stage, the aim is to design a method that allows

the calculation of predefined optimal routes, ensuring that the planned path minimizes associated costs, such as the distance traveled. This algorithm will take into account environmental constraints and the initial conditions of the system, establishing a movement plan that guarantees efficiency and safety in the execution of the desired path.

The second stage will focus on the development of an algorithm for dynamic online path planning. In this phase, a method will be designed to calculate optimal routes in real-time, adapting to changes and constraints in the environment as the system is in motion. The goal is to create an algorithm that efficiently responds to unexpected variations, ensuring a trajectory planning process that is flexible, safe, and effective during execution.

The third stage will focus on testing vehicle coordination strategies, applying both a strategy that incorporates a conventional consensus algorithm and a strategy that uses a genetic algorithm for vehicle coordination. The objective of this stage is to evaluate the effectiveness and efficiency of both methodologies in coordinating multiple vehicles, identifying the strategy that offers better performance in terms of adaptability and response to dynamic environmental conditions.

CONTRIBUTIONS AND CONCLUSIONS: In this work, a significant contribution was made to the development of new strategies for complex tasks such as trajectory planning and vehicle coordination. The proposed strategies were based on the use of evolutionary algorithms, with genetic algorithms playing a central role. Although genetic algorithms are traditionally used as offline strategies due to their high computational requirements and the need for multiple iterations to reach optimal solutions, this project implemented innovative modifications to the genetic operators. These modifications allowed for an increased exploration capacity of the solution space and improved the speed of convergence toward optimal results. Thanks to these advancements, it was possible to adapt and apply genetic algorithms in an online context, achieving an efficient and effective implementation that responds to

dynamic real-time environmental demands.

It was concluded that evolutionary algorithms can be effectively implemented in an online environment, overcoming the traditional limitations that confined them to offline applications due to their computational demands. Thanks to the modifications made to the genetic operators, it was possible not only to accelerate the convergence of these algorithms but also to maintain a high degree of solution exploration, allowing their application in real time. Furthermore, it was demonstrated that the vehicle coordination strategies developed in this work could be implemented efficiently, proving their ability to adapt to dynamic and changing environments. This opens up new possibilities for the implementation of coordination and planning systems in scenarios where speed and adaptability are essential for optimal performance.

Contributions:

- M. A. Gutierrez-Martinez, E. G. Rojo-Rodriguez, L. E. Cabriaes-Ramirez, L. A. Reyes-Osorio, P. Castillo, and O. Garcia-Salazar, “Collision-free path planning based on a genetic algorithm for quadrotor UAVs,” in *Proc. 2020 Int. Conf. Unmanned Aircraft Systems (ICUAS)*, 2020, pp. 948–957.
- L. E. Cabriaes-Ramírez, V. H. Dominguez, M. A. Gutierrez-Martinez, E. J. Ollervides-Vazquez, E. G. Rojo-Rodriguez, P. A. Tellez-Belkotosky, and O. Garcia-Salazar, “Trajectory tracking based on a saturated PD controller on $SE(3)$ for a micro coaxial drone,” in *Proc. 2021 Int. Conf. Mechatronics, Electronics and Automotive Engineering (ICMEAE)*, 2021, pp. 146–153.
- P. A. Tellez-Belkotosky, E. J. Ollervides-Vazquez, E. G. Rojo-Rodriguez, J. L. Santillan-Avila, L. E. Cabriaes-Ramirez, M. A. Gutierrez-Martinez, and O. Garcia-Salazar, “Nonlinear flight navigation controller for an octorotor unmanned aerial vehicle,” in *Proc. 2021 Int. Conf. Mechatronics, Electronics and Automotive Engineering (ICMEAE)*, 2021, pp. 52–59.

- M. A. Gutierrez-Martinez, L. E. Cabriaes-Ramirez, E. U. Rojo-Rodriguez, E. J. Ollervides-Vazquez, P. Castillo, and O. Garcia-Salazar, “Genetic algorithm for path planning of UAVs as a maze-solving problem,” in *Proc. 2022 Int. Conf. Unmanned Aircraft Systems (ICUAS)*, 2022, pp. 881–890.
- P. A. Tellez-Belkotosky, L. E. Cabriaes-Ramirez, M. A. Gutierrez-Martinez, and E. J. Ollervides-Vazquez, “Intelligent PIV fuzzy navigation and attitude controller for an octorotor Mini-UAV,” *Machines*, vol. 11, no. 2, p. 266, 2023.
- M. A. Gutierrez-Martinez, E. G. Rojo-Rodriguez, L. E. Cabriaes-Ramirez, K. Estabridis, and O. Garcia-Salazar, “Genetic algorithm-based path planning of quadrotor UAVs on a 3D environment,” *Aeronaut. J.*, vol. 129, no. 1334, pp. 902–938, 2025.

Advisor signature: _____

Nombre del asesor

CHAPTER 1

INTRODUCTION

1.1 STATE OF ART

Path planning is a key task in robotics and one of the main challenges in aerial vehicles, as it involves finding an efficient path from an initial point to a final point while avoiding obstacles [2],[3]. This research field has practical applications in improving the efficiency and autonomy of mobile robots and unmanned aerial vehicles, as demonstrated by Shub et al. [4], Majeed et al. [5], and Wang et al. [6], showing the adaptability of these algorithms for different platforms and environments. As an optimization problem, the objective is to improve both time and energy efficiency while ensuring avoidance of obstacles.

Various traditional methods have been explored to resolve the planning of the path, such as potential fields [7, 8], cell decomposition [9], and visibility graphs [10]. However, the complexity of path planning classifies it as an NP-hard optimization problem [11], which makes classical approaches less effective due to the high computational costs and the possibility of falling into local minima [12]. To address these limitations, bioinspired and heuristic techniques such as Particle Swarm Optimization (PSO) [13], neural networks [14], Ant Colony Optimization (ACO) [15], and Genetic Algorithms (GA) [16] have been implemented. These approaches are

effective in navigating around obstacles, minimizing distances, and avoiding being trapped in local minima [17].

Genetic Algorithms (GAs), originally proposed by J. Holland in [18], are meta-heuristic techniques inspired by natural processes such as evolution and genetics. GAs work with populations of individuals, where each individual represents a potential solution to a specific problem. Genetic operators are applied to create new individuals and eliminate less suitable ones, maintaining diversity and enabling convergence to optimal solutions. An objective function evaluates the population, guiding the selection process towards optimal paths.

Path-planning algorithms have been used in static environments where the layout is well known. For example, Lamini et al. [19] used an improved crossover operator to avoid early convergence. Similarly, Wang [20] integrated prior knowledge to avoid radar zones. However, UAVs often operate in dynamic environments. To address this, Seyed et al. [12] proposed a hybrid GA with fuzzy logic, while Elhoseny et al. [21] developed GADPP, incorporating Bezier curves and time constraints for near-optimal paths.

Several studies have explored enhanced GA variants. Roberge et al. [22] employed parallel GAs in real 3D scenarios, integrating UAV dynamics. Mohamed [23] introduced a modified GA using Bézier curves for smooth and energy-efficient paths. Shivgan [24] minimized the path length and turns for energy savings in planning the path of UAVs based on TSP. Arantes et al. [25] combined multipopulation GA with visibility graphs for fast computation in uncertain environments. Volkan et al. [26, 27] integrated GA with ACO, Voronoi diagrams, and clustering, achieving high performance and reduced evaluations. Chaoqun et al. [28] combined heuristic crossover with SAR for real-time adjustments, while Pan et al. [29] merged deep learning with GA in the DL-GA model. Chen et al. [30] designed a GA for autonomous cruise, outperforming conventional methods.

Among other techniques, various hybrid algorithms have been proposed. In

[31], an improved PSO was presented that incorporates typical characteristics of GA, such as selection, crossover, and mutation, combined with the PSO exploration capabilities. This approach enabled the generation of efficient routes in complex three-dimensional environments, without inflection points, and with shorter and smoother trajectories, reaching the optimal solution in only 23 iterations. On the other hand, [32] developed path planning algorithms for aerial vehicles with actuator failures, with the aim of enabling emergency landings by minimizing either distance or time. To achieve this, multidimensional versions of PSO and GA were implemented, adapted to real physical constraints. Finally, in [33], a bioinspired algorithm based on plant growth was proposed. It was evaluated against classical methods such as A*, RRT, and the ant colony algorithm, both in simulations and in real-world UAV tests. The results showed superior performance, achieving smoother trajectories, improved maneuverability, and no inflection points, particularly outperforming the ant colony algorithm.

Another important approach to formation flight is the reconfiguration of formation. Although some studies focus on control methods, others propose algorithms for reconfiguration. For example, in [34], a hybrid algorithm named HPSOGA, combining PSO and GA, treats reconfiguration as an optimization problem with continuous control inputs modeled as parameterized curves. This transforms linear segments into continuous trajectories, allowing optimization-based reorganization. Formation reconfiguration can therefore be framed as an optimization problem involving dynamics and constraints, where artificial intelligence techniques can be applied [35], [36].

Machine learning methods have experienced significant growth in various areas, especially in UAV systems. In controlling and parameter tuning, machine learning has been applied effectively. For example, in [37], the PID parameters were tuned using feedback neural networks (FNN). In adaptive control, [38] trained neural networks to learn the dynamics of UAVs to track the trajectory. For cooperative flight, [39] proposed the Geometric Learning Algorithm (GLA), which optimizes obstacle

avoidance and trajectory economy based on geometric and inter-vehicle information.

Other techniques of artificial intelligence, such as metaheuristics, are also used for multi-vehicle coordination. In [40], an evolutionary algorithm handled 3D robot formations with up to 10 agents, achieving stable results in 95% of the cases. In [41], a GA optimized a master-slave robot coordination system, minimizing rotations while maximizing formation time. These examples show that both machine learning and heuristic approaches are effective for coordination tasks.

1.2 JUSTIFICATION

Unmanned Aerial Vehicles (UAVs) are becoming increasingly important in both military and civilian applications, such as intelligence gathering, surveillance, environmental monitoring, search and rescue, among others. In this context, the formation flight of multiple UAVs has emerged as a key area of interest for scientists and engineers due to the advantages it offers in terms of efficiency, time savings, and higher task completion rates. However, to maximize these advantages, it is not enough for the vehicles to maintain a static formation; the formation must be dynamic and adaptable to changing environmental conditions.

Genetic algorithms, as widely used optimization methods in various fields, have found new applications in vehicle coordination. These algorithms allow for the optimization of coordination and adaptation of UAV formations, adjusting to changes in the environment. Therefore, the use of genetic algorithms for trajectory planning and coordination in intelligent formations presents an effective solution to fully exploit the capabilities of UAVs in complex missions.

1.3 HYPOTHESIS

The design of intelligent formation flight strategies, based on genetic algorithms, will enable the coordination of multiple agents for the execution of various tasks. In addition, these algorithms will be capable of generating adaptive formations, responding to environmental changes, and adjusting the geometric arrangement of the formation.

1.4 OBJECTIVES

1.4.1 GENERAL OBJECTIVE

To design of intelligent formation flight strategies for the coordination of multiple agents using genetic algorithms.

1.4.2 SPECIFICS OBJECTIVES

- To develop and implement a static path planning algorithm.
- To develop and implement the trajectory planning algorithm for a static environment.
- To develop and implement a vehicle coordination strategy based on consensus and genetic algorithms.
- Implement vehicle coordination with a genetic trajectory generator.

1.5 METHODOLOGY

The methodology of this thesis is structured in six main stages, each addressing a specific objective in the progressive development of the trajectory planning and formation control algorithms:

1. **Design of genetic algorithms for trajectory planning in static environments**

- Definition of the cost function to evaluate trajectory quality.
- Selection of genetic operators (selection, crossover, mutation).

2. **Offline implementation and validation of genetic algorithms in maze-type environments**

- Maze-type environments will be solved using the genetic algorithm as a trajectory planner.
- Navigation points (waypoints) defining the optimal path will be extracted from the solution.
- Continuous and feasible trajectories will be generated from the waypoints using smoothing techniques.
- The resulting trajectories will be implemented in the aerial vehicle model within a simulation environment to validate feasibility and performance.

3. **Development of genetic algorithms for trajectory generation with dynamic obstacle avoidance (online)**

- Adaptation of the algorithm for real-time operation.
- Modification of genetic operators to ensure rapid reaction in dynamic environments.

4. Online implementation and validation on a single UAV with dynamic obstacle avoidance

- Simulation of genetic algorithms for trajectory generation with dynamic obstacle avoidance.
- Implementation of trajectory planning algorithms on the Crazyflie UAV platform.

5. Design of formation control algorithms with obstacle avoidance

- Development of control laws to coordinate multiple UAVs in formation.
- Establishment of stability and safety criteria for formation flight.

6. Simulation of formation control with dynamic obstacle avoidance

- Simulation of the multi-UAV system in dynamic environments.
- Validation of the system's ability to maintain formation and avoid collisions.

1.6 CONTRIBUTIONS

The works derived from this thesis that were successfully published are the following:

- **“Collision-free path planning based on a genetic algorithm for quadrotor UAVs”**: A genetic algorithm was implemented for trajectory planning. Through this approach, waypoints were generated and subsequently linked using well-defined turning radii to construct the final trajectory. Although the algorithm was developed in an offline environment, this result allowed us to establish the initial configurations of our genetic algorithms, as well as to analyze the characteristics and performance of different genetic operators. As a main contribution, this study laid the groundwork for the use of evolutionary

algorithms in the generation of safe trajectories for UAVs, providing a reference framework for future implementations in online and real-time environments.

- **“Genetic algorithm for UAV path planning as a maze solving problem”**: The study in static environments was continued, deepening the use of genetic algorithms for trajectory planning and their applications. For this purpose, a maze-type environment was designed to represent complex scenarios. The algorithm was evaluated in these environments, characterized by combinatorial sequences of movements, demonstrating that genetic algorithms are efficient for trajectory planning in this type of problem. Subsequently, using high-degree polynomials, smoothed trajectories were generated, which facilitated the construction of feasible and continuous routes for UAV tracking.
- **“Genetic algorithm-based path planning of quadrotor UAVs in a 3D environment”**: The previous research was extended, expanding the scope to three-dimensional environments and dynamic scenarios. In this case, the genetic algorithm was implemented directly on a quadrotor, which enabled real-time trajectory planning with improved response times and the ability to react to moving obstacles. This study demonstrated not only the feasibility of applying genetic algorithms in online applications, but also their potential to generate adaptive and safe trajectories for UAVs operating in complex environments.
- **“Trajectory tracking based on a saturated PD controller on $SE(3)$ for a micro coaxial drone”**: In this study, the trajectory tracking problem for the micro coaxial rocket drone v3.0 (MCR UAV v3.0) was addressed. Using the Newton–Euler formulation, the equations of motion were obtained to accurately represent the vehicle dynamics. A saturated PD controller in cascade was designed on $SE(3)$, where the position dynamics were regulated through a guidance frame and the orientation dynamics were stabilized with the proposed control law. Numerical simulations, including scenarios with wind disturbances, demonstrated the robustness of the approach. The satu-

rated PD strategy effectively avoided overshoots while ensuring convergence to the desired trajectory, highlighting its suitability for UAV guidance and control under realistic conditions.

- **“Nonlinear flight navigation controller for an octorotor unmanned aerial vehicle”**: In this work, a nonlinear feedback controller was developed for an octorotor UAV. The proposed control strategy combined a Nonlinear Proportional Velocity (NLPV) term with a Linear Integral (LI) term, structured as a two-layer bounded controller with saturation. Using the Newton–Euler formulation, the equations of motion were derived, capturing both translational and attitude dynamics through interconnected sets of equations. The inner control layer, based on NLPV plus LI, was applied to the attitude dynamics, generating reference signals for the outer layer, which regulated the translational dynamics. Numerical simulations validated the robustness and effectiveness of the approach, showing that the proposed controller ensured accurate navigation and trajectory tracking under nonlinear flight conditions.
- **“Intelligent PIV fuzzy navigation and attitude controller for an octorotor Mini-UAV”**: This work presents the design of a proportional integral velocity (PIV) fuzzy gain scheduling controller for an octorotor mini-UAV. The scheme integrates the classical PIV approach with fuzzy gain scheduling in a two-layer cascade with saturation. Using the Newton–Euler formulation, the equations of motion were derived to model the coupling between translational and attitude dynamics. The inner layer applied the fuzzy PIV controller to attitude dynamics, providing references for the outer layer regulating translational motion. Saturation mechanisms bounded translational forces and rotor thrust, avoiding excessive deviations in Euler angles and angular speeds. Simulations validated the approach, demonstrating robust trajectory tracking across all axes and confirming the effectiveness of the fuzzy PIV controller.

CHAPTER 2

GENETIC ALGORITHM FUNDAMENTALS

In this chapter, the basic foundations of how biological processes in nature work will be addressed, with special emphasis on the mechanisms of natural selection, adaptation, and the evolution of species. It will explain how these principles have inspired the development of computational algorithms known as evolutionary algorithms, among which genetic algorithms stand out.

In addition, the general operation of a genetic algorithm will be described, including key concepts such as population, selection, crossover, mutation, and evaluation through fitness functions. Their main advantages, applications, and limitations will be analyzed in order to understand how they can be used to solve complex optimization problems in engineering and other fields of knowledge.

2.1 BIOLOGICAL INSPIRATION

Genetics is a field of biology that focuses on the study of how traits are inherited and vary among organisms. It analyzes how characteristics are passed down from one generation to the next and how they are expressed and evolve over time. The

principles of Genetic Algorithms are directly inspired by natural inheritance, as well as their origin and development, and are grounded in natural evolutionary processes. The main terminologies related to the biological context of species are presented in the following.

The cell is the fundamental unit of life and represents the smallest structure capable of performing all the essential functions necessary for the existence of an organism. Within cells, genetic information is stored, which is typically found in the form of deoxyribonucleic acid (DNA). Chromosomes are structures composed of deoxyribonucleic acid that store genetic information. In humans, chromosomes are organized in pairs, with a total of 23 pairs. Each chromosome is divided into smaller units called the gene, which encodes the traits of a species and determines the characteristics of an individual. Variations in a gene are known as alleles, and a gene can have multiple alleles. For example, the gene that determines the color of the eyes may have alleles such as black, brown, blue, or green. The collection of alleles within a specific population is referred to as the gene pool. In an individual, the complete combination of genes is known as a genotype. The phenotype, on the other hand, refers to the physical appearance resulting from the expression of the genotype. An important aspect of evolution is that selection acts on the phenotype, while reproduction recombines the genotype.

The reproduction process aims to produce offspring that improve the characteristics of previous generations by adapting to new environmental conditions, thereby ensuring the survival of the species. In genetics, reproduction occurs through mitosis or meiosis. During mitosis, genetic information is duplicated identically to generate new daughter cells, without information exchange taking place. This process is essential for the growth of multicellular structures, such as organs. Meiosis serves as the foundation for sexual reproduction. During meiotic division, two gametes are generated. In the reproductive process, these gametes merge to form a zygote, which then develops into a new individual. Natural selection is a process in which environmental conditions exert pressure on populations of organisms, promoting traits

that enhance adaptation and reproductive capacity. This mechanism allows species to gradually adapt to their environment, contributing to biological evolution and diversity.

Figure 2.1 shows a classic example that illustrates how evolutionary processes work in nature, using giraffes as a case study. The initial population consists of giraffes of various heights: tall, medium, and short. Through the process of natural selection, giraffes that are better adapted to their environment have a higher chance of obtaining food. In this case, taller giraffes can reach the treetops more easily, giving them a survival advantage.

These better adapted giraffes are also more likely to reproduce and pass their genes on to the next generation. In nature, spontaneous genetic changes known as mutations can occur, resulting in new traits such as sudden changes in height or fur color. Although many mutations do not necessarily improve individual adaptation, they contribute to maintaining genetic diversity within the population, which is essential to prevent evolutionary stagnation. After many generations (or iterations), the best adapted giraffes prevail, demonstrating how evolution favors the organisms that are the most fit for survival.

2.2 GENETIC ALGORITHM

Genetic algorithms are classified as adaptive heuristic optimization methods and are inspired by natural processes such as sexual reproduction and the survival principle of species. In this approach, individuals who are better adapted to their natural environment have a higher likelihood of passing on their genetic information to future generations. The original idea of genetic algorithms was introduced by Holland in 1975 [18], and Goldberg later provided a detailed explanation in 1989 [42].

Genetic algorithms, as an evolutionary method, allow individuals to adapt and evolve through an iterative process aimed at finding increasingly optimal solutions. As a

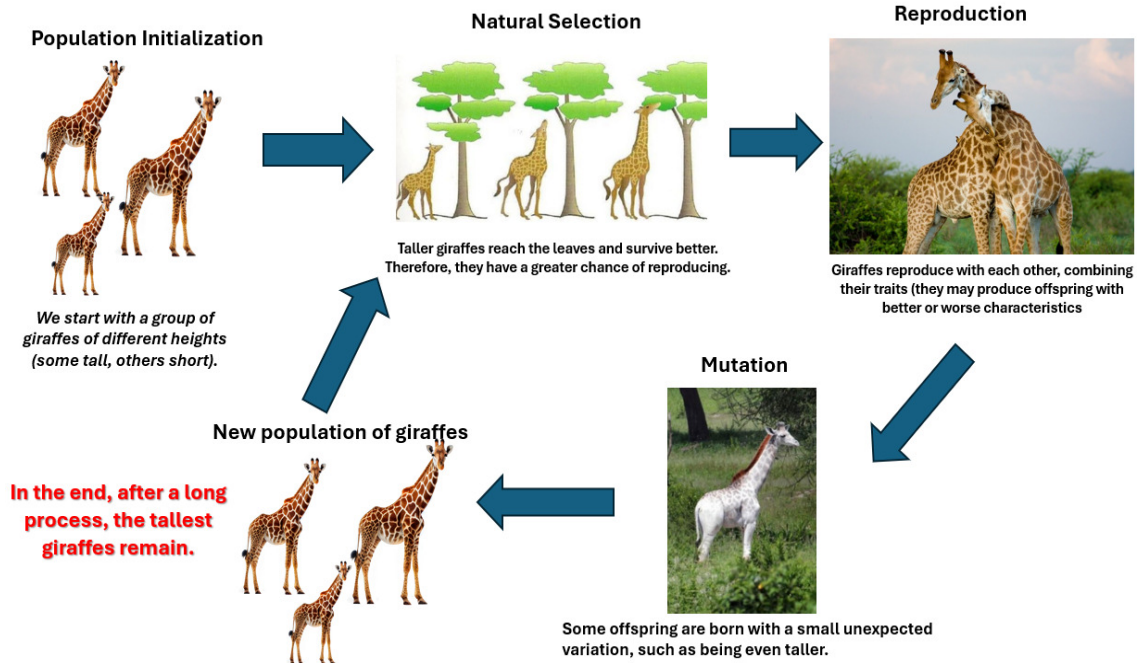


Figure 2.1: Example of evolutionary processes using giraffes.

heuristic approach, they have the advantage of not relying on complex analytical methods but rather on computational power and the efficient handling of large populations. These algorithms operate with populations of individuals that are generally generated randomly and whose characteristics can be represented in binary format, although decimal representations are also used. One of their main strengths is the ability to generate new individuals within the population, which facilitates the exploration of a wider and more diverse search space. The representation and encoding of the problem within the algorithm are fundamental aspects. In this context, individuals represent possible solutions and are described by a set of parameters known as genes. Each gene is typically composed of alleles, and a group of genes is organized into a bit string, commonly referred to as a chromosome. The conversion from the chromosome to its real-world application is known as the phenotype.

Figure 2.2 shows an analogy between the genetic algorithm and natural processes. At first, there is an environment with individuals represented by gray circles. In computer terms, this is similar to a problem with many possible solutions, where

better-adapted individuals have a higher chance of surviving and creating new ones. This idea becomes a selection rule based on the constraints of the problem, where the best individuals shown as blue circles are more likely to create offspring, whereas the weaker ones shown as green circles are removed.

This process, similar to sexual reproduction in nature, is called crossover in genetic algorithms. It combines bit strings from different individuals to create new solutions and better explore the search space. Sometimes, a mutation occurs, which adds variety to the population. In genetic algorithms, mutation helps avoid getting stuck in local solutions and makes it easier to find the best answer. After repeating this process many times, a new individual appears that represents the best features of previous generations. In genetic algorithms, this means finding a solution that meets the problem conditions in an efficient way.

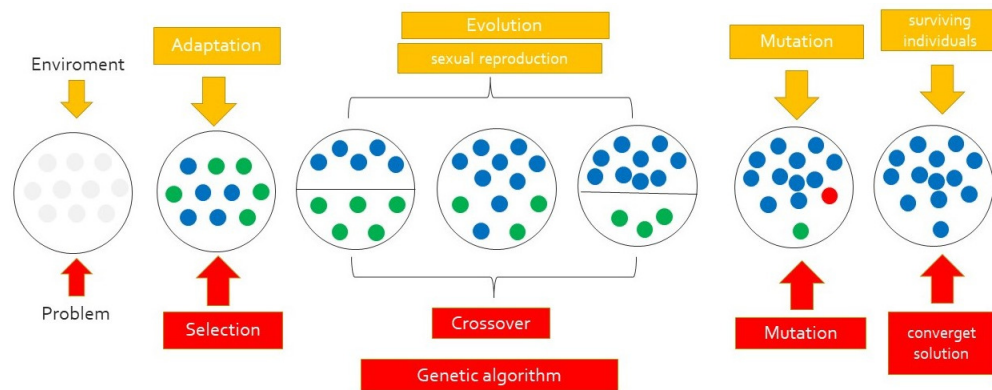


Figure 2.2: Analogy between nature and evolutionary computation.

2.3 PARTS OF THE GENETIC ALGORITHM

2.3.1 OBJECTIVE FUNCTION

Genetic algorithms make use of objective functions that evaluate individuals within the population. This function assigns a value to each individual, allowing the evaluation of their effectiveness in solving the given problem. The objective function serves as the central element of the algorithm and is applicable to both the continuous and discrete cases. Heuristic methods, such as genetic algorithms, are typically employed in problems where objective functions are nondifferentiable, which poses challenges for conventional optimizers. Compared to nature, the objective function acts similarly to a natural adaptation criterion. In this context, individuals who demonstrate better adaptation to their environment have a greater likelihood of reproducing and passing on their genetic information. Some examples of objective functions are as follows.

Direct fitness value: In this interpretation of the objective function, the fitness value is used as originally defined in the context of the problem, without applying any standardization or normalization.

Standardized fitness value: This approach is applied to both maximization and minimization problems. In the minimization cases, the direct fitness value is used, while for maximization, it is adjusted by subtracting it from an upper limit value, as shown in Equation (2.1).

$$fvalue_s = \begin{cases} fvalue_p, & \text{minimization} \\ fvalue_{max} - fvalue_p, & \text{maximization} \end{cases} \quad (2.1)$$

Where $fvalue_p$ denotes the objective function value for an individual p in the population, which directly depends on the quality of the solution in the trajectory

planning problem. The term $fvalue_{max}$ corresponds to the maximum objective function value achieved by the population in the current generation, and it is used as a reference in maximization problems. Finally, $fvalue_s$ represents the standardized fitness value, which allows individuals to be normalized and fairly compared, regardless of whether the problem involves minimization or maximization.

Thus, minimization problems, the fitness value is taken directly as $fvalue_p$, while in maximization problems, the transformation $fvalue_{max} - fvalue_p$ is applied. This ensures a unified selection criterion within the genetic algorithm.

Adapted fitness value: This value is obtained by applying a transformation to the standardized fitness value as shown in Equation (2.2). In this way, the adapted fitness value always falls within the range $[0, 1]$. A value closer to 1 indicates better performance of the individual.

$$fvalue_a = \frac{1}{1 + fvalue_s} \quad (2.2)$$

Where $fvalue_a$ denotes the adapted fitness value, obtained by applying a transformation to the standardized fitness value $fvalue_s$. This transformation bounds the fitness values within the range $[0, 1]$, where values closer to 1 indicate better individuals and values closer to 0 represent poorer solutions.

Normalized fitness value: The normalized fitness value adds an additional component, as it reflects the quality of a solution compared to other solutions within the population. Similarly to the adapted fitness value, it remains within the range $[0, 1]$, with the individuals being more efficient the closer they are to 1.

2.3.2 POPULATION

In a genetic algorithm, the population consists of a set of individuals or candidate solutions that evolve over multiple generations. Each individual in the population

represents a potential solution to the problem being addressed. The key factors related to the population in genetic algorithms include the size of the population, the number of chromosomes present, and the number of genes contained in each chromosome.

Chromosomes: Each individual within the population has a unique genetic representation known as a chromosome. The chromosome is a structure that contains individual genetic information and is made up of an ordered sequence of genes. Typically, chromosomes are represented as binary chains of bits (0 and 1), although it is also common to find them represented as decimal chains; these are the most widely used forms.

Genes: Genes are the fundamental units of information within a chromosome, each defining a specific characteristic or variable of the proposed solution. For example, in an optimization problem, a gene might represent a parameter that needs to be adjusted to improve the solution. The bits that make up the gene, known as **alleles**, specify the possible values that this characteristic can assume.

Phenotype: The phenotype is the visible manifestation of an individual's genetic information. In the context of genetic algorithms, the phenotype refers to how the information contained in a chromosome is transformed into a concrete solution or a set of parameters that can be applied to the problem at hand.

Figure 2.3 shows an example of a binary population. In this case, each individual consists of a chromosome. In this particular example, the chromosomes are divided into two genes, each gene further composed of four alleles. Therefore, the chromosome contains a total of eight alleles. The phenotype, which represents the concrete interpretation of the problem, can be visualized in this example as the color of the circle.

The size of the population, as well as the length of the chromosome, depends on the complexity of the problem. If a more precise resolution is desired, it is beneficial to increase both the population size and the chromosome length. However, this comes

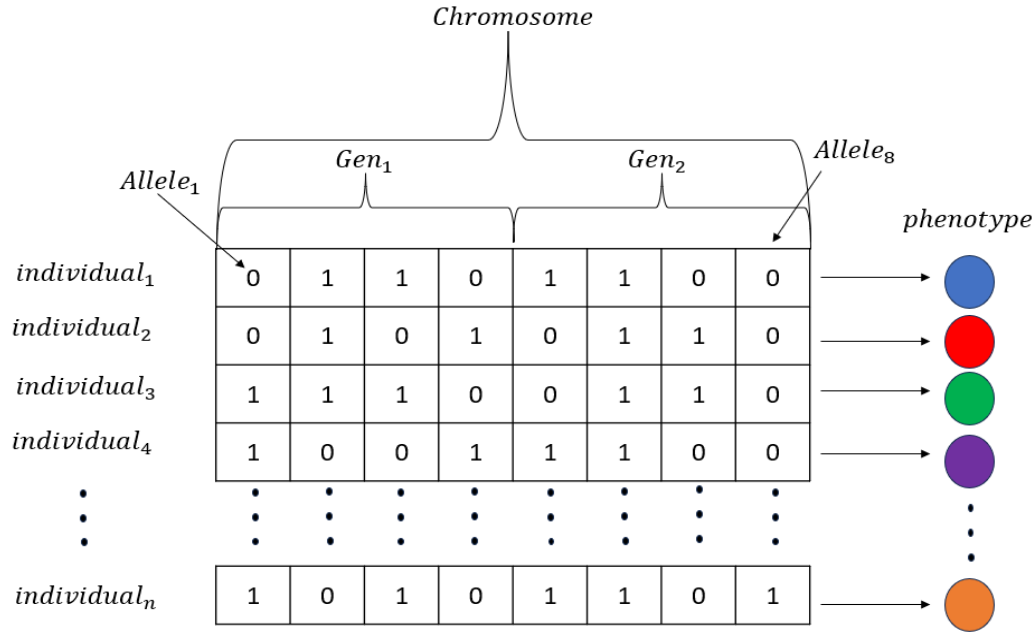


Figure 2.3: Binary representation of chromosomes and corresponding phenotypes.

with disadvantages, as larger populations and longer chromosomes imply higher computational costs. However, if both the population and chromosome sizes are small, the resolution of the problem will be more limited. Therefore, it is advisable to maintain a balance between these two parameters: they should be neither too large nor too small. Maintaining this balance allows for effective exploration of the search space and helps avoid issues associated with overly large or small populations, thus improving the algorithm ability to find optimal solutions. In addition, the population is usually generated randomly to ensure diversity. Maintaining diversity is beneficial, as it broadens the search space. If diversity is low, the search space is reduced, increasing the risk of falling into local minima. Heuristic methods can also be used to initialize the population, helping to achieve a higher average fitness value and accelerating the algorithm ability to find quality solutions. However, it is crucial to ensure that the gene pool is sufficiently large; otherwise, the population may lack diversity, become restricted to a limited portion of the search space, and fail to reach the global minimum.

Another significant challenge in genetic algorithms is the convergence time. A larger population size generally leads to better search space exploration and higher-quality solutions. However, it also increases not only the computational cost but also the execution time. It has been established that the time required for a genetic algorithm to converge is a function $O(n \log n)$ of the number of function evaluations, where n is the size of the population. A population is considered to have converged when all individuals are very similar, and further improvement is only possible through mutation. It has been shown that the effectiveness of the algorithm in reaching the global optimum, rather than a local one, largely depends on the population size. A large population is highly useful, but it also requires more memory, processing, and time.

To promote genetic diversity and generate new individuals, genetic algorithms employ operators such as crossover and mutation. These mechanisms are essential for exploring new regions of the search space and avoiding stagnation in local solutions, thus improving the algorithm's ability to reach optimal results.

2.3.3 ENCODING AND DECODING

Encoding is a technique used to represent genetic information that allows genetic data to be transformed into a specific format. This process can employ different types of representation, such as bits, numbers, trees, matrices, lists, or other objects, depending on the nature of the problem being addressed. The choice of encoding method depends on the specific problem. In certain cases, real or integer numbers can be used to achieve more efficient encoding.

A widely used encoding method is binary encoding. In this approach, each chromosome is represented by a binary string made up of bits. Each bit in the string can influence the proposed solution. It is important to note that, although a bit can represent a possible solution, this does not guarantee that it is the most optimal one.

The way bit strings encode information may vary depending on the characteristics and requirements of the specific problem.

Binary encoding offers a wide range of chromosome combinations using a relatively small number of alleles. However, it may not be suitable for certain types of problem and sometimes requires additional adjustments. Its primary use involves bit strings, and the length of the string varies according to the required precision.

The main characteristics of this type of encoding are:

- Integers are represented with precision.
- A limited number of real numbers can be represented.
- As the length of the string increases, the number of real numbers that can be represented also increases.

In this approach, each chromosome is represented as a string of values that may be related to the problem. This type of encoding is particularly effective for certain specific problems. However, it often requires the development of genetic operators tailored to the problem at hand. Direct value encoding is useful for problems involving complex values, such as real numbers, where binary encoding could be challenging and less efficient.

In value encoding, each chromosome consists of a sequence of values. These values can be any element related to the problem, ranging from integers and real numbers to characters or more complex objects.

2.3.4 GENETICS OPERATORS

Genetic operators can be considered the fundamental pillar of the algorithm, as they are responsible for preserving genetic diversity within the population. Their main

function is to generate new individuals, preventing solutions from becoming stagnant and allowing the algorithm to evolve and adapt to identify the best possible solution. The most commonly used genetic operators are selection, crossover, and mutation, and they often work in an interrelated manner.

Selection: The selection operator is responsible for determining which individuals will be chosen to reproduce and which will not. Analogous to natural processes, where survival depends on how well a species adapts to its environment, in genetic algorithms the selection is based on criteria defined by the application. This operator can be grouped into two main types: deterministic and probabilistic. Although they use different methods, both aim to choose the fittest individuals more often, increasing the chance of producing better solutions in future generations.

Deterministic criteria are based on the fitness value of each individual, and the number of times an individual is selected directly depends on its fitness value. In contrast, probabilistic criteria, as the name suggests, use random factors for selection. This approach offers the advantage of avoiding local minima, but may extend the convergence time. Among the most well-known probabilistic methods are roulette wheel selection and tournament selection.

Figure 2.4 shows the general functioning of these criteria. It presents a table with the individuals and their corresponding fitness values, allowing a clear view of how these values influence the selection process. Roulette wheel: Each individual in the population is assigned a percentage that represents their proportion on the roulette wheel, ensuring that the total sum of all percentages is equal to 1. Individuals with higher fitness will receive a larger section of the roulette wheel compared to those with lower fitness. In this way, when roulette spins, the people with the best fitness values have a significantly higher probability of being selected. Tournament: This selection method involves choosing genetic individuals through direct comparisons of their fitness values. In the deterministic version of the tournament method, a group of p individuals is randomly selected. Then, the individual with the highest

fitness value within this group is chosen to move on to the next generation, ensuring that only the fittest are selected.

Deterministic Selection: A deterministic selection criterion implies that the choice of individuals is made in a predictable manner, based on specific fitness values for each individual within the population. Fitness is calculated by evaluating how well an individual adapts to a particular problem or task. In Figure 2.4, it is shown that according to the deterministic criterion, the individuals are ranked from the lowest to the highest fitness value, and only half of the individuals with the best values are selected. It should be noted that the literature suggests that it is not advisable to completely eliminate all individuals from the parent generation. It is more effective to retain some individuals from previous populations to prevent the population from becoming overly homogeneous in just a few iterations [43].

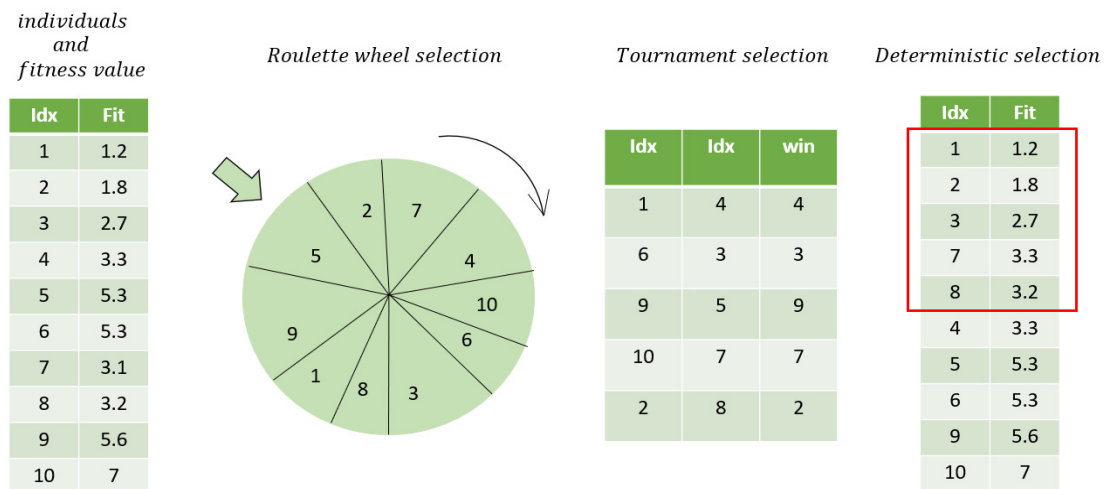


Figure 2.4: Selection methods in genetic algorithms.

Crossover operator: After the selection genetic operator identifies the fittest individuals, they proceed to the next operator, which is the crossover operator. In nature, this process is comparable to sexual reproduction, where the best-adapted or strongest parents are responsible for passing on their genes to future generations.

This operator is crucial within the algorithm, as it is responsible for combining genetic information to generate new individuals, known as offspring in genetic algorithms. Crossover percentages play an important role, as some researchers use high percentages to increase solution diversity, while others choose to work with lower percentages [43]. The crossover operator can be applied using different approaches, such as a destructive or non-destructive strategy. In a destructive strategy, a temporary population is used; even if the parents have a higher fitness value, the offspring are inserted into this temporary population. In contrast, with a non-destructive strategy, the offspring only move to the next generation if they meet an established fitness criterion.

The fundamental principle of the crossover operator is to combine two individuals to merge their genes or bits, especially when using binary encoding. There are several crossover criteria, the most common being single-point crossover, two-point crossover, and uniform crossover, as illustrated in Figure 2.5. These methods use crossover points that can vary randomly or involve multiple crossover points. Furthermore, genetic algorithms incorporate an elitism process that preserves the chromosome with the best performance, meaning the fittest solution is identified. Although this configuration favors rapid convergence, it sacrifices diversity.

Mutation operator: The final genetic operator is the mutation operator whose main function is to maintain diversity within the population and prevent it from becoming stagnant, meaning getting trapped in a local minimum. Although individuals can be selected directly from the population, the mutation process is carried out randomly, as shown in Figure 2.6. This operator introduces changes to the population before it is replaced by a new generation. Although the crossover operator conducts a global search at each stage, the mutation operator becomes more effective as the population begins to converge. Various researchers [44] have highlighted the advantages of mutation over other methods based solely on selection and crossover.

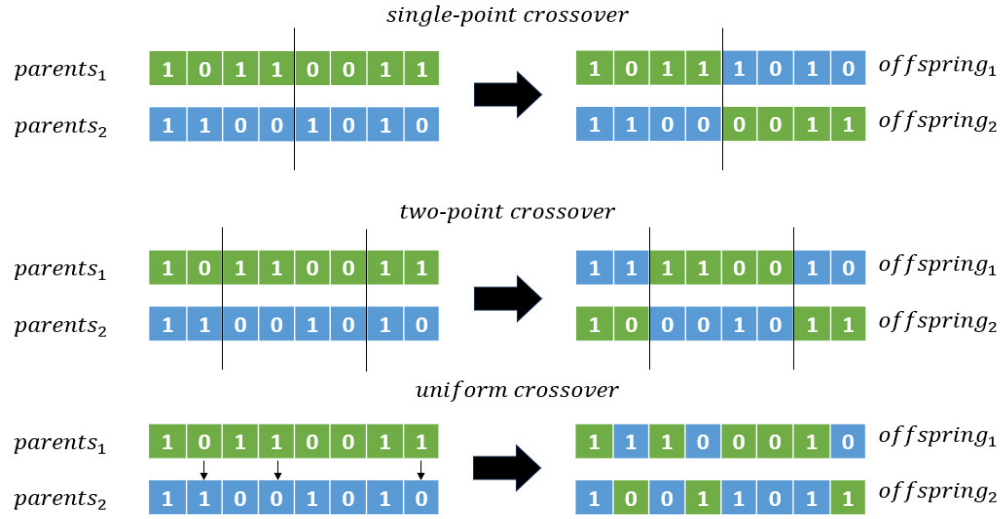


Figure 2.5: Types of crossover operators in genetic algorithms.

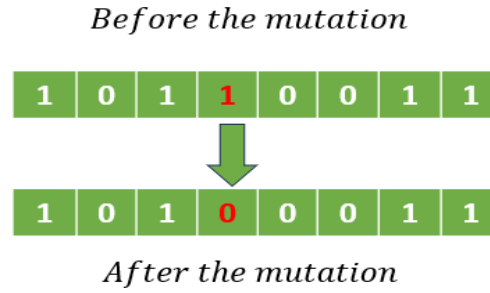


Figure 2.6: Mutation operator.

2.4 CONSENSUS THEORY

The collective movements of animal groups are among the most intriguing phenomena in nature and capture the attention of the research community. Although each individual has its own way of moving, their coordination creates the illusion of a single entity following its own set of motion laws. Examples such as flocks of birds, schools of fish, and herds of animals demonstrate how these groups gain a distinct

identity through the shared movement instincts of their members Figure 2.7. The group moves and turns to achieve goals such as finding food, migrating, and evading predators and obstacles.

Developing these skills and transferring them to a program, computer, or a group of robots is a complex challenge. Programming behavior as if it were a single entity would be very complicated if one tried to define all trajectories individually. Analyzing groups from a perspective focused on social behaviors can be difficult; however, members of these groups follow behaviors guided by rules that, despite seeming simple, facilitate efficient, responsive, and nearly instantaneous movements.

Computer animation can simulate the collective movement of animal groups by programming each individual with a set of simple, shared rules. These rules enable them to respond effectively to various situations. The combination of individual responses leads to the collective movement of the group. This content is inspired by the principles presented in the Reynolds rules.

Reynolds Rules: The collective movement of large groups can be described through the application of a set of simple rules that guide individual behavior. The movements of each member of the group result from the balance of two opposing tendencies: the intention to stay close to the group and the need to avoid collisions with others. Reynolds [45] adapted these tendencies into three fundamental rules, influenced by the movements of the immediate neighbors of each member.

1. **Collision avoidance:** maintain distance to prevent collisions with neighbors;
2. **Velocity matching:** synchronize speed and direction of movement with neighbors; and
3. **Flock centering:** remain within proximity to neighbors.

Reynolds rules efficiently capture the collective movement of groups of animals, and these rules can be applied to control schemes for human systems such as



Figure 2.7: Flock [1].

vehicle formations. These rules are based on each individual's perception of their environment and the actions of their close neighbors. Connecting structures, such as networks, are often used to illustrate these dynamically forming systems. Usually, these systems are modeled as nodes in the graph, commonly referred to as agents.

The communication graph figure 2.8 shows a representative graph for 6 agents. A graph is a pair $G = (V, E)$ with $V = \{v_1, \dots, v_N\}$ being a set of N nodes or vertices and E a set of edges or arcs. Elements of E are denoted as (v_i, v_j) which is termed an edge or an arc from v_i to v_j , and is represented as an arrow with tail at v_i and head at v_j . N_i is the number of neighbors where $N_i = \{v_j : (v_j, v_i) \in E\}$, and represents the nodes from which agent i receives information and in response to which it determines its own movement.

There are various methods for implementing Reynolds rules in dynamical system control. When considering the motion in R^2 , we can define $(p_i(t), q_i(t))$ as the position of the node i in the (x, y) plane. The agent state i is defined as $x_i = [p_i \ q_i]^T \in R^2$. In this context, it is assumed that all interaction graphs are

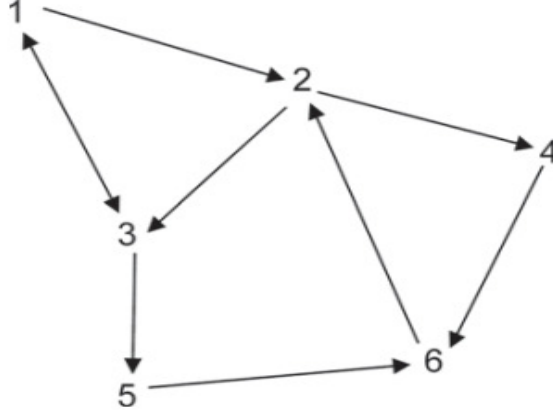


Figure 2.8: Connection graph.

undirected. That is, $a_{ij} = a_{ji}$, which implies that communication between agents occurs bidirectionally.

Agents are also capable of avoiding collisions. For this purpose, a radius ρ_c can be defined, beyond which each agent seeks to maintain a safe distance from others. The collision neighborhood for an agent i is expressed as $N_i^c = \{j : r_{ij} \leq \rho_c\}$, where r_{ij} indicates the distance between the nodes i and j , as presented in equation (2.3).

$$r_{ij} = |x_j - x_i| = \sqrt{(p_j - p_i)^2 + (q_j - q_i)^2} \quad (2.3)$$

Define also the interaction radius $\rho > \rho_c$ and the interaction neighborhood by $N_i = \{j : r_{ij} \leq \rho\}$.

The dynamics used to simulate individual group members can be very simple; yet realistic results are obtained. Consider the motion of the agent in two dimensions ($2 - D$) according to the dynamics Equation (2.4).

$$\dot{x}_i = u_i \quad (2.4)$$

with states $x_i = [p_i \ q_i]^T \in R^2$. This is a simple point-mass dynamics with

velocity control inputs $u_i = [u_{pi} \ u_{qi}]^T \in R^2$.

A suitable law for collision avoidance is given by Equation (2.5).

$$u_i = - \sum_{j \in N_i^c} c_{ij}(x_j - x_i) \quad (2.5)$$

which causes agent i to move away from other agents within the collision neighborhood N_i^c . Here, c_{ij} represents the collision avoidance gain.

A suitable law for flock centering is presented in Equation (2.6).

$$u_i = \sum_{j \in N_i / N_i^c} a_{ij}(x_j - x_i) \quad (2.6)$$

which causes agent i to turn towards other agents inside the interaction neighborhood N_i and outside the collision neighborhood N_i^c . The flock-centering protocol can be written in terms of the components of velocity as Equations (2.7),(2.8).

$$\dot{p}_i = u_{pi} = \sum_{j \in N_i / N_i^c} a_{ij}(p_j - p_i) \quad (2.7)$$

$$\dot{q}_i = u_{qi} = \sum_{j \in N_i / N_i^c} a_{ij}(q_j - q_i) \quad (2.8)$$

The relationship between the collision-avoidance gains c_{ij} and the flock-centering gains a_{ij} influences the different response behaviors, depending on whether agents tend to flock together or avoid their neighbors. Similarly, the relationship between the radii ρ and ρ_c is crucial for defining different behavioral patterns.

2.4.1 SECOND-ORDER CONSENSUS

One application of consensus in the field of cooperative control is the management of vehicle formations. Therefore, we aim to analyze consensus in coupled systems

that follow Newton's law $\ddot{x} = u_i$, resulting in second-order (double integrator) node dynamics, as given by Equations (2.9) and (2.10).

$$\dot{x}_i = v_i \quad (2.9)$$

$$\dot{v}_i = u_i \quad (2.10)$$

with a position $x_i \in \mathbb{R}$, speed $v_i \in \mathbb{R}$, and acceleration input $u_i \in \mathbb{R}$. The distributed position / velocity feedback is considered at each node, following the second-order local neighborhood protocols, given by Equation (2.11).

$$\begin{aligned} u_i &= c \sum_{j \in N_i} a_{ij}(x_j - x_i) + c\gamma \sum_{j \in N_i} a_{ij}(v_j - v_i) \\ &= \sum_{j \in N_i} ca_{ij}((x_j - x_i) + \gamma(v_j - v_i)) \end{aligned} \quad (2.11)$$

where $c > 0$ represents a stiffness gain and $c\gamma > 0$ a damping gain. This approach is based on local voting protocols that consider both position and velocity, allowing each node to adjust its positions and velocities with respect to its neighbors. This is a variant of the proportional-derivative control.

2.4.2 FORMATION CONTROL SECOND-ORDER PROTOCOL

In formation control, the goal is for all vehicles not only to reach a common consensus speed, but also to achieve steady-state positions that form a specific arrangement relative to one another. The protocol mentioned previously ensures that all nodes converge to the same final position. Additionally, in formation control, the positions are considered in a 2D or 3D space.

Consider that the formation moves in a n dimensional space and that the dynamics of the nodes are given by Equations (2.9) and (2.10).

Each node has a position vector $x_i \in \mathbb{R}^n$ and a velocity vector $v_i \in \mathbb{R}^n$. In the case of motion within a two-dimensional plane, the position of a node can be expressed as $x_i = [p_i \ q_i]^T$, where $(p_i(t), q_i(t))$ represents the position of node i in the coordinate plane (x, y) .

The desired constant position of node i relative to the moving formation center x_0 is defined as $\Delta_i \in \mathbb{R}^n$. If the intention is for the nodes to follow a leader or control node whose position and velocity are given by $\dot{x}_0 = v_0$, distributed second-order leader-following formation protocols can be applied at each node, with the behavior of this protocol described by Equation (2.12).

$$\begin{aligned}
 u_i &= \dot{v}_0 + \gamma k_v (v_0 - v_i) + k_p (x_0 + \Delta_i - x_i) \\
 &\quad + c \sum_{j \in N_i} a_{ij} ((x_i - \Delta_j) - (x_j - \Delta_i)) + c\gamma \sum_{j \in N_i} a_{ij} (v_j - v_i) \\
 &= \dot{v}_0 + \gamma k_v (v_0 - v_i) + k_p (x_0 + \Delta_i - x_i) \\
 &\quad + \sum_{j \in N_i} ca_{ij} ((x_i - \Delta_j) - (x_j - \Delta_i) + \gamma(v_j - v_i))
 \end{aligned} \tag{2.12}$$

where $c > 0$ represents a stiffness gain and $\gamma > 0$ denotes a damping gain. This protocol incorporates the leader velocity feedback with a gain of $\gamma k_v > 0$, the leader position feedback with a gain of $k_p > 0$, as well as the leader acceleration feedback.

CHAPTER 3

GENETIC ALGORITHM FOR MAZE ENVIRONMENT

The first application of the genetic algorithm focused on solving trajectories in complex environments, specifically mazes. Using a genetic algorithm that evaluated different movement sequences, the maze was successfully solved and intermediate navigation points were determined. Subsequently, smooth trajectories were generated using high-order polynomials based on these points. These trajectories were simulated using a dynamic quadrotor model and finally implemented in a real aerial vehicle to validate their effectiveness.

This approach demonstrated that the proposed genetic algorithm can find feasible solutions in complex search spaces and that these solutions can be integrated with dynamic models for execution on real platforms.

A genetic algorithm was proposed to solve a maze in a static environment. Instead of using a binary population, this algorithm uses a random population of real numbers, which reduces the length of the chromosome and simplifies the calculations of genetic operators. Each gene in the population is defined as $A \in \mathbb{R}$, with possible values in the set $A = \{1, 2, 3, 4, 5\}$. In addition, a population of 100 individuals with a maximum of 20 genes per chromosome is implemented.

The phenotype of the genetic algorithm is interpreted as the motion state variables for maze resolution. Each gene can take a value between 1 and 4, representing the four possible movements: right, up, left, and down. In addition, a fifth state is included, indicating that there is no movement. In this sense, when the sequence of movements reaches its final point, the individual remains static. To save processing time and avoid a fixed chromosome length, a variable-length chromosome criterion is used. This allows a case to be solved in three movements, while the same case could also be solved in two movements, as illustrated in Figure 3.1.

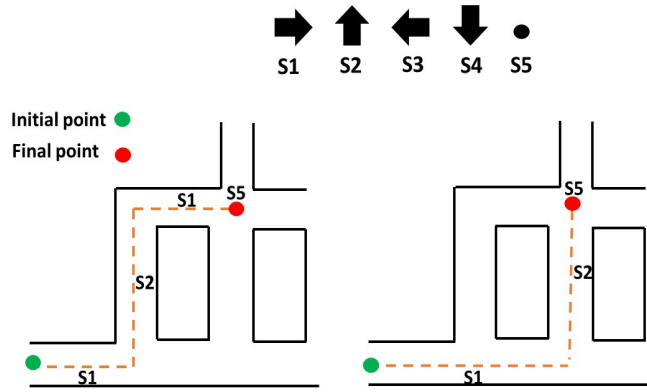


Figure 3.1: Schematic representation of movement sequences.

The proposed genetic algorithm aims to minimize the number of movements required to reach the final point in the maze. The arrangement of genes in the chromosome plays a crucial role, as it determines the specific sequence of movements within the environment. In this approach, the algorithm only evaluates the necessary genes until the destination is reached. The length of the chromosome is not fixed, but varies depending on the number of movements needed to reach the goal. Thus, the algorithm executes movements based on the index of each gene in the chromosome. In this model, the gene index is updated only when specific conditions are met.

The gene change on the chromosome follows three specific conditions. The first occurs when the candidate collides with a wall, preventing further movement

in that direction. Although the algorithm can anticipate the collision, the gene that defines the movement is not immediately modified. Instead, the candidate continues to attempt to move in the same direction until the actual collision occurs. Only when a gene-change condition is met does the algorithm update the gene to allow a new movement. This situation is illustrated in Figure 3.2. The second condition for gene change occurs when the candidate reaches a decision point, which means a location where multiple paths are available. In these cases, the algorithm must modify the gene to select a new movement that allows the best possible navigation within the maze. Finally, the third condition arises when the candidate reaches the final point of the maze. At this stage, a change occurs in the movement chromosome and a new candidate is selected within the population to evaluate an alternative solution. Figure 3.3 illustrates the gene change process under these conditions.

In Figure 3.3 a), it can be seen that after reaching the first decision point, the candidate has two viable movement options: moving upward or to the right. Although it could also move downward or backward, these paths present disadvantages, such as greater distance or the possibility of colliding with a wall. Due to these constraints, the algorithm penalizes these routes and discards them in future iterations. At this decision point, the algorithm increases the index that controls the movement gene, but the gene value remains unchanged. As a result, the candidate continues to move in the same direction, advancing to the right. Later, after reaching the second decision point, the algorithm modifies the chromosome again, increasing the movement gene index once more. At this stage, the next movement recorded on the chromosome dictates an upward displacement, allowing the candidate to continue in that direction until the final point, as illustrated in Figure 3.3 b). Finally, after reaching the final point, Figure 3.3 c) shows that the next gene on the movement chromosome acquires a value of five, indicating that the candidate has completed its path and will remain in its position without further movement.

In addition, penalty conditions *pen* are introduced for movement sequences that cause the candidate to exit the maze or collide with a wall. These penalized paths are discarded in subsequent iterations. Genetic operators prioritize movement sequences that bring the candidate closer to the final point while avoiding collisions and ensuring that it remains within the maze boundaries.

$$dist_j = \sqrt{(xp_{j,m} - stopx)^2 + (yp_{j,m} - stopy)^2} \quad (3.1)$$

In Equation (3.1), *dist* represents the distance between the last movement and the final point. The values *xp* and *yp* indicate the current position of the movement sequence and are updated at each iteration. Meanwhile, *stopx* and *stopy* correspond to the coordinates of the final point in the maze.

The parameter *crash* acts as a counter that records the number of times a movement sequence collides with a wall, while *pen* is a significant penalty applied when the candidate moves beyond the boundaries of the maze. In addition, *Countm* tracks the number of movement changes within the sequence and *Countp* counts the steps required to reach the final point. In this context, a movement sequence that successfully reaches the goal meets the defined criteria and that does so in fewer steps is considered a better solution because it minimizes the total path distance. From this perspective, the objective function is defined by Equation (3.2), where k_i is a constant with $i = \{1, 2, 3, 4\}$.

$$fv_j = dist_j + k_1 * Countm_j + k_2 * Countp_j + k_3 * crash_j + k_4 * pen_j \quad (3.2)$$

3.2 GENETIC OPERATORS FOR MAZE SOLVING

3.2.1 SELECTION-CROSSOVER-MUTATION

For the selection of candidates, a deterministic criterion is used. Based on the fitness value of the movement sequences, the candidates are ordered from lowest to highest, and the top half of the population with the best fitness values is chosen for crossover. The crossover operator implements a double crossover point criterion. As shown in Figure 3.4, a primary crossover point PC splits the chromosome, establishing a boundary between the secondary crossover points P_{csub1} and P_{csub2} , which are updated in each iteration. Since all selected individuals participate in the crossover process, the recombination rate reaches 100 %.

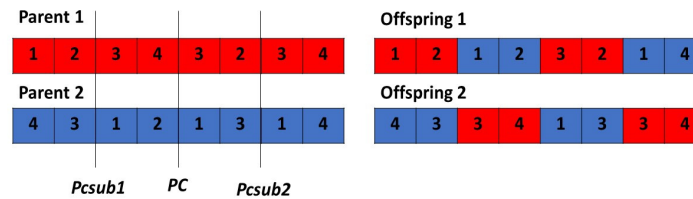


Figure 3.4: Double crossover point.

A high mutation rate criterion is applied, with a percentage of 4 %, where 80 % of the population undergoes genetic modifications. However, the best-performing individuals are preserved for future iterations, allowing some to remain unchanged to promote convergence. A high mutation rate improves exploration within the search space and increases movement diversity. Mutation occurs by altering the sequence of movements and assigning a new random value between 1 and 4.

The general functioning of the genetic algorithm is detailed in *Algorithm 1*, where the gene modification criteria in the movement chromosome are also established, with the variable i managing the index. Furthermore, *Algorithm 2* illustrates different movement scenarios and the counters used to determine the number of accessible paths. When there are two possible paths, it is considered that the candidate has reached a decision point, which is recorded by *con1*, while *con2* functions as a counter to ensure that the candidate remains within the limits of the maze.

Algorithm 1 Genetic algorithm to solve maze

Set population size P , number of chromosomes Nc

Set crossover rate Pc , mutation rate Pm Set counters $i = 1, j = 1, g = 1$ Initialize population $Move_{P,Nc}$ **while** $g \leq gene$ **do** **while** $j \leq P$ **do** $xp = startx; yp = starty;$ **while** $i \leq Nc$ **do** $n = move_{j,i}$ **Algorithm2**(n) **if** $con2 \geq 4$ **then** **if** $con1 < 1$ **then** $i = i + 1$ **if** $xp < 0$ **or** $yp < 0$ **or** $xp > pfx$ **or** $yp > pfy$ **then** $fv_j = fv_j + p$ $i = i + 1$ **if** $xpf = obsx$ **and** $yfp = obsy$ **then** $i = i + 1$ **if** $xp = obsx$ **and** $yp = obsy$ **then** $crash_j = crash_j + 1$ $i = i + 1$ $ic = i$ **if** $xp = stopx$ **and** $yp = stopy$ **then** $move(i + 1) = 5; i = Nc$ $m = m + 1$ $countp = m; countm = ic$

Calculate distance by Eq. (2.4)

 $j = j + 1;$

Evaluate objective function with Eq. (2.3)

Selection, Crossover, Mutation, Replace population

 $g = g + 1$

Algorithm 2 Cases algorithm

switch n **do** **case** 1 *Move right* **do** $xpf = posx + (dp * 2); ypf = yp;$ $xp = xp + dp; yp = yp;$ **case** 2 *Move up* **do** $xpf = xp; ypf = yp + (dp * 2);$ $xp = xp; yp = yp + dp;$ **case** 3 *Move left* **do** $xpf = xp - (dp * 2); ypf = yp;$ $xp = xp - dp; yp = yp;$ **case** 4 *Move down* **do** $xpf = xp; ypf = yp - (dp * 2);$ $xp = xp; yp = yp - dp;$ **case** 5 *No move* **do**

—

 $con1 = 0; con2 = 0;$ $x_1 = xp + \delta p; y_1 = yp + \delta p; x_2 = xp; y_2 = yp + \delta p; x_3 = xp - \delta p; y_3 = yp; x_4 = xp;$ $y_4 = yp - \delta p; x_5 = xp + \delta p; y_5 = yp + \delta p; x_6 = xp + \delta p; y_6 = yp - \delta p; x_7 = xp - \delta p;$ $y_7 = yp + \delta p; x_8 = xp - \delta p; y_8 = yp - \delta p;$ **while** $n \leq 8$ **do** **if** $x_n = obsx$ *and* $y_n = obsy$ **then** $con1 = con1 + 1;$ **if** $i \geq 5$ **then** $con2 = con2 + 1;$

CHAPTER 4

GENETIC ALGORITHM FOR PATH PLANNING AND FLIGHT FORMATION.

4.1 GENETIC ALGORITHM INITIALIZATION

The process for initializing the genetic algorithm used for vehicle path planning and obstacle avoidance is outlined below. The algorithm begins with a randomly generated initial population consisting of 24 alleles and 60 individuals for static and dynamic environments. These individuals are organized in a matrix-like structure, as illustrated in Equation (4.1), to facilitate the handling of information within the algorithm. In this context, $a \in \mathcal{A}$, where $\mathcal{A}$ represents a set containing the binary elements $\{0, 1\}$, i.e., $\mathcal{A} = \{0, 1\}$.

$$Po_i = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,24} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,24} \\ \vdots & \vdots & \ddots & \vdots \\ a_{60,1} & a_{60,2} & \cdots & a_{60,24} \end{pmatrix} \quad (4.1)$$

After the initialization process, an encoding step is required to convert the binary data into values that the application can interpret. In this context, submatrices

are used, each storing the spatial coordinate information for each axis. The submatrix sub_1 holds the equation of the coordinate X Equation (4.3), sub_2 corresponds to the Equation of the coordinate Y Equation (4.4), and sub_3 contains the Z coordinate information, as shown in Equation (4.5).

$$Po_i = \begin{pmatrix} [sub_1] & [sub_2] & [sub_3] \end{pmatrix} \quad (4.2)$$

$$[sub_1] = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,8} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,8} \\ \vdots & \vdots & \ddots & \vdots \\ a_{60,1} & a_{60,2} & \cdots & a_{60,8} \end{pmatrix} \quad (4.3)$$

$$[sub_2] = \begin{pmatrix} a_{1,9} & a_{1,10} & \cdots & a_{1,16} \\ a_{2,9} & a_{2,10} & \cdots & a_{2,16} \\ \vdots & \vdots & \ddots & \vdots \\ a_{60,9} & a_{60,10} & \cdots & a_{60,16} \end{pmatrix} \quad (4.4)$$

$$[sub_3] = \begin{pmatrix} a_{1,17} & a_{1,18} & \cdots & a_{1,24} \\ a_{2,17} & a_{2,18} & \cdots & a_{2,24} \\ \vdots & \vdots & \ddots & \vdots \\ a_{60,17} & a_{60,18} & \cdots & a_{60,24} \end{pmatrix} \quad (4.5)$$

Each coordinate value is represented by 8 bits, as illustrated in equation (4.6). During binary-to-decimal conversion, if $a_{1,1} \dots a_{1,8} = 0$, then $x_{dec} = 0$, indicating that the minimum possible value is 0. Conversely, if $a_{1,1} \dots a_{1,8} = 1$, the binary string reaches its maximum representable value, which is $x_{dec} = 255$, as shown in equation (4.7). Finally, the obtained decimal value is assigned to a real-world space value. In this example, a three-dimensional space with dimensions 15x15x15 is assumed. To accomplish this, the binary value matrix is divided by a conversion factor of 17, as detailed in Equation (4.8).

$$x_b = [a_{1,1}, a_{1,2}, a_{1,3}, a_{1,4}, a_{1,5}, a_{1,6}, a_{1,7}, a_{1,8}] \quad (4.6)$$

$$x_{dec11} \in B, B = \{0...255\} \quad (4.7)$$

$$Prob_{dec} = \left([x_{dec}]/17 \quad [y_{dec}]/17 \quad [z_{dec}]/17 \right) \quad (4.8)$$

In genetic algorithms that utilize binary populations, the search resolution is influenced both by the length of the chromosome chain and by the size of the population. In the proposed algorithm, 8-bit chains are used for each spatial coordinate X , Y , and Z , with genetic operators applied to create new individuals.

4.1.1 OBJECTIVE FUNCTIONS

The objective function is used to evaluate the quality of the path, using a multi-objective approach with three distinct criteria. Let the multi-objective function f_i be defined by three separate objective functions, each serving as a criterion. Consider that the multi-objective function f_i consists of three objective functions that are treated as an individual criteria Equation (4.9).

$$\min f_i = (f1_i + f2_{i,j} + f3_{i,j}) \quad (4.9)$$

In this context, $f1_i$ corresponds to the criterion related to the total length of the path, $f2_{i,j}$ represents the criterion related to the distance between a point of entry and an obstacle, and $f3_{i,j}$ accounts for the criterion related to the cumulative sum of the path angles. Here, i denotes the index of the individual, while j indicates the index of obstacles.

The first criterion focuses on reducing the length of the path to achieve the shortest possible distance. The second criterion aims to minimize the distance between the waypoints and obstacles to identify the unfeasible waypoints. The third criterion addresses the minimization of the sum of the path angles to evaluate whether the path intersects obstacles. Consequently, the genetic algorithm is designed to minimize the function f_i .

4.1.2 LENGTH OF PATH

The first criterion of the multi-objective function measures the distance from the initial point through the waypoint to the final destination, as illustrated in Figure 4.1. Each path is evaluated, with longer distances receiving a higher fitness value, which reduces the likelihood of reproduction. The pseudocode corresponding to this first objective function can be found in the algorithm 3 within the section for objective function 1. Equation (4.9) is used to calculate the distance from the starting point, through the waypoint, to the endpoint.

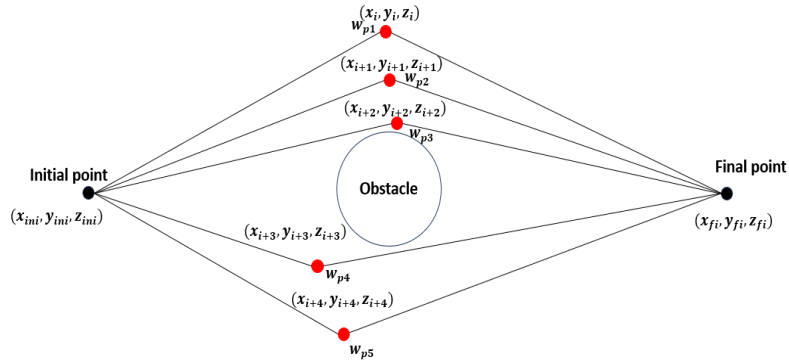


Figure 4.1: Path length evaluation.

$$f1_i = \sqrt{(x_{ini} - x_i)^2 + (y_{ini} - y_i)^2 + (z_{ini} - z_i)^2}$$

$$+ \sqrt{(x_i - x_{fi})^2 + (y_i - y_{fi})^2 + (z_i - z_{fi})^2} \quad (4.10)$$

4.1.3 DISTANCE BETWEEN WAYPOINT AND OBSTACLES

The second criterion applies penalties to paths where waypoints overlap with obstacles, leading to their elimination in early iterations. This prevents genetic operators from working with inefficient paths in later stages. By discarding unfeasible waypoints, this criterion encourages rapid convergence, which is particularly advantageous in dynamic environments.

The second criterion of the multi-objective function is described in Algorithm 1, under the section for the objective function 2. As shown in Figure 4.2, the algorithm calculates the distance between the waypoints and obstacles. Each waypoint is evaluated, and if the distance is less than or equal to the radius of an obstacle, the waypoint is considered to be inside the obstacle. In such cases, a penalty is applied that negatively impacts the fitness value.

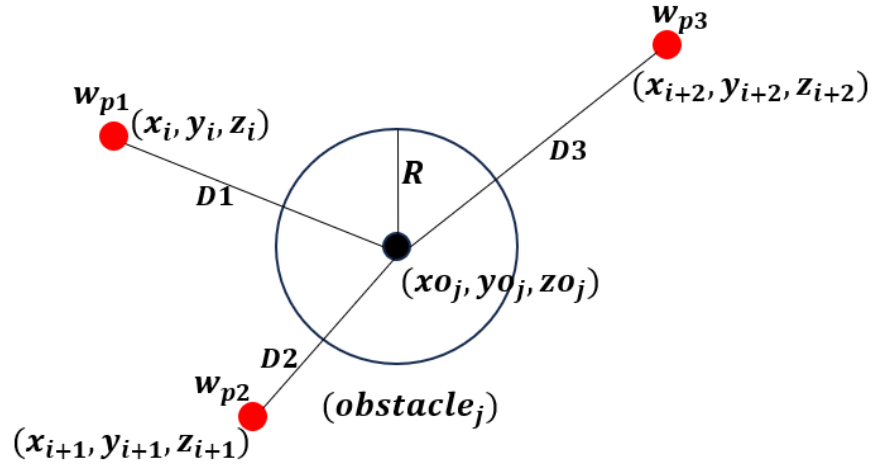


Figure 4.2: Evaluation waypoint.

A strategy in a two-dimensional plane is implemented, as shown in Figure 4.3. Calculations involve projecting obstacles and waypoints onto the XY , YZ , and XZ planes. Equations (4.11)-(4.13) are used to calculate the distances of obstacles within each respective plane, with x_{o_j} , y_{o_j} , and z_{o_j} representing the obstacle's coordinates. If all three conditions are met, indicating that the waypoint is within the obstacle, a penalty is applied, as indicated in Equation (4.14).

$$doxy_i = \sqrt{(x_i - x_{o_j})^2 + (y_i - y_{o_j})^2} \quad (4.11)$$

$$doxz_i = \sqrt{(x_i - x_{o_j})^2 + (z_i - z_{o_j})^2} \quad (4.12)$$

$$doyz_i = \sqrt{(y_i - y_{o_j})^2 + (z_i - z_{o_j})^2} \quad (4.13)$$

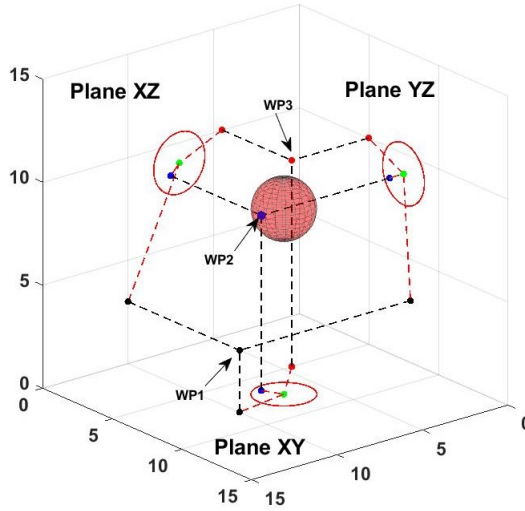


Figure 4.3: Evaluation planes 2D.

$$f2_i = \sum_{j=1}^{numobs} fs2_{i,j} \quad (4.14)$$

where is initialized to 0 and is expressed by the following Equation (4.15)

$$fs2_{i,j} = \begin{cases} fs2_{i,j} + p, & doxy_i \& doxz_i \& doyz_i \leq radius \\ fs2_{i,j} + 0, & doxy_i \text{ or } doxz_i \text{ or } doyz_i > radius, \end{cases} \quad (4.15)$$

4.1.4 SUM OF ANGLES OF PATH

The third criterion is included in algorithm 1 under objective function 3 and is used to verify whether the path intersects the obstacle. To determine whether the angle crosses an obstacle, the angle formed between the path and the virtual vectors of the right and left tangent points is calculated, as shown in Figure 4.4. Equation (4.16) is used to find the angle between the tangent points, called αDI (angleDI). The angle between the path and the virtual vector on the right is calculated using Equation (4.17) and is indicated by αLI (angleLI). Finally, the angle with the virtual vector on the left is determined using Equation (4.18) and is labeled αLD (angleLD).

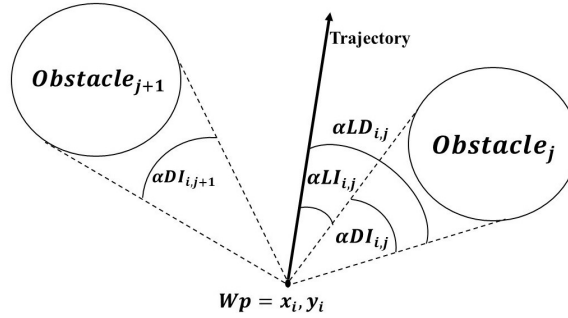


Figure 4.4: Evaluation of path crossing by obstacles.

$$\alpha DI_{i,j} = \arccos \frac{\overrightarrow{U_{i,j}} \cdot \overrightarrow{V_{i,j}}}{|\overrightarrow{U_{i,j}}| |\overrightarrow{V_{i,j}}|} \quad (4.16)$$

$$\alpha LI_{i,j} = \arccos \frac{\overrightarrow{U_{i,j}} \cdot \overrightarrow{W_{i,j}}}{|\overrightarrow{U_{i,j}}| |\overrightarrow{W_{i,j}}|} \quad (4.17)$$

$$\alpha LD_{i,j} = \arccos \frac{\overrightarrow{W_{i,j}} \cdot \overrightarrow{V_{i,j}}}{|\overrightarrow{W_{i,j}}| |\overrightarrow{V_{i,j}}|} \quad (4.18)$$

The calculation of angles (αLD), (αLI), (αDI) is performed in a 2D plane to optimize processing and computation time. This procedure is applied to all three planes: XY , XZ , and YZ . The number of times a path intersects an obstacle is stored in the variables $pesoxy_{i,j}$, $pesoxz_{i,j}$, and $pesoyz_{i,j}$ for each plane. Equation (4.19) outlines the conditions needed to determine whether a path crosses an obstacle. The fitness values are multiplied by a constant and then added to their original fitness value using Equation (4.20).

$$\begin{bmatrix} pesoxy_{i,j} \\ \dots \\ pesoxz_{i,j} \\ \dots \\ pesoyz_{i,j} \end{bmatrix} = \begin{cases} pesoxy_{i,j} + 1, \alpha DIxy_{i,j} = \alpha LDxy_{i,j} + \alpha LDxy_{i,j} \\ pesoxy_{i,j} + 0, \alpha DIxy_{i,j} \neq \alpha LDxy_{i,j} + \alpha LDxy_{i,j} \\ \dots \\ pesoxz_{i,j} + 1, \alpha DIxz_{i,j} = \alpha LDxz_{i,j} + \alpha LDxz_{i,j} \\ pesoxz_{i,j} + 0, \alpha DIxz_{i,j} \neq \alpha LDxz_{i,j} + \alpha LDxz_{i,j} \\ \dots \\ pesoyz_{i,j} + 1, \alpha DIyz_{i,j} = \alpha LDyz_{i,j} + \alpha LDyz_{i,j} \\ pesoyz_{i,j} + 0, \alpha DIyz_{i,j} \neq \alpha LDyz_{i,j} + \alpha LDyz_{i,j} \end{cases} \quad (4.19)$$

$$f3_i = \sum_{j=1}^{numobs} pesoxy_{i,j} * c + \sum_{j=1}^{numobs} pesoxz_{i,j} * c + \sum_{j=1}^{numobs} pesoyz_{i,j} * c \quad (4.20)$$

4.2 GENETIC OPERATORS

The parameters of the genetic operators are precisely adjusted to achieve rapid convergence without leading to an inefficient solution, allowing the genetic algorithm to respond effectively to unexpected changes in dynamic environments. The strategy employs an elitist selection approach that facilitates quick convergence while avoiding being trapped in local minima, thanks to a broad sampling resolution and high crossover and mutation rates, which further enhance the sampling resolution. To ensure convergence even with a high mutation criterion, a subset of the population that remains unaffected by mutation is implemented, called the convergence population. This population ensures that the convergence process is not negatively impacted by a high mutation rate.

4.2.1 SELECTION OPERATOR

The first genetic operator, called natural selection, is created after evaluating individuals using the objective function. This operator selects individuals who best meet the conditions based on their fitness value. The best individuals have a fitness value of 0 for conditions like $pesoxy_{i,j}$, $pesoxz_{i,j}$, and $pesoyz_{i,j}$, and they have the shortest path lengths. However, even individuals who have not met the conditions can be selected, promoting diversity in the population. A natural selection operator with elitist criteria is used, ordering individuals from the lowest to the greatest fitness value. The operator then selects the half of the population with the lowest fitness value for reproduction in the crossover operator.

4.2.2 CROSSOVER OPERATOR

The crossover operator is essential for generating new potential solutions to the problem. It functions by reproducing individuals selected through a process of natural selection. A single crossover point is assigned to each axis, ensuring that crossover takes place only along these points. This crossover point is determined randomly in each iteration, and the pairing of the parent individuals is also performed randomly, as shown in Figure 4.5. A crossover rate of 100% is applied to the population selected by the crossover operator, which means that all individuals in the selected population undergo a crossover.

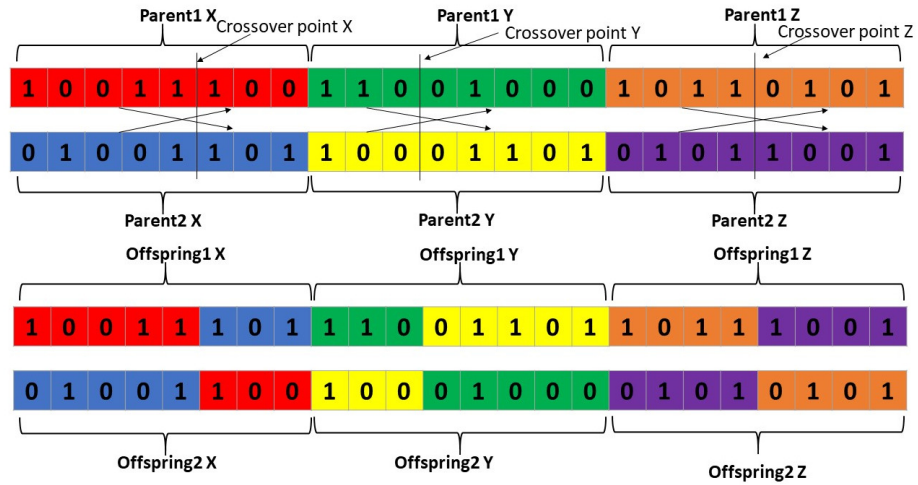


Figure 4.5: Crossover genetic operator.

4.2.3 MUTATION OPERATOR

The mutation operator plays a vital role in maintaining population diversity and preventing premature convergence. Similarly to the crossover operator, mutation occurs randomly by modifying the bit chains within the population. The number

of genes affected by the mutation is determined using Equation (4.21) and (4.22), where P denotes the total number of individuals and n represents the number of chromosomes.

$$Tgenes = P \cdot n \quad (4.21)$$

$$Genesmut = Tgenes \cdot prob/100 \quad (4.22)$$

Algorithm 3 Genetic algorithm for path planning.

```

Set initial point  $x_{ini}, y_{ini}, z_{ini}$  and final point  $x_{fi}, y_{fi}, z_{fi}$ 

Set variable  $P$  population size,  $N_g$  number of genes.
Set  $Pc$  crossover rate,  $Pm$  Mutation rate
Set  $numobs$  as number of obstacles.
Set counter  $i = 1, j = 1, k = 1$ 
Set  $radius$  as radio of obstacles.
Set  $numway$  as number of waypoints.
Initializes population
while UAV moves to the final point do
    while  $i \leq P$  do
        Calculate the distance  $f1_i$  by (4.9)                                //Objective function 1
        while  $j \leq numobs$  do
            Calculate distance between the waypoints and obstacles by
             $doxy_i$  eq (4.11),  $doxz_i$  eq (4.12),  $doyz_i$  eq (4.13)    //Objective function 2
            if  $doxy_i \& doxz_i \& doyz_i \leq radius$  then
                | add penalty by eq (4.14)
            else
                | do not add penalty
            end
            while  $k \leq numway + 1$  do
                get angle  $\alpha DI_{i,j}$  by eq. (4.16)                    //Objective function 3
                get angle  $\alpha LI_{i,j}$  by eq.(4.17)
                get angle  $\alpha LD_{i,j}$  by eq.(4.18)
                 $AngleSum = \alpha LD_{i,j} + \alpha LI_{i,j}$ 
                if  $\alpha DI_{i,j} = AngleSum$  then
                    |  $peso_k = 1;$ 
                else
                    |  $peso_k = 0;$ 
                end
                |  $k = k + 1$ 
            end
            |  $j = j + 1$ 
        end
        |  $i = i + 1$ 
    end

    Selection operator
    Crossover operator
    Mutation operator
    Criteria for dynamic environment
    Replace population
end

```

4.3 CRITERIA FOR DYNAMIC ENVIRONMENT

The algorithm is designed for dynamic environments, where recalculating the path is essential. When a waypoint converges and an obstacle interferes, maintaining population diversity is crucial to generating a new path. To achieve this, a high mutation strategy is used with a mutation rate of 2%, which means that 50% of the population is altered in each iteration. This increases diversity but may lead to abrupt changes in the UAV path due to the high mutation rate.

To avoid this problem, a population convergence criterion is implemented, which ensures that the first 25% individuals remain unaffected by mutation. This convergence population preserves the best waypoint, allowing the UAV to follow it consistently. In addition, when an individual proves to be an effective solution, it is incorporated into the convergence population, replacing a lower-quality individual. This iterative process improves the quality of the convergence population over time.

Once a UAV successfully navigates around an obstacle, there is a chance that it will stall if the waypoint has already converged, leading to a loss of population diversity. To avoid this issue, a new random population is generated, introducing diversity and allowing the aerial vehicle to identify a new waypoint to follow. The algorithm activates this mechanism when the distance between the UAV and the waypoint drops below a predefined threshold (*distr*). In this way, 50% of the population is regenerated when the UAV approaches the waypoint, maintaining diversity and allowing the UAV to continue following a different trajectory.

When the vehicle avoids all obstacles, it may have trouble finding a new waypoint to reach the final destination, which can cause delays. To avoid this problem, a rule is used that allows the UAV to move directly to the final destination without recalculating the path, as long as there are no obstacles between its current path and the goal.

The coordinates of the final point are found, converted into binary format, and added to the convergence population. This path is not affected by the high mutation rate. If there are no obstacles in the way, the final point stays in the convergence population. As a result, it becomes the best waypoint for the population. The Algorithm 4 shows the proposed method.

Algorithm 4 Criteria for static and dynamic environment.

```

Set distr as distance limit between initial point and waypoint.

Set Variable n as number of alleles

Set Variable fc as conversion factor

Set constant psafe as number of population of convergence

 $Tgenes = P * n;$ 
 $GenesMut = Tgenes \cdot prob/100$ 
while  $i \leq GenesMut$  do
     $fila = randi([1, P])$ 
     $colum = randi([1, n])$ 
    if  $fila \geq P_{safe}$  then
         $Pob(fila, colum) = 1 - Pob(fila, colum);$ 
    end
     $i = i + 1$ 
end

Calculate dist between Initial point and waypoint eq (4.9)

if  $dist \leq distr$  then
     $Pob(P/4 + 1 : P, 1 : n) = round(rand(P/4, n));$ 
end

 $x_{fi}, y_{fi}, z_{fi}$  //Final point
 $xc = x_{fi}/fc, yc = y_{fi}/fc, zc = z_{fi}/fc$ 
 $xb = dec2bin(xc), yb = dec2bin(yc), zb = dec2bin(zc)$  //Convert to binary
 $Pob(5, 1 : n) = [xb, yb, zb]$  //enter the final point in the population of conver-
gence

```

4.4 SIMULATION AND EXPERIMENTAL PLATFORM

The equations for generating the continuous path are shown by Equations (4.23)-(4.25), where a low-pass filter is used to differentiate the path; $xd(t)$, $yd(t)$, $zd(t)$ are the desired position, t is the time; ta is an earlier time, and x_0, y_0, z_0 is the initial position.

$$xd(t) = mx * (t - ta) + x_0 \quad (4.23)$$

$$yd(t) = my * (t - ta) + y_0 \quad (4.24)$$

$$zd(t) = mz * (t - ta) + z_0 \quad (4.25)$$

where

$$mx = ((x_f - x_0)/(t_f)) \quad (4.26)$$

$$my = ((y_f - y_0)/(t_f)) \quad (4.27)$$

$$mz = ((z_f - z_0)/(t_f)) \quad (4.28)$$

The coordinates x_f , y_f , and z_f represent the waypoint selected by the genetic algorithm. The final time is defined as $t_f = \text{newdist}/\text{vel}$, where newdist is the new path distance if a change occurs, and vel is the velocity of the UAV. This final time is recalculated each time the waypoint changes to ensure that the velocity stays constant.

Furthermore, t_f and the velocity components mx , my , and mz are only updated when the waypoint changes. If there is no change, the values of these components remain the same.

4.4.1 A CASE STUDY: CRAZYFLIE QUADROTOR

Simulations were conducted to evaluate the performance of the algorithm, which was programmed and executed using LabVIEW software. All simulations were performed on a computer equipped with a Core i7-8750H processor, an Nvidia GTX-1060 GPU, and 16 GB of DDR4 RAM running at 2400 MHz. For the experimental test, a Crazyflie drone was used [46] [47], and its transfer functions, equation (4.29) for the positions X and Y , and equation (4.30) for the position Z .

$$G(s) = \frac{0.9549}{s^2 + 1.229s + 0.9324} \quad (4.29)$$

$$G(s) = \frac{1850}{s + 1896} \quad (4.30)$$

Figure 4.6 provides an overview of the algorithm proposed for 3D path planning for quadrotor UAVs. The 3D path planning generator is based on a genetic algorithm executed on a computer that acts as a ground station. This generator uses a genetic algorithm whose objective function evaluates the position of the UAV, the environmental conditions (obstacles), and the population in decimal form Pob_{dec} (waypoints).

The genetic operators are then applied to create new individuals and maintain population diversity. As a result, the population is updated and a new trajectory is generated. The quadrotor uses a PID controller in real time to follow the UAV references produced by the trajectory generator. It is important to note that this approach is implemented for both single and multiple UAVs (swarms).

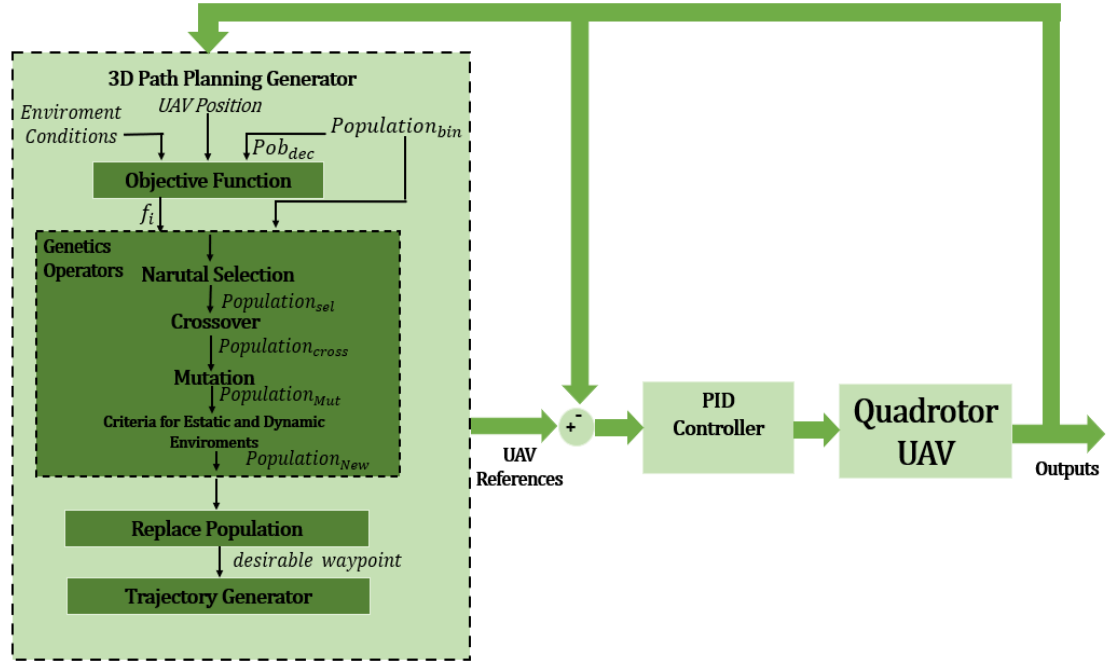


Figure 4.6: Block Diagram of the UAV Guidance System Using Genetic Algorithms.

4.4.2 EXPERIMENTAL PLATFORM

Real-time experiments are carried out in the Navigation Laboratory of the Center for Research and Innovation in Aeronautical Engineering (CIIIA), FIME-UANL. For the detection of UAVs and obstacles, a motion capture system consisting of 16 T-40 VICON cameras is used and Crazyflie 2.1 drones are used as Crazyswarm crazyswarm.

CHAPTER 5

FORMATION FLIGHT BASED ON CONSENSUS

In this chapter, the theory behind the consensus algorithm used is presented. This algorithm is based on a second-order protocol designed to coordinate multiple unmanned aerial vehicles (UAVs). This type of algorithm allows each agent to adjust its position and velocity based on information provided by its neighbors, thus achieving distributed coordination without the need for a central control unit.

The design of a delta formation composed of three vehicles is described in detail, enabling the maintenance of a balanced triangular structure during flight. Subsequently, the coordination of five vehicles is developed under two different configurations: an extended delta formation and a line formation, which are suitable for exploration missions and tracking along straight paths.

In addition, the adjacency matrices, neighbor matrices, and offset vectors that define the relative positions between vehicles in each configuration are included. The stability of the system and the convergence of the trajectories toward the desired objectives are also analyzed, validating the algorithm's performance through numerical simulations and experimental tests.

Finally, this chapter also presents how the consensus strategy was integrated

with the genetic algorithm developed in the previous chapter.

To maintain the consensus formation, Equation (5.1) was applied, where c represents the influence of the connection, which functions as a proportional gain. In this equation, $a_{ij} = A$ denotes the adjacency matrix, x_i is the position of the vehicle to determine its control signal, x_j indicates the position of the neighboring vehicle, $\gamma = 3$ reflects the influence of the velocity error or derivative gain, v_i is the velocity of the current vehicle, and v_j is the velocity of the adjacent vehicle.

$$u_i = \sum_{j \in N_i} c a_{ij} ((x_i - \Delta_j) - (x_j - \Delta_i) + \gamma(v_j - v_i)) \quad (5.1)$$

5.1 THREE UAVS IN FLIGHT

Figure 5.1 displays the connection graph for the configuration involving 3 vehicles. In this formation, a distance of $2m$ is maintained between each vehicle. Vehicle 3 acts as the leader, vehicle 2 follows vehicle 3, and vehicle 1 trails behind vehicle 2.

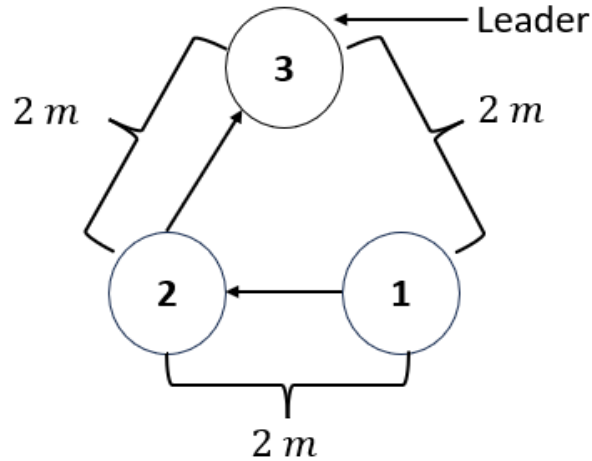


Figure 5.1: Graph of Connections for 3 vehicles.

These initial positions, which are critical for analyzing the dynamics of the consensus algorithms, are detailed in Table 5.1.

Table 5.1: Initial positions of 5 vehicles

	$x(m)$	$y(m)$	$z(m)$
UAV1	2	2	2
UAV2	4	4	4
UAV3	6	6	6

The connectivity matrix A in Equation (5.2) serves as an adjacency matrix, detailing the connections between nodes in a network. A is structured as a square matrix, where each row and column correspond to a specific node in the network. The elements of this matrix indicate whether there is a direct connection between any pair of nodes. A diagram illustrating these connections is presented in Figure 5.1.

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad (5.2)$$

The neighbor matrix N_i is a key tool in network analysis, offering detailed information about the direct neighbors of each node within the network. The matrix N_i is presented in Equation (5.3).

$$N_i = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix} \quad (5.3)$$

The matrices Δ_i and Δ_j represent the offsets for the formation.

$$\Delta_i = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (5.4)$$

$$\Delta_j = \begin{bmatrix} 2 & -1 & 0 \\ 0 & 1.732 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (5.5)$$

5.2 FIVE UAVS IN FLIGHT

For the scenario that involves 5 vehicles, the connection diagram is illustrated in Figure 5.2. It can be seen that vehicle 5 serves as the leader and does not follow any other vehicle. Vehicle 4 follows vehicle 5, vehicle 3 follows vehicle 4, vehicle 2 follows vehicle 3, and vehicle 1 follows vehicle 2. The initial positions for the case of 5 vehicles are observed in Table 5.2. The connectivity matrix A for the case involving 5 vehicles is presented below equation (5.6).

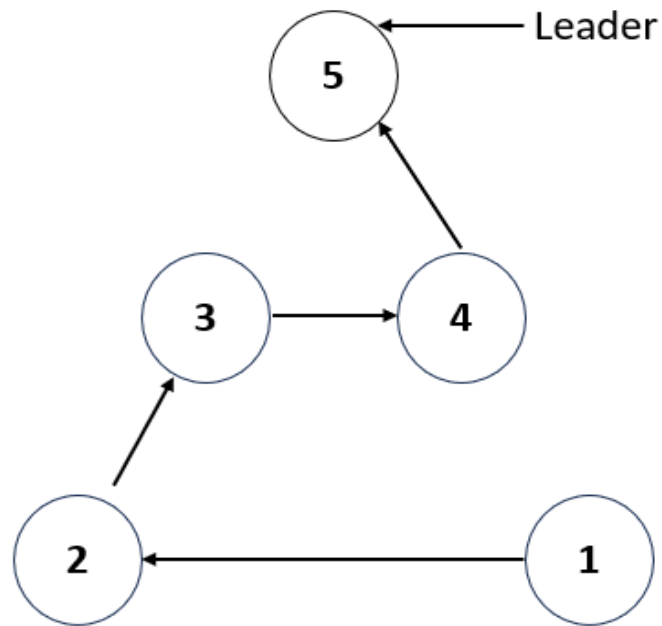


Figure 5.2: Graph of Connections for 5 vehicles.

Table 5.2: Initial positions of 5 vehicles

	$x(m)$	$y(m)$	$z(m)$
UAV1	2	2	2
UAV2	4	4	4
UAV3	6	6	6
UAV4	8	8	8
UAV5	10	10	10

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (5.6)$$

The neighbor matrix N_i for the scenario involving 5 vehicles is presented in Equation (5.7). For the scenario with five vehicles, two different formations were proposed: a delta formation, as depicted in Figure 5.3, and a line formation, as illustrated in Figure 5.4.

$$N_i = \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \\ 1 \end{bmatrix} \quad (5.7)$$

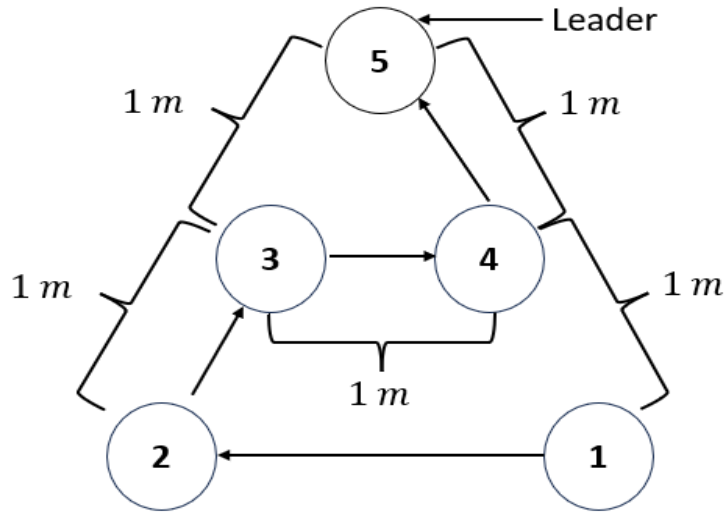


Figure 5.3: Formation with 5 vehicles for delta.

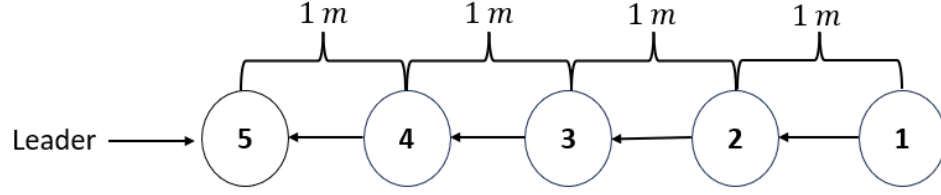


Figure 5.4: Formation with 5 vehicles for line.

The matrices Δ_i and Δ_j represent the offsets for delta formation with 5 UAVs, as presented in Equations (5.8) and (5.9). The matrices Δ_i and Δ_j represent the offsets for the formation of the line with 5 UAVs, as presented in Equation (5.10) and Equation (5.11).

$$\Delta_i = \begin{bmatrix} 2 & -0.52 & -1 & 0.52 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (5.8)$$

$$\Delta_j = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0.86 & 0 & 0.86 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (5.9)$$

$$\Delta_i = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (5.10)$$

$$\Delta_j = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (5.11)$$

5.3 CONSENSUS WITH TRACKING TRAJECTORY

To follow the formation along the guide trajectory, Equation (5.12) was used. It is important to note that the guide trajectory refers to the trajectory generated by the genetic trajectory generator. In this equation, $\dot{v}_0$, v_0 , and x_0 represent the acceleration, velocity, and position provided by the genetic trajectory generator, as described in the previous subchapter.

$$\begin{aligned}
 u_i &= \dot{v}_0 + \gamma k_v(v_0 - v_i) + k_p(x_0 + \Delta_i - x_i) \\
 &\quad + c \sum_{j \in N_i} a_{ij}((x_i - \Delta_j) - (x_j - \Delta_i)) + c\gamma \sum_{j \in N_i} a_{ij}(v_j - v_i) \\
 &= \dot{v}_0 + \gamma k_v(v_0 - v_i) + k_p(x_0 + \Delta_i - x_i) \\
 &\quad + \sum_{j \in N_i} ca_{ij}((x_i - \Delta_j) - (x_j - \Delta_i) + \gamma(v_j - v_i))
 \end{aligned} \tag{5.12}$$

First, an experiment was conducted to track the formation of three UAVs, using the same formation offsets shown in Figure 5.1. In this experiment, the behavior of the UAVs was evaluated because they followed a defined trajectory within an environment that contained obstacles. For the test involving the three UAVs, their initial positions and the locations of the obstacles are detailed in Table 5.3. This configuration allowed for an analysis of the UAVs ability to maintain formation and avoid obstacles, offering a comprehensive understanding of the system of performance within a controlled environment.

Table 5.3: Initial positions of 5 vehicles

	$x(m)$	$y(m)$	$z(m)$
UAV1	4	4	4
UAV2	3	3	3
UAV3	2	2	2
Obs	7.5	7.5	7.5

For the test with five UAVs, a formation tracking experiment was conducted using the delta formation offsets shown in Figure 5.2. In this experiment, the ability of the UAVs to maintain formation while navigating along a specific trajectory and avoiding obstacles was evaluated. The initial positions of the UAVs and the location of the obstacle for this test are detailed in Table 5.4.

Table 5.4: Initial positions of 5 vehicles

	$x(m)$	$y(m)$	$z(m)$
UAV1	2	2	2
UAV2	3	3	3
UAV3	4	4	4
UAV4	4	5	4
UAV5	5	4	4
Obs	7.5	7.5	7.5

The genetic trajectory discussed in previous chapters for this case had an initial position of (0,0,0) and a final point of (15,15,15). The velocity of the generated trajectory was 5 m/s, with an obstacle positioned at (7.5,7.5,7.5).

Furthermore, for the test with three UAVs, the gains used for trajectory tracking were as follows: $\gamma = 3$, $k_v = 1$ as the coefficient for velocity control and $k_p = 5$ as the coefficient for position control.

For tests with five UAVs, the gains used for trajectory tracking were: $\gamma = 3$, $k_v = 2$ as the coefficient for velocity control and $k_p = 5$ as the coefficient for position control.

CHAPTER 6

RESULTS AND DISCUSSION

Finally, this chapter presents the results obtained from the path planning algorithm applied to maze-like environments, evaluated in offline mode. This approach demonstrated the ability of the genetic algorithm to find feasible routes within complex search spaces. In addition, the algorithm was tested in scenarios with static and dynamic obstacles, highlighting its adaptability to changing environmental conditions.

The chapter also includes experimental results related to the consensus algorithm, in which multiple unmanned aerial vehicles were able to maintain predefined formations and follow reference trajectories in a coordinated manner. Both numerical simulations and real-world tests conducted on an experimental platform are analyzed, validating the effectiveness of the proposed distributed control scheme. In general, these results confirm the feasibility of integrating artificial intelligence techniques with collaborative control strategies for autonomous navigation applications.

6.1 VALIDATION OF GENETIC ALGORITHM FOR MAZE ENVIRONMENT

Two maze-type environments were prepared for six different cases to evaluate the genetic algorithm. In addition, numerical simulations and experimental validation are

performed to verify that the control algorithms can follow high-order polynomials.

6.1.1 SEQUENCE GENERATION FOR MAZE CASE 1 USING GENETIC ALGORITHM

In Case 1, a maze is set as shown in Figure 6.1. The green point marks the starting position at the coordinates $(0.3\text{ m}, 0.3\text{ m})$, while the red point indicates the target position at $(1.2\text{ m}, 3.9\text{ m})$. The blue dotted line represents the sequence of movements generated by the genetic algorithm, which successfully solves the case in four moves, the minimum number required to reach the target. The orange line illustrates the trajectory formed by the navigation points obtained from the sequence of movements; this trajectory is generated using a high-order polynomial. The solution was achieved using seven genes on the chromosome.

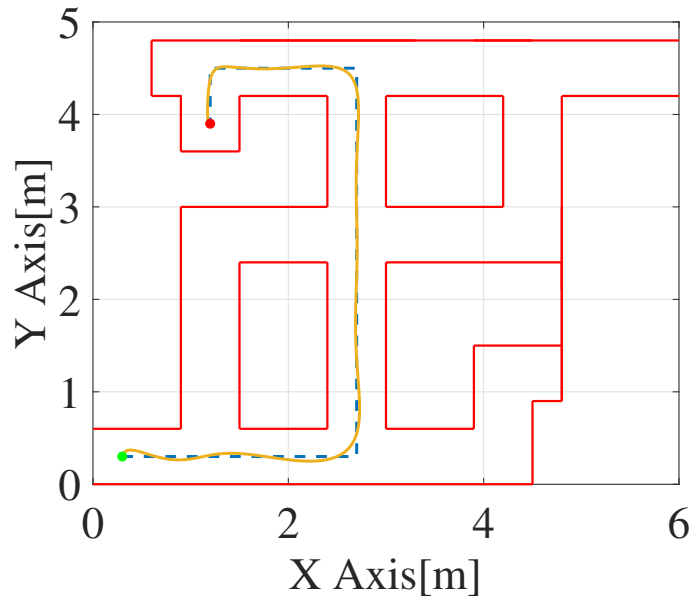


Figure 6.1: Path generated by genetic algorithm for case 1.

6.1.2 SEQUENCE GENERATION FOR MAZE CASE 2 USING GENETIC ALGORITHM

Case 2 is shown in Figure 6.2. Similarly to the previous case, the initial point is located at $(0.3\text{ m}, 0.3\text{ m})$ in the coordinates x, y , while the final point is at $(4.5\text{ m}, 3.9\text{ m})$. It can be seen that the sequence of movements generated by the genetic algorithm passes through three decision points and one intersection point. The algorithm solves this case using the minimum number of movements, which in this case is four. The trajectory represented by the orange line is generated using a high-order polynomial, while the navigation points along the sequence of movements are produced by the genetic algorithm. This case is solved using six genes on the chromosome.

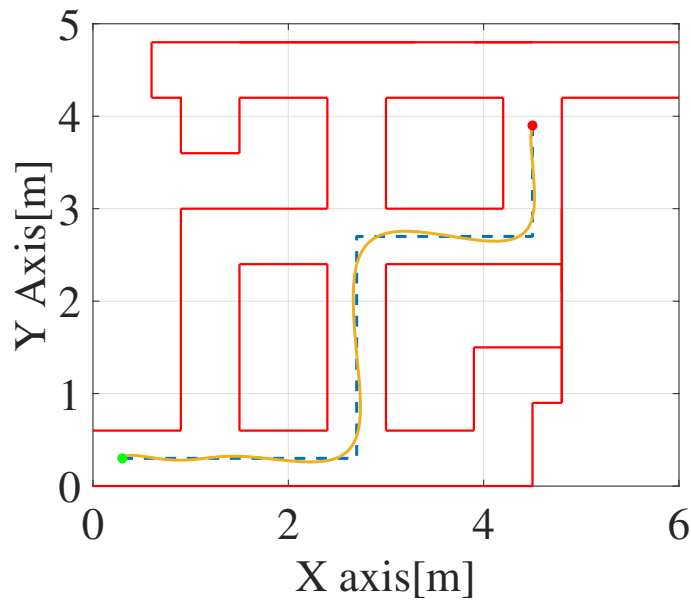


Figure 6.2: Path generated by genetic algorithm for Case 2.

6.1.3 SEQUENCE GENERATION FOR MAZE CASE 3 USING GENETIC ALGORITHM

Case 3 is shown in Figure 6.3, where a more complex maze-like environment is presented with additional decision points. The initial point, marked green, is located at $(0.10\text{ m}, 0.10\text{ m})$ in the x, y coordinates, while the final point, marked red, is at $(0.3\text{ m}, 2.6\text{ m})$. The path generated by the sequence of movements is represented by a blue dotted line.

The path can be seen to pass through five decision points, where the algorithm could have taken incorrect routes or made unnecessary movements. However, the algorithm identifies the optimal sequence of movements to reach the final point using the minimum number of steps, which in this case is six.

Once the sequence of movements is obtained, it is discretized to generate the navigation points, which are then used to construct the continuous trajectory using a high-order polynomial, represented by the orange line. This case is solved using nine genes on the chromosome.

6.1.4 SEQUENCE GENERATION FOR MAZE CASE 4 USING GENETIC ALGORITHM

In Case 4, the initial point is located at $(0.1\text{ m}, 0.1\text{ m})$ in the coordinates x, y , while the final point is at $(2.9\text{ m}, 2.6\text{ m})$. The final point was placed at a considerable distance from the initial point to evaluate the performance of the algorithm, as shown in Figure 6.4. The path generated by the sequence of movements is represented by a blue dotted line.

It can be observed that the algorithm must pass through several decision points, making it challenging to reach the final point and even more difficult to find the

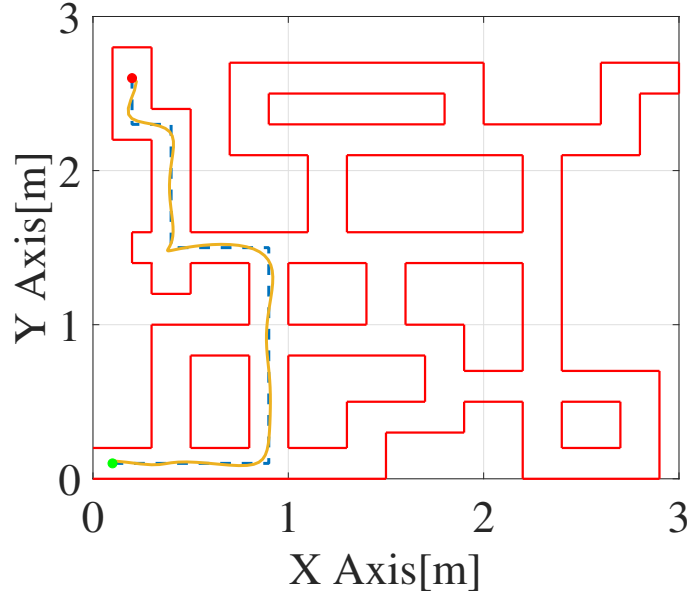


Figure 6.3: Path generated by genetic algorithm for case 3.

solution using the least number of moves. Despite this, the algorithm successfully determines the optimal trajectory using only seven moves, which is the minimum required for this case. This case is solved using twelve genes on the chromosome.

6.1.5 SEQUENCE GENERATION FOR MAZE CASE 5 USING GENETIC ALGORITHM

Case 5 is shown in Figure 6.5. In this case, the initial and final points are placed at opposite ends, with the initial point at $(2.8m, 0.1m)$ and the final point at $(0.3m, 2.6m)$ in the coordinates of x, y . The algorithm finds the solution using the minimum number of movements, which in this case is six.

As in the previous cases, the points obtained from the sequence of movements are used to compute a high-order polynomial that defines the trajectory. The blue dotted line represents the discrete path generated by the sequence of movements, while the orange line corresponds to the continuous trajectory obtained from the

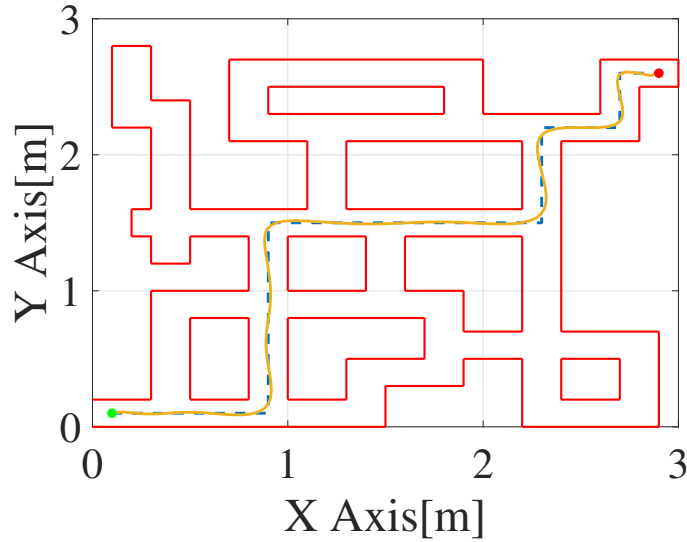


Figure 6.4: Path generated by genetic algorithm for case 4.

polynomial. This case is solved using nine genes on the chromosome.

6.1.6 SEQUENCE GENERATION FOR MAZE CASE 6 USING GENETIC ALGORITHM

Case 6 is shown in Figure 6.6. The initial point is located at $1.4m$ in x and $0.2m$ in y . The blue dotted line represents the sequence of movements generated by the genetic algorithm. Although the algorithm could have taken the path on the right to avoid moving downward, it evaluates and chooses the path on the left, even if it requires moving downward, since this path involves fewer steps and movements.

The orange line corresponds to the trajectory generated using a high-order polynomial and the navigation points obtained from the sequence of movements. This case is solved using ten genes on the chromosome.

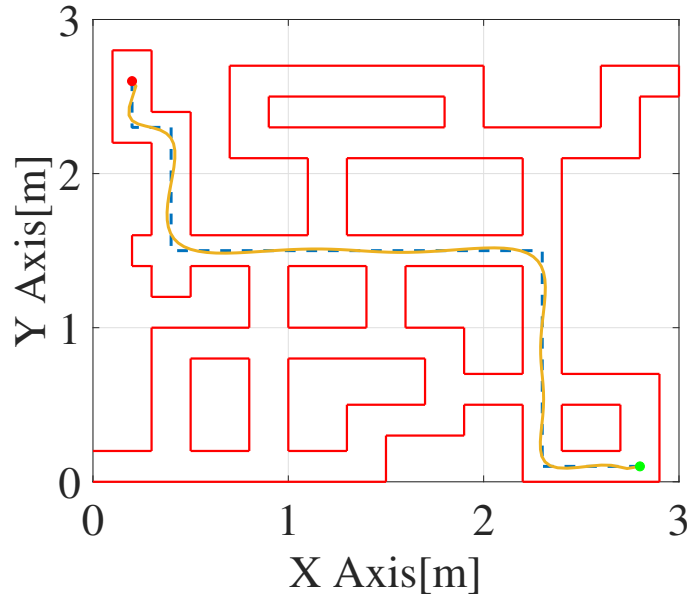


Figure 6.5: Path generated by genetic algorithm for case 5.

6.1.7 FITNESS VALUES FOR MAZE ENVIRONMENT

The fitness value graphs are shown in Figure 6.7, which illustrates the evolution of the algorithm until the solution is found. The Figure presents the fitness value graphs for Cases 1, 2, 3, 4, 5, and 6, which are labeled a), b), c), d), e), and f).

It can be observed that cases one and two, corresponding to maze 1 (Figure 6.7 a) and b)), are solved in fewer than 20 iterations since this maze has fewer decision points. In addition, most of the cases are solved in approximately 20 iterations. The only case that required more iterations was case e), corresponding to case 5, due to the higher number of decision points and the possible paths to reach the solution.

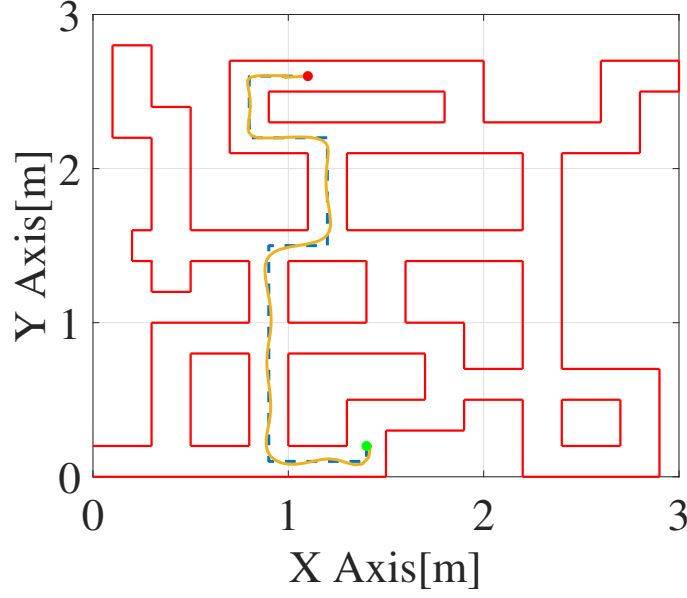


Figure 6.6: Path generated by genetic algorithm for case 6.

6.2 TRAJECTORY SIMULATION FOR THE MAZE ENVIRONMENT

In this section, simulations are performed to validate the proposed algorithm through a numerical evaluation of the trajectories generated using GA and high-order polynomials. The UAV model used corresponds to a mini-quadrotor that is part of a fleet of aerial vehicles developed at CIHA-FIME-UANL [48]. Table 6.1 presents the numerical values of the model parameters.

Parameter	Value	Parameter	Value
c_T	$9 \times 10^{-6} [N - s^2]$	d	$0.2 [m]$
c_Q	$3 \times 10^{-7} [N - m - s^2]$	J_{xx}	$15 \times 10^{-3} [kg - m^2]$
m	$0.3 [kg]$	J_{yy}	$15 \times 10^{-3} [kg - m^2]$
g	$9.81 [m/s^2]$	J_{zz}	$30 \times 10^{-3} [kg - m^2]$

Table 6.1: Quadrotor parameters.

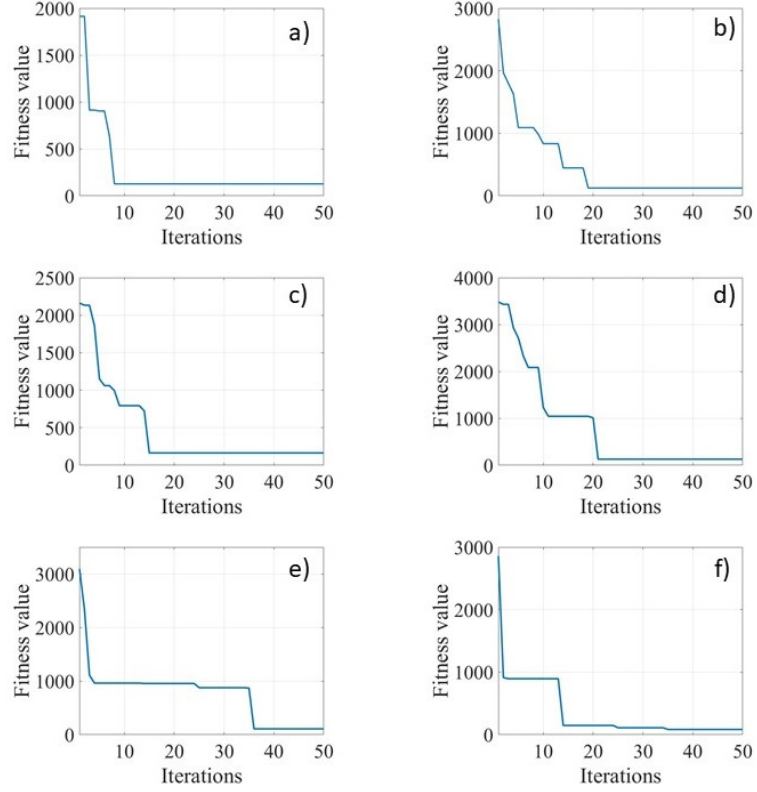


Figure 6.7: Fitness value graphs for cases 1 to 6.

The parameters are defined as follows: c_T and c_Q are the aerodynamic coefficients, and d represents the distance between the center of gravity of the quadrotor and the axis of rotation of the propellers.

6.2.1 SIMULATION OF PATHS

Figure 6.8 shows six trajectories ((a) to (f)) obtained using the genetic algorithm and high-order polynomials, which were executed with the simulated quadrotor model mentioned above. The comparison between the reference trajectory (red) and the trajectory followed by the UAV (blue) allows evaluating the performance of the control algorithm and the simulation model. It can be observed that the UAV was able to accurately follow most of the proposed trajectories; however, in Figure (f),

corresponding to case 6 of the maze, a small initial discrepancy is observed. This highlights the complexity involved in accurately following trajectories generated with high-order polynomials.

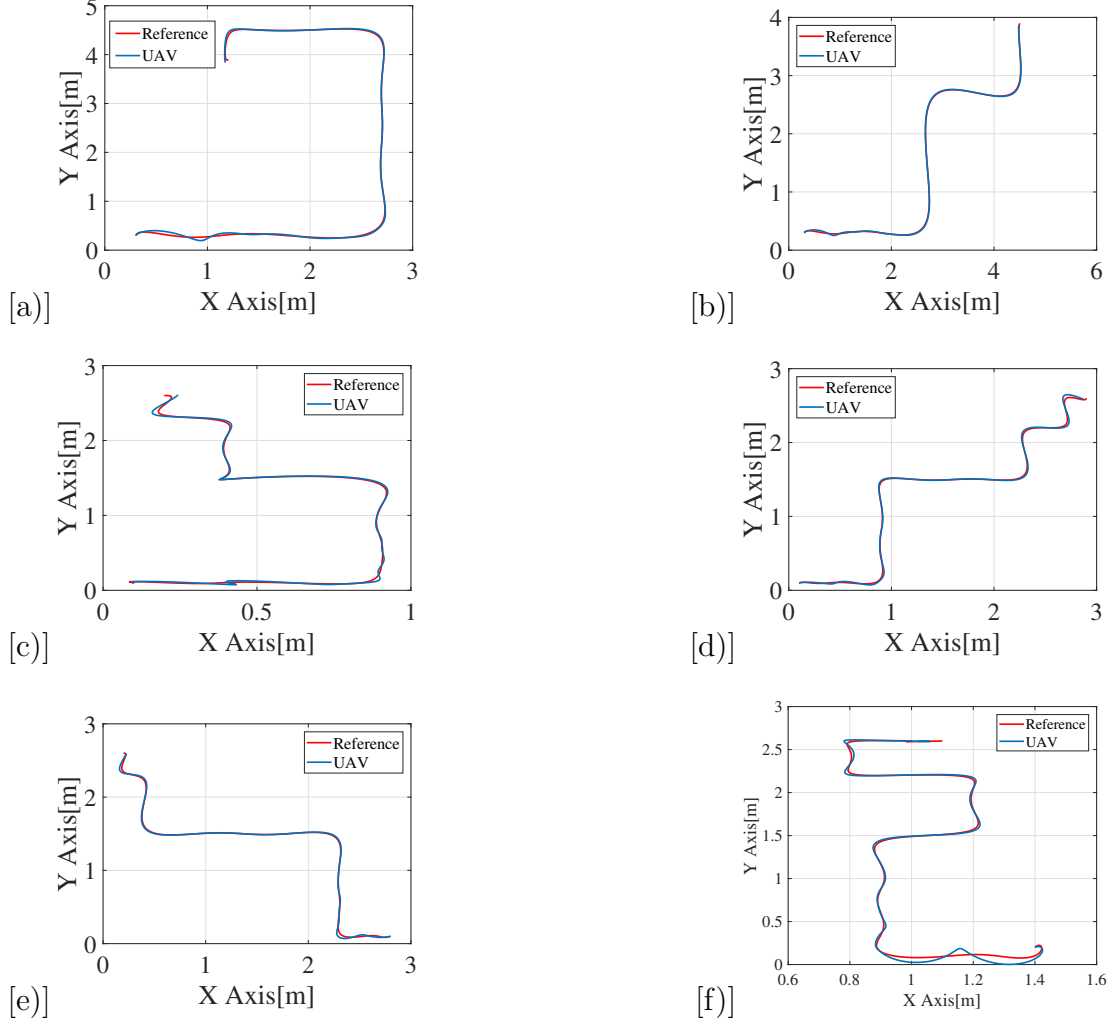


Figure 6.8: Trajectories in the XY -axis for different cases.

6.2.2 REAL-TIME MAZE EXPERIMENTS

Real-time experiments were conducted to demonstrate trajectory planning in a known environment. Two trajectories were defined offline and transmitted to the quadrotor using a time-dependent continuous polynomial function. A small-sized

quadrotor was used as the experimental platform, along with a motion capture system. The tests were carried out at the Navigation Laboratory of the Aeronautical Engineering Research and Innovation Center (CHIA) of the Autonomous University of Nuevo León (UANL).

The motion capture system consists of 16 VICON T40 cameras, which stream position and orientation data to a central computer at a frequency of 100 Hz, with an accuracy of 0.1 mm and 0.1° . The quadrotor used was designed and built for testing purposes in the same facility, as shown in Figure 6.9. Its basic characteristics include an "X" configuration, with 12.5 cm arms and a thrust capacity of 1.9 N per rotor.



Figure 6.9: Quadrotor used for experimental tests.

The trajectory graphs performed by the real quadrotor can be observed in Figure 6.10 and Figure 6.11. These correspond to Cases 1 and 2 solved by the genetic algorithm but implemented in real-time. The trajectories were generated using high-order polynomials based on the discretization of the sequence of movements generated by the genetic algorithm; these trajectories are represented by the red reference trajectories.

It can be observed that the trajectory performed by the quadrotor corresponds to the blue line and does not perfectly follow the reference trajectory. This may be

due to interference effects produced by the air interacting with the walls of the maze used for the test.

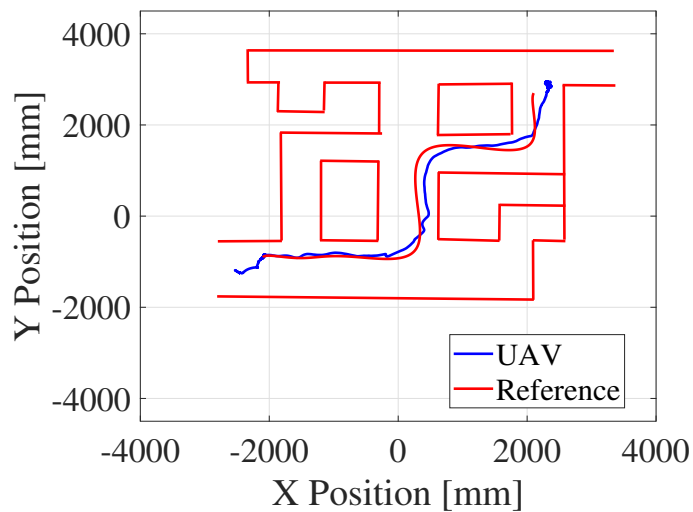


Figure 6.10: Trajectory in the XY -axis for first experimental scenario.

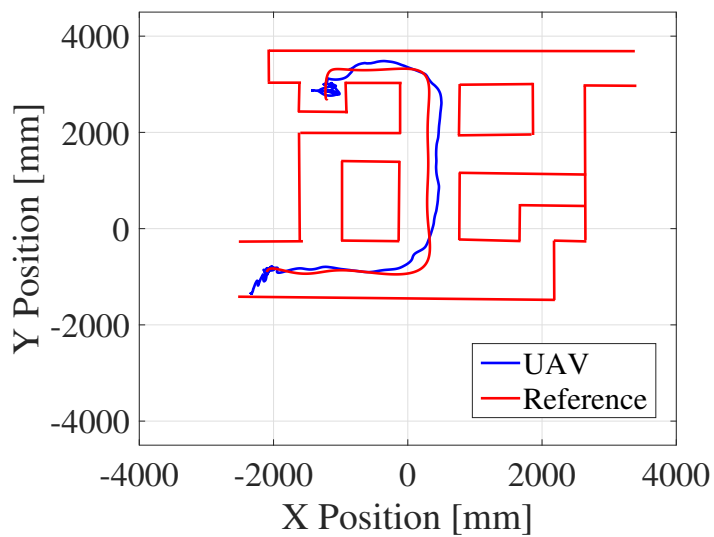


Figure 6.11: Trajectory in the XY -axis for second experimental scenario.

6.3 ALGORITHM CHARACTERIZATION IN STATIC ENVIRONMENT

A test scenario with four obstacles placed between the initial point $(0, 0, 0)$ and the final point $(15, 15, 15)$ is carried out in a static environment to evaluate the algorithm ability to find the local minimum in the fewest possible iterations. The specifications of the obstacles are provided in Table 6.2. This evaluation compares three different configurations of the GA algorithm. The first configuration, labeled Genetic Algorithm with Convergence Population (GaCPo), uses a convergence population along with a high mutation rate. The second configuration, Genetic Algorithm with high mutation (GaHMO), applies a high mutation rate, but does not include a convergence population. The third configuration, Genetic Algorithm with Low Mutation (GaLM), represents a conventional genetic algorithm with a low mutation rate. The mutation rates used for GaCPo, GaHM, and GaLM are 2%, 2%, and 0.2%, respectively.

These configurations are tested within a static environment. Figure 6.12 shows the minimum fitness values achieved, while Figure 6.13 presents the number of iterations required. It is important to note that a maximum of 30 iterations was set for this evaluation.

Among the 30 test cases executed for the three configurations, it is clear that the configuration that incorporates a convergence population (GaCPo) most frequently identified the lowest fitness value and also achieved lower fitness values compared to the other two configurations. The averages of the 30 trials, as shown in Figure 6.14, indicate that GaCPo achieved the lowest overall average fitness value.

In contrast, the average number of iterations for GaCPo was slightly higher than that of the configuration with a low mutation rate (GaLM). As expected, the GaLM configuration showed the best average in terms of convergence speed;

however, it also produced the highest fitness values. Meanwhile, the high mutation configuration without a convergence population (GaHM) performed the worst in terms of iteration count. This is due to the high mutation rate, which complicates the convergence process and can sometimes even lead to higher fitness values.

case static environment		
No. Obs	Position[x(m),y(m),z(m)]	Radio[m]
1	7.5,7.5,7.5	1.5
2	8.5,8.5,8.5	1.5
3	9.5,9.5,9.5	1.5
4	10.5,10.5,10.5	1.5

Table 6.2: Case static environment.

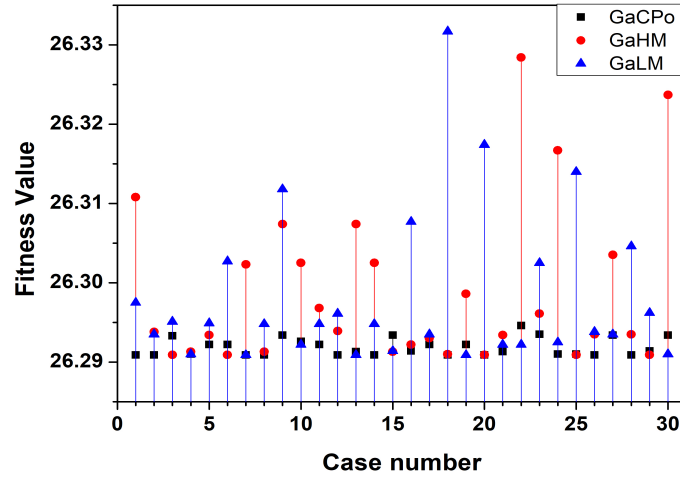


Figure 6.12: Fitness value for GaCPo,GaHM,GaLM.

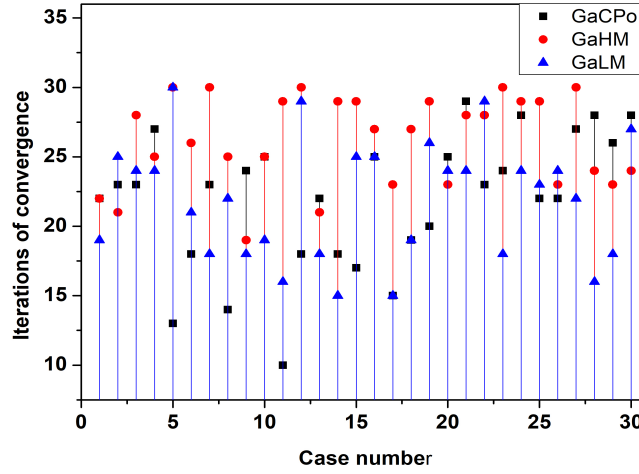


Figure 6.13: Iterations of convergence for GaCPo, GaHM, GaLM.

6.4 ALGORITHM EVALUATION IN DYNAMIC ENVIRONMENT.

To validate the algorithm, numerical simulations were performed with four obstacles, each having a radius of $1.5m$, within a 3D space measuring $15m \times 15m \times 15m$. The drone operates at a constant speed of $0.5m/s$, while the obstacles exhibit linear and angular movements. To visualize the path taken by the UAV, as well as how it avoids moving obstacles, a video demonstrating the simulations can be viewed at the following link: <https://youtu.be/0aLJaGszeD4>.

6.4.1 SIMULATION OF PATH PLANNING CASE 1

Figures 6.15 and 6.16 present the generated 3D path with static obstacles from two different points of view. Although the obstacles are static, the path is calculated in real-time as the UAV moves. In Figure 6.15, it is evident that the algorithm

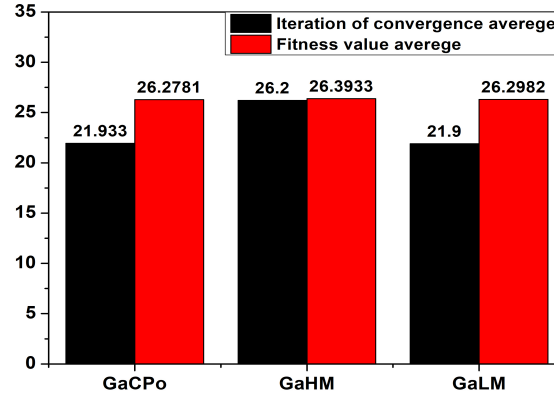


Figure 6.14: Average for GaCPo, GaHM, GaLM.

generated a path that passes under obstacles 1 and 2. The generated path successfully avoids all obstacles and, since they are static, no changes in the path are observed. Additionally, the path passes tangentially to obstacle 2. In scenarios where a minimum distance is sought in static environments, it is typical for the path to be tangential to at least one obstacle [49].

Figures 6.17, 6.18, and 6.19 show the movements on the axes X , Y , and Z , along with the comparison between the actual path and the desired path. The actual path is executed by the simulated model and control system, while the desired path is generated by the GA and motion equations. Figure 6.20 illustrates the fitness value of the algorithm, providing insight into how the behavior of the path evolves as the UAV progresses. It is observed that the fitness value of the entire population experiences higher peaks, whereas the convergence population exhibits smaller peaks due to its smaller size and the presence of individuals with lower fitness values.

Between 20 and 30 seconds, a slight increase in the population fitness value is observed, caused by adjustments made to avoid obstacle 2. When the UAV is near an obstacle or the final destination, peaks in the fitness value tend to appear. These are due to the repopulation criterion, which introduces diversity into the population

to adjust the path.

After 30 seconds, with all obstacles avoided, there are no more barriers that affect the UAV route to the final point. As a result, the path distance becomes the only active criterion, and the fitness value gradually decreases to zero. This decline is also due to the total and convergence populations beginning to merge with individuals that satisfy the obstacle avoidance requirements.

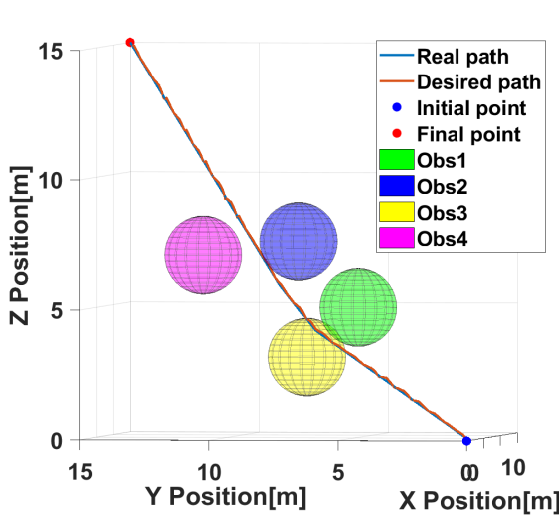


Figure 6.15: Simulation: 3D path static obstacles case 1 view 1.

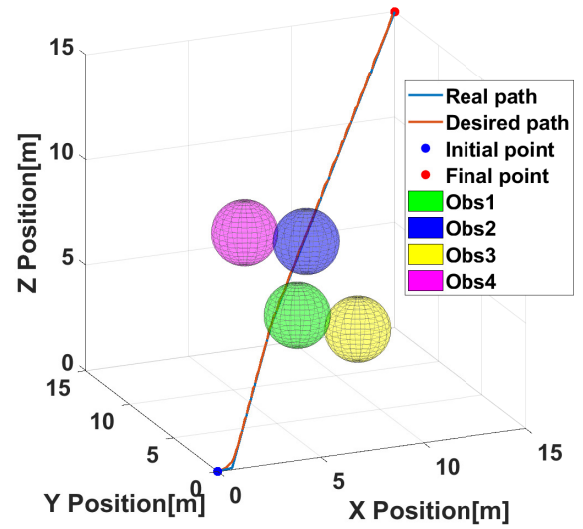


Figure 6.16: Simulation: 3D path static obstacles case 1 view 2.

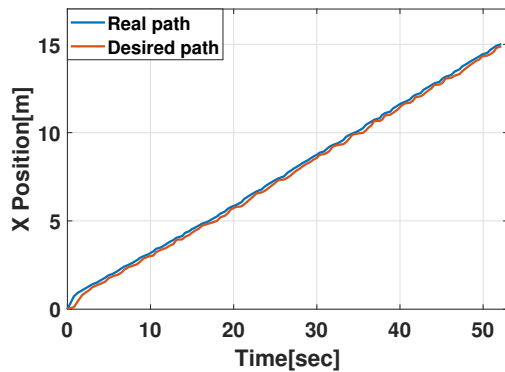


Figure 6.17: Simulation: single axis X case 1.

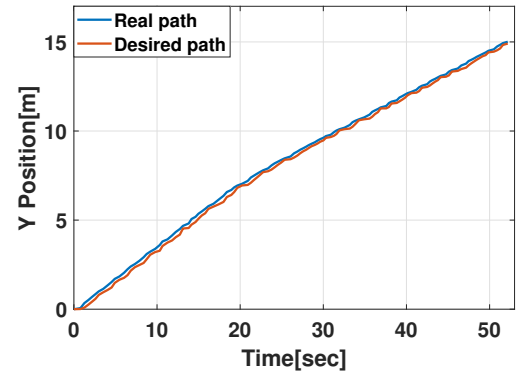


Figure 6.18: Simulation: single axis Y case 1.

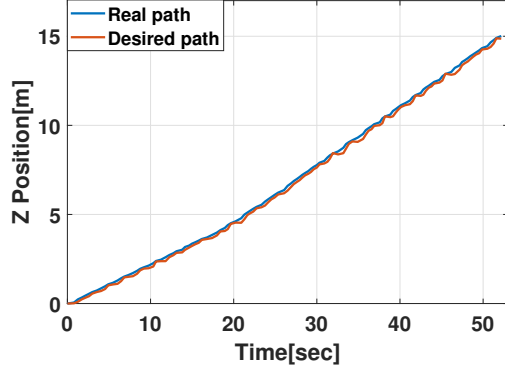


Figure 6.19: Simulation: single axis Z case
1.

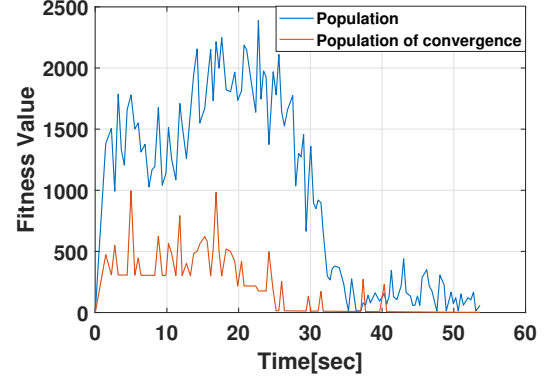


Figure 6.20: Simulation: Fitness value
static case 1.

6.4.2 SIMULATION OF PATH PLANNING CASE 2

Figures 6.21 and 6.22 present the generated 3D path for a scenario where obstacles 1 and 2 are in constant motion. Obstacle 1 follows an ascending circular trajectory, while obstacle 2 moves in an ascending linear path. It can be seen that obstacle 1 is the first to intersect with the UAV's path, prompting the algorithm to make a slight adjustment in the initial seconds to avoid a collision, as illustrated in Figures 6.23, 6.24, and 6.25. This adjustment is reflected in the fitness value, as shown in Figure 6.26, where a high peak appears during the first few seconds. This peak is even more pronounced in the convergence population due to its interaction with obstacle 1; the fitness value then decreases once the obstacle is avoided.

After successfully avoiding obstacle 1, obstacle 2 enters the path of the UAVs. Consequently, the algorithm recalculates and generates a path that crosses obstacle 2, allowing the UAV a clear path to reach the final destination. Figure 6.27 shows that, after 30 seconds, the fitness value drops, indicating that there are no more obstacles along the route to the final point. It is also notable that between 20 and 30 seconds, the fitness value of the convergence population exhibits significant peaks, which can be attributed to the repopulation criterion as the UAV approaches obstacle 2. In the movement axes, the path changes are evident between seconds 20

and 30, where there are adjustments in both the X and Z axes to successfully avoid obstacle 2.

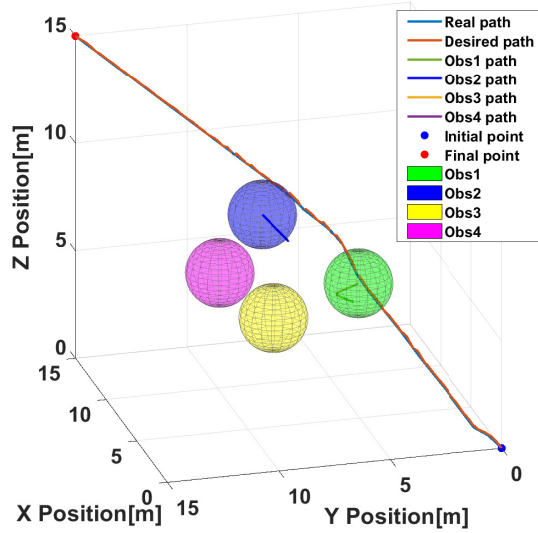


Figure 6.21: Simulations: 3D path dynamics obstacles case 2 view 1.

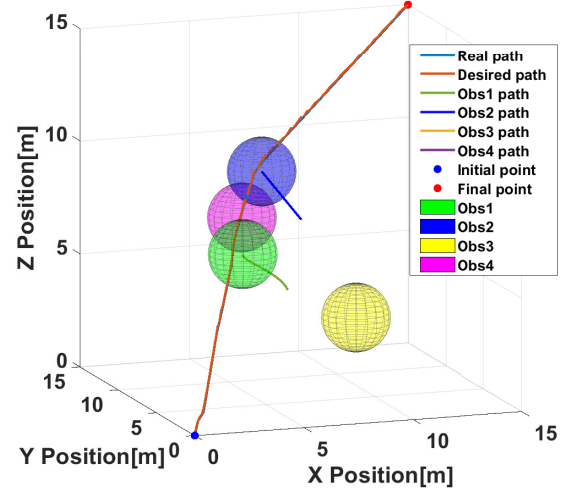


Figure 6.22: Simulations: 3D path dynamics obstacles case 2 view 2.

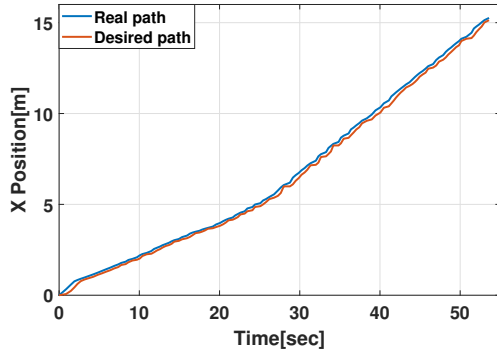


Figure 6.23: Simulation: single axis X case 2.

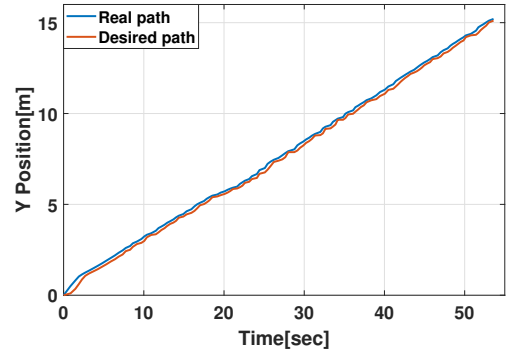


Figure 6.24: Simulation: single axis Y case 2.

6.4.3 SIMULATION OF PATH PLANNING CASE 3

In case 3, three obstacles are in motion while one remains static; obstacles 1 and 3 follow circular movements, obstacle 2 moves linearly upward, and obstacle 4 remains

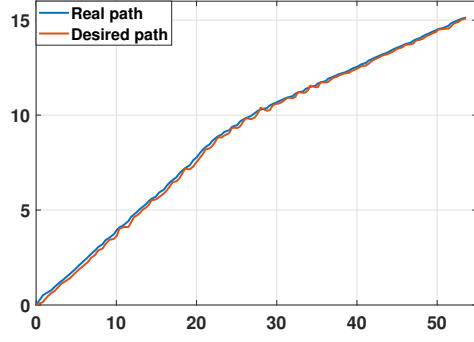


Figure 6.25: Simulation: single axis Z case 2.

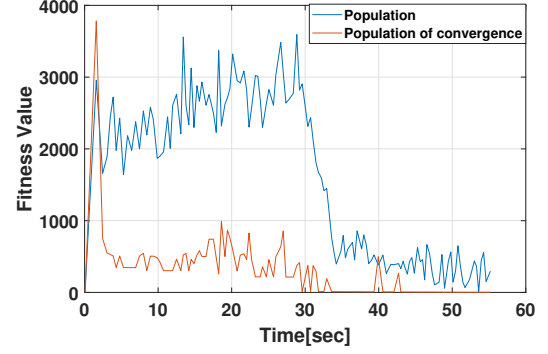


Figure 6.26: Simulation: fitness value dynamic case 2.

stationary. The generated 3D path is presented from two different perspectives in Figures 6.27 and 6.28. Obstacle 3 is observed to block the right side, forcing the path to be generated through the center. However, obstacle 1 then crosses the path, resulting in a change around the 20-second mark to avoid it. This adjustment is shown in the movement axes in Figures 6.29, 6.31, and 6.32, and is also reflected in the fitness value, where a significant peak occurs around 20 seconds. This peak affects not only the total population but also the convergence population, indicating that some individuals in the convergence population become inefficient but are quickly eliminated by genetic operators in subsequent iterations.

After avoiding obstacle 1, the path encounters obstacles 2 and 4 simultaneously. Consequently, the algorithm generates a path that passes below obstacle 2 and above obstacle 4. This adjustment is reflected in the Y and Z movement axes around the 30-second mark and is also seen in the fitness value, where there is a noticeable increase. Once obstacles 2 and 4 are avoided, all obstacles have been successfully bypassed, leading to a decrease in the fitness value to zero. The adjustments made to avoid all obstacles are most apparent in the Z-axis movement, with two distinct changes in altitude before reaching the final point. These changes are more significant than the adjustments seen in the X- and Y-axes.

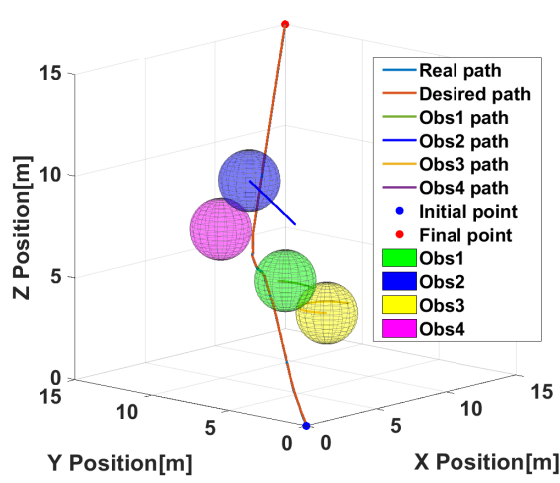


Figure 6.27: Simulation: 3D path dynamics obstacles case 3 view 1.

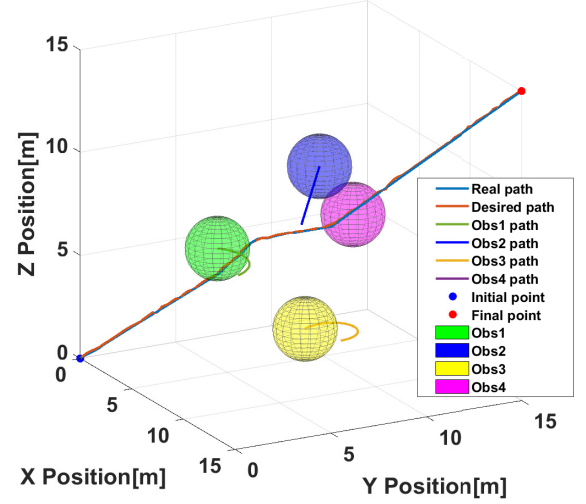


Figure 6.28: Simulation: 3D path dynamics obstacles case 3 view 2.

6.4.4 SIMULATION OF PATH PLANNING CASE 2

In case 4, all four obstacles are in motion. The generated 3D path is illustrated in Figures 6.33 and 6.34, where it is evident that most obstacles are positioned toward the right side, prompting the algorithm to generate a path to the left and above. Initially, the algorithm successfully creates a path that avoids all obstacles, but as obstacle 2 moves into the path, the algorithm adjusts to avoid it. This change is most prominently reflected in the graph of movement of the X-axis between seconds 20 and 30, as shown in Figure 6.35. In the figure of the fitness value graph 6.38, a noticeable peak appears between seconds 20 and 30, indicating that obstacle 2 unexpectedly entered the path. Once the UAV successfully avoids obstacle 2 and all other obstacles, the fitness value decreases after the 30-second mark, signifying that an obstacle-free path has been found toward the final destination.

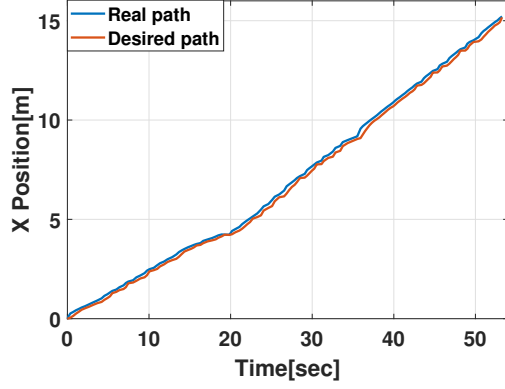


Figure 6.29: Simulation: single axis X case
3.

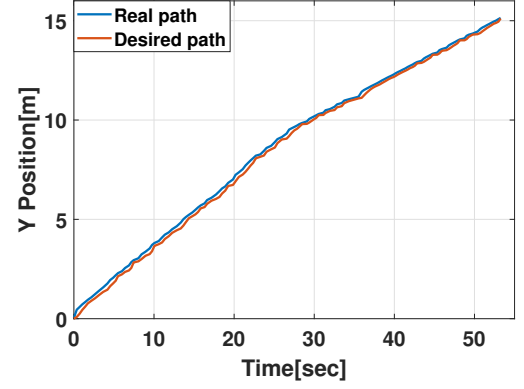


Figure 6.30: Simulation: single axis Y case
3.

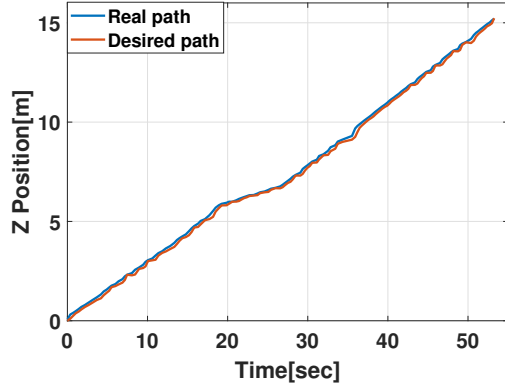


Figure 6.31: Simulation: single axis Z case
3.

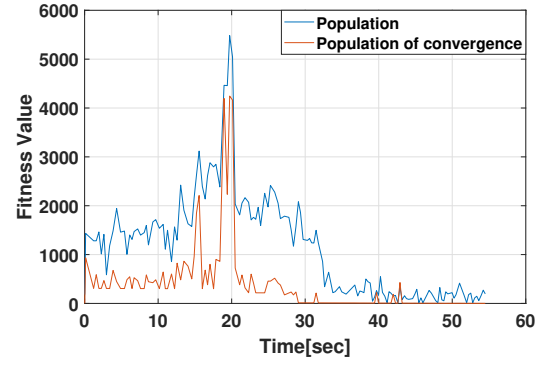


Figure 6.32: Simulation: fitness value dynamic case 3

6.5 EXPERIMENTAL RESULTS

The experimental tests were carried out within a space measuring $4m \times 4m \times 4m$, starting from an initial point at $(-2m, -2m, 0.5m)$ and ending at a final point of $(1.8m, 1.8m, 1m)$. The UAV maintained a constant velocity of $0.5m/s$ throughout the tests. The obstacles had a radius of $0.7m$. A video showcasing the experiments can be viewed at the following link: <https://youtu.be/D-zrkZpAFoM&t=22s>.

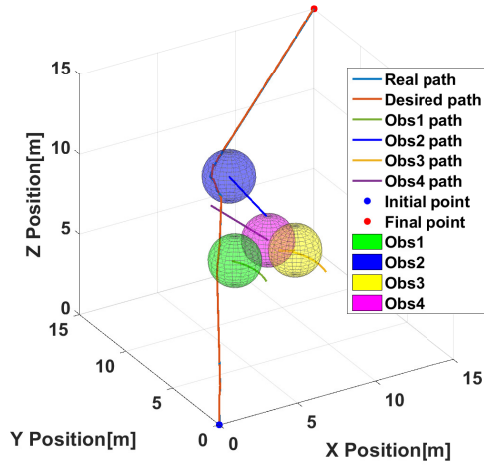


Figure 6.33: Simulation: 3D path dynamics obstacles case 4 view 1.

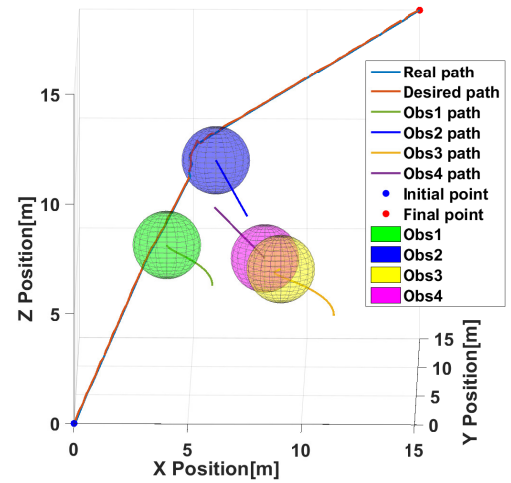


Figure 6.34: Simulation: 3D path dynamics obstacles case 4 view 1.

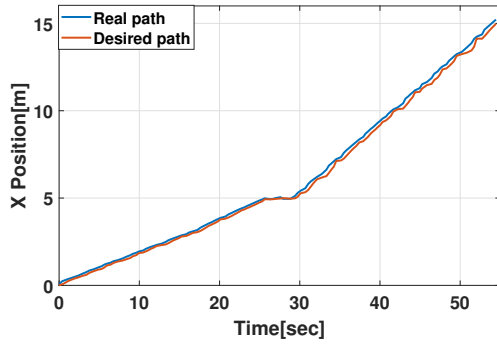


Figure 6.35: Simulation: single axis X case 4.

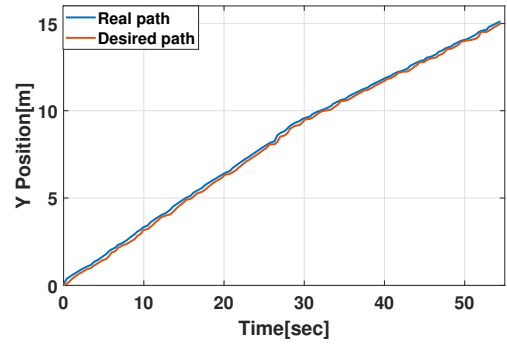


Figure 6.36: Simulation: single axis Y case 4.

6.5.1 EXPERIMENTAL PATH PLANNING CASE 1

In Case 1, a scenario with static obstacles is analyzed. Figures 6.39 and 6.40 show the generated 3D path. This path is characterized by two distinct movements, as illustrated in Figures 6.41, 6.42, and 6.43, which show the movement along each axis.

The algorithm successfully identifies a path, with the first change occurring around 2 seconds and a second adjustment taking place near 6 seconds. The fitness value graph, shown in Figure 6.44, shows a decrease of approximately 3 seconds,

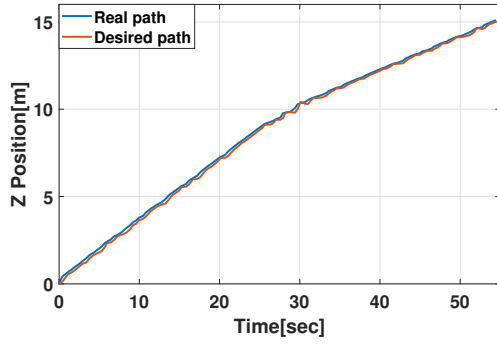


Figure 6.37: Simulation: single axis Z case
4.

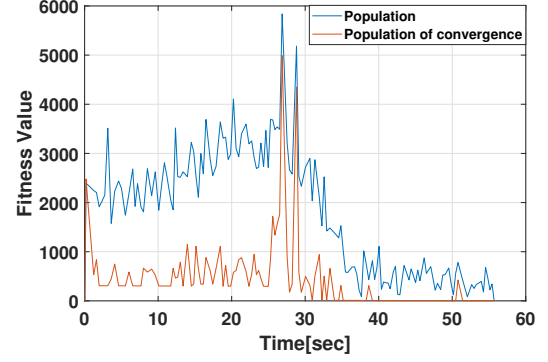


Figure 6.38: Simulation: fitness value dynamic case 4.

followed by a temporary increase between 4 and 6 seconds.

This increase suggests that before 2 seconds, the algorithm had already found a valid path that avoided all obstacles, resulting in an initial drop in the fitness value. After 3 seconds, a brief increase is observed due to the repopulation criterion, especially as the UAV moves close to obstacles.

The UAV then adjusts its trajectory to avoid obstacle 2 by flying to the left and above it. Eventually, it avoids all obstacles and proceeds directly to the final destination. This is reflected in the gradual decrease of the fitness value to zero after 8 seconds.

6.5.2 EXPERIMENTAL PATH PLANNING CASE 2

In case 2, obstacle 2 exhibits two movements, while obstacles 1 and 3 remain static, as shown in Figures 6.45 and 6.46. Initially, the UAV has an unobstructed path; however, after two seconds, obstacle 2 enters the trajectory, prompting a change in the UAV path. This adjustment is depicted in figures 6.47 and 6.48, which show the movement along the axes. The impact of this change is also evident in the fitness value shown in figure 6.48, where the fitness value of the convergence population

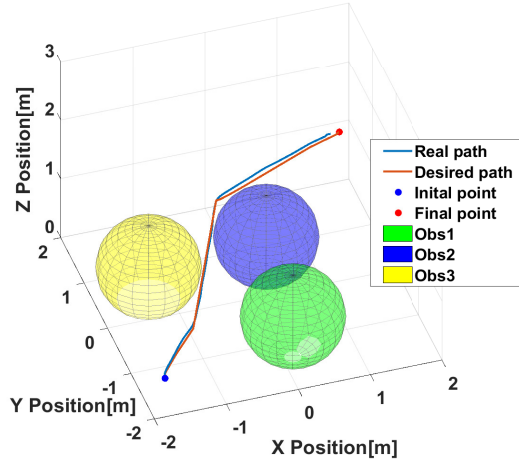


Figure 6.39: Experiment: 3D path dynamic obstacles case 1 view 1.

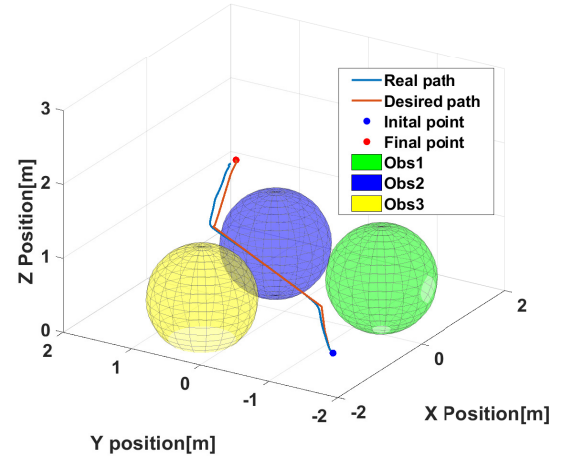


Figure 6.40: Experiment: 3D path static obstacles case 1 view 2.

remains near zero between 0 and 2 seconds but then increases as obstacle 2 intersects the path.

To avoid obstacle 2, the algorithm redirects the path above it, which is clearly shown in the second view of the path in Figure 6.46 and in the movement of the Z axis around the 2-second mark in Figure 6.48. Subsequently, obstacle 2 makes a second upward movement, leading the algorithm to adjust the path once more, this time avoiding the obstacle to the right and descending in height. This downward shift is also visible in the Z-axis graph around the 7-second mark, where the height decreases from $1.8m$ to the desired $1m$. A peak in the fitness value appears near 8 seconds when the obstacle intersects the path again. Once the UAV successfully avoids the obstacle, the fitness value steadily drops to almost zero after 10 seconds.

6.5.3 EXPERIMENTAL PATH PLANNING CASE 3

In case 3, two obstacles are in motion. The generated 3D path is depicted in Figures 6.51 and 6.52. Initially, the UAV has an unobstructed path; however, as obstacle 2

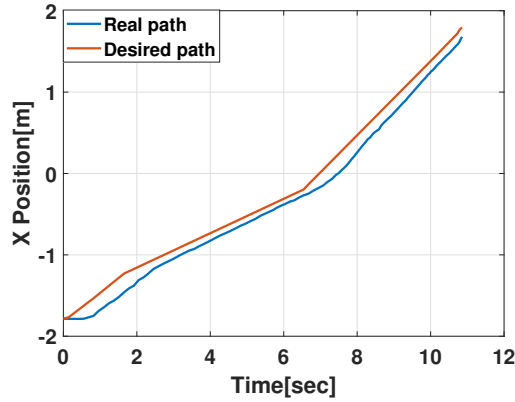


Figure 6.41: Experiment: single axis X case 1.

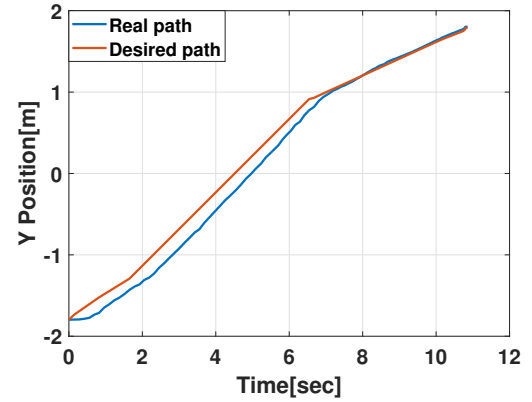


Figure 6.42: Experiment: single axis Y case 1.

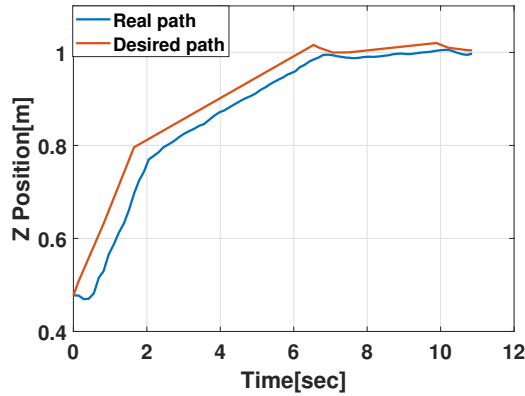


Figure 6.43: Experiment: single axis Z case 1.

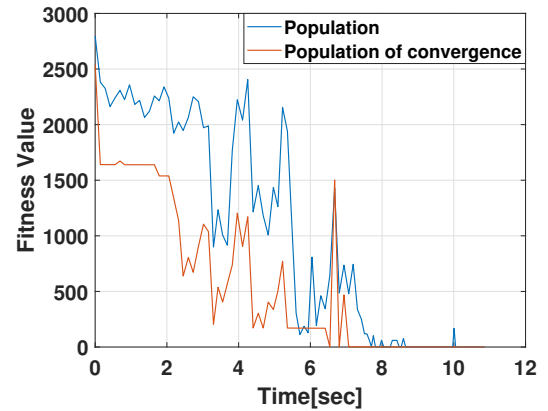


Figure 6.44: Experiment: fitness value dynamic case 1.

enters the trajectory, the algorithm adjusts the path to the left to avoid it. Subsequently, when obstacle 1 appears in the path, the algorithm shifts the path to the right to bypass it. The second view in figure 6.52 highlights that these adjustments are not confined to a single plane but involve significant changes in height as well.

The movement axes are illustrated in Figures 6.53, 6.54, and 6.55. In movement on the Z-axis, a drop in height is observed around the 6-second mark, followed by a rise to reach the desired altitude. The fitness value, shown in figure 6.56, indicates that the algorithm initially identifies a clear path, but around 2 seconds, when

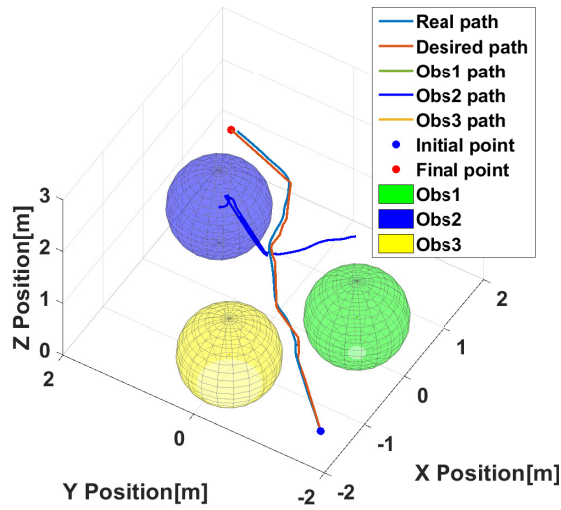


Figure 6.45: Experiment: 3D path dynamics obstacles case 2 view 1.

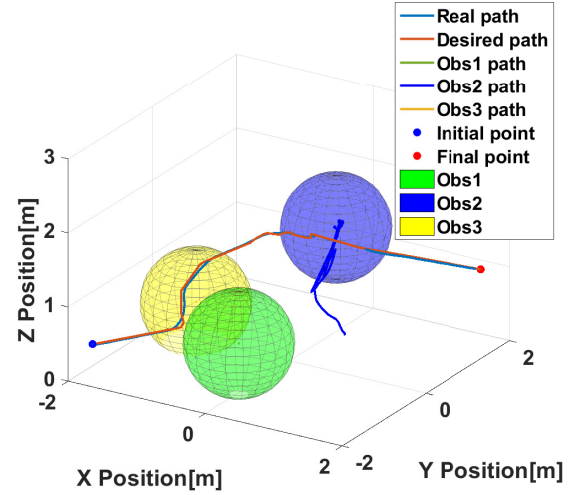


Figure 6.46: Experiment: 3D path dynamics obstacles case 2 view 2.

obstacle 2 obstructs the route, an immediate path adjustment occurs. Similarly, as obstacle 1 intersects the path at around 6 seconds, another adjustment is made, accompanied by a temporary increase in the fitness value. Ultimately, the algorithm finds an unobstructed path and the fitness value gradually decreases to zero. A small peak is noticeable near the 10-second mark, which might be due to the absence of an optimal waypoint that reached the final point until that moment.

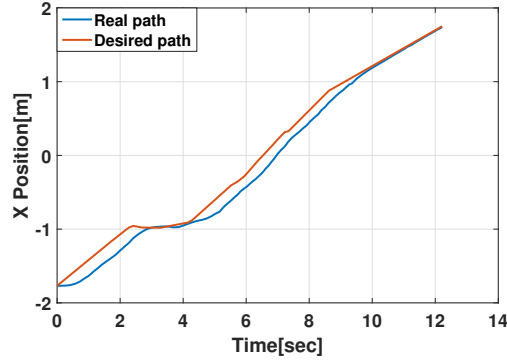


Figure 6.47: Experiment: single axis X components case 2.

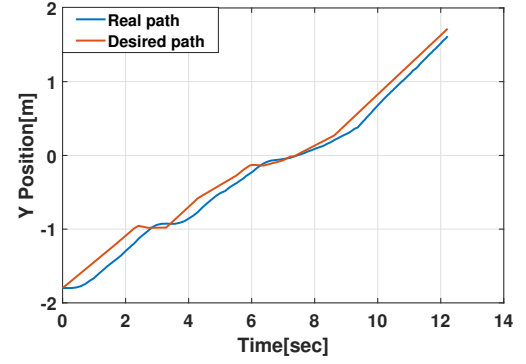


Figure 6.48: Experiment: single axis Y components case 2.

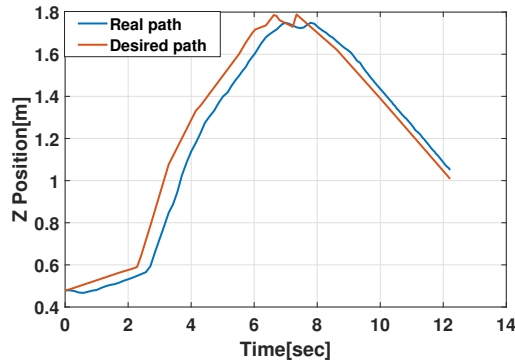


Figure 6.49: Experiment: single axis Z case 2.

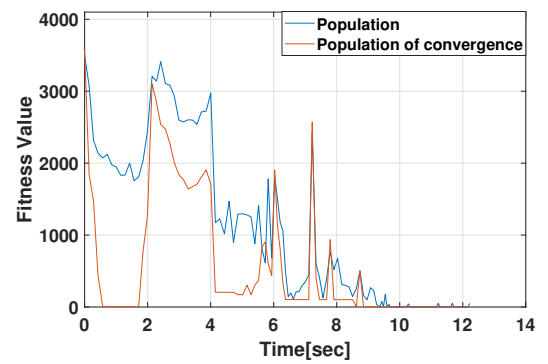


Figure 6.50: Experiment: fitness value dynamic case 2.

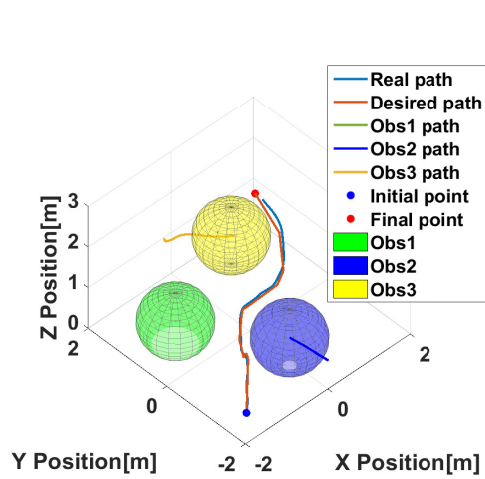


Figure 6.51: Experiment: 3D path dynamics obstacles case 3 view 1.

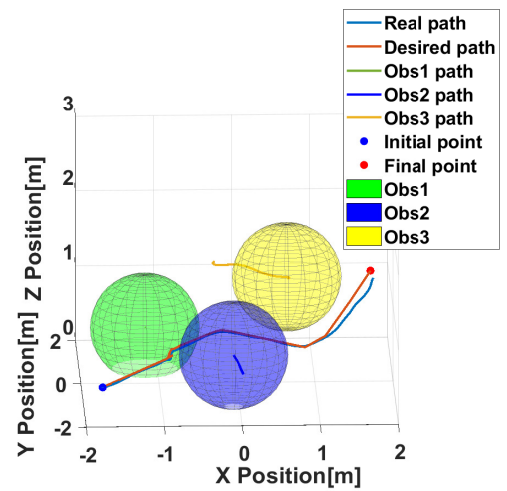


Figure 6.52: Experiment: 3D path dynamics obstacles case 3 view 2.

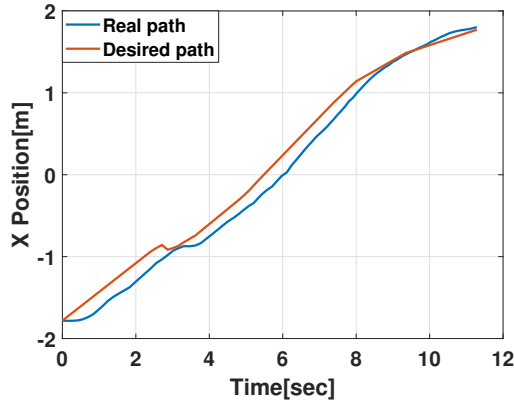


Figure 6.53: Experiment: single axis X case 3.

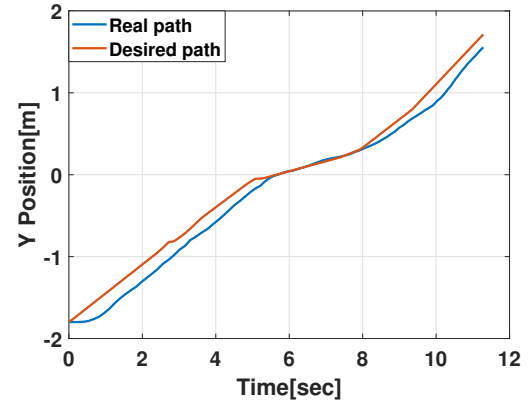


Figure 6.54: Experiment: single axis Y case 3.

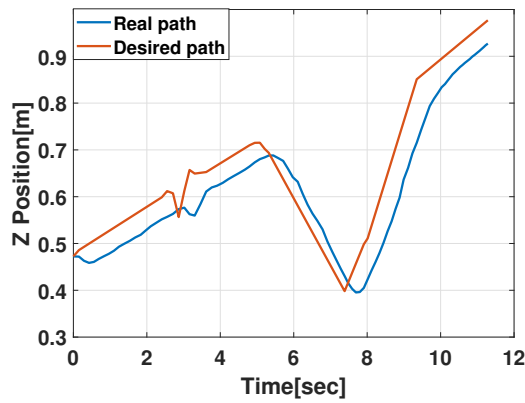


Figure 6.55: Experiment: single axis Z case 3.

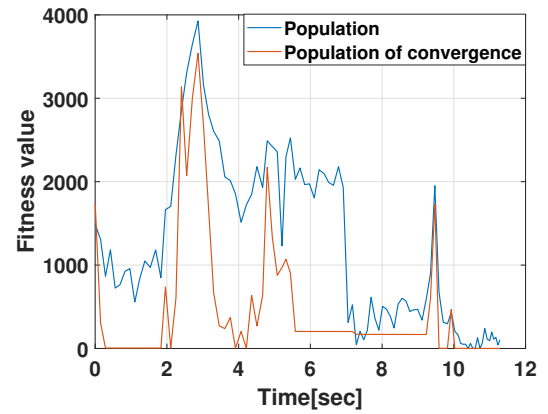


Figure 6.56: Experiment: Fitness value dynamic case 3.

6.5.4 EXPERIMENTAL PATH PLANNING CASE 4

In Case 4, three obstacles are in motion. The generated 3D path is shown in Figures 6.56 and 6.57, where it is clear that the path successfully avoids all obstacles. Figures 6.58, 6.59, and 6.60 further illustrate the complete movements of the path. The movement along the axes shows a change in the trajectory around the 2-second mark, which corresponds to obstacles 1 and 2 entering the UAV path. The UAV

responds by avoiding these obstacles from above, as observed in the movement along the Z-axis.

After bypassing obstacles 1 and 2, there is a brief decrease in the fitness value, indicating a momentary absence of obstacles along the path. However, the fitness value subsequently increases as obstacle 3 moves from the bottom to the top, requiring the UAV to adjust its path upward to avoid the collision. Around 10 seconds, the algorithm identifies a clear path without obstacles, and the path adjustments are reflected in the fitness value, as shown in figure 6.62.

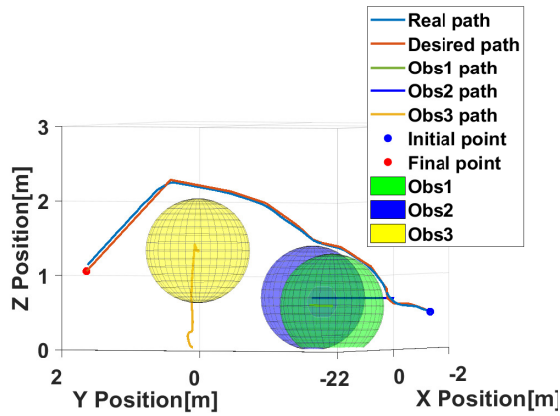


Figure 6.57: Experiment: 3D path dynamics obstacles case 4 view 1.

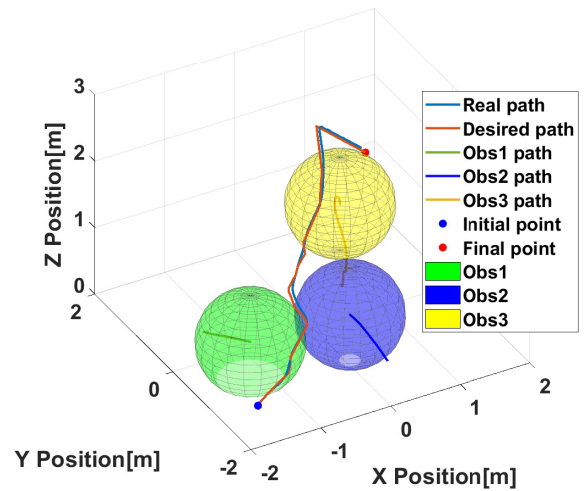


Figure 6.58: Experiment: 3D path dynamics obstacles case 4 view 2.

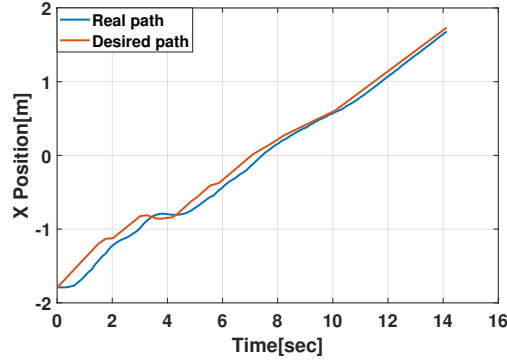


Figure 6.59: Experiment: single axis X case 4.

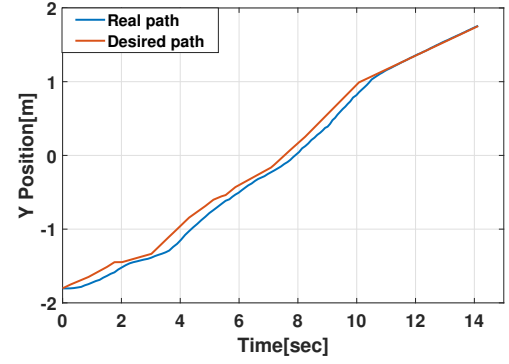


Figure 6.60: Experiment: single axis Y case 4.

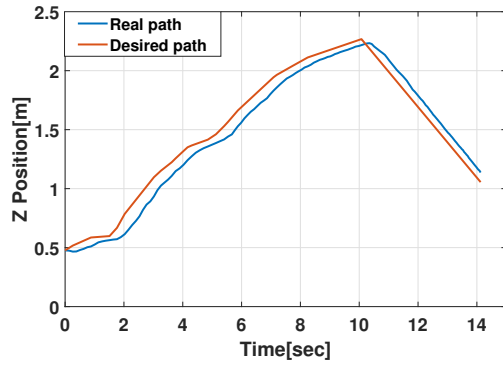


Figure 6.61: Experiment: single axis Z case 4.

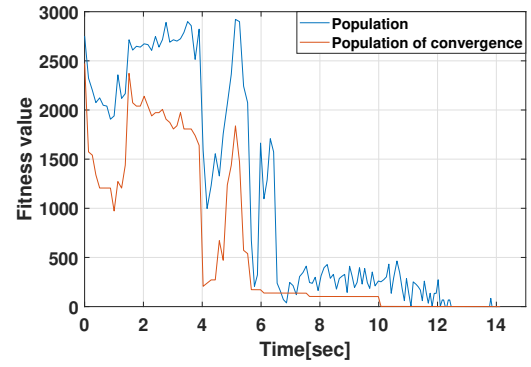


Figure 6.62: Experiment: fitness value dynamic case 4.

6.6 RESULTS OF CONSENSUS

6.6.1 FORMATION RESULTS WITH THREE UAVs

The results obtained for the three vehicles are presented below. In Figures 6.63, the movement of vehicles along the X , Y and Z axes is shown. These figures demonstrate how each vehicle accurately reaches the positions established in the formation.

Analyzing the trajectories followed by the vehicles reveals the effectiveness of the implemented control system in coordinating their movements. The convergence of vehicles to the desired positions underscores the ability of the algorithm to handle the dynamics of the system in a robust and efficient manner.

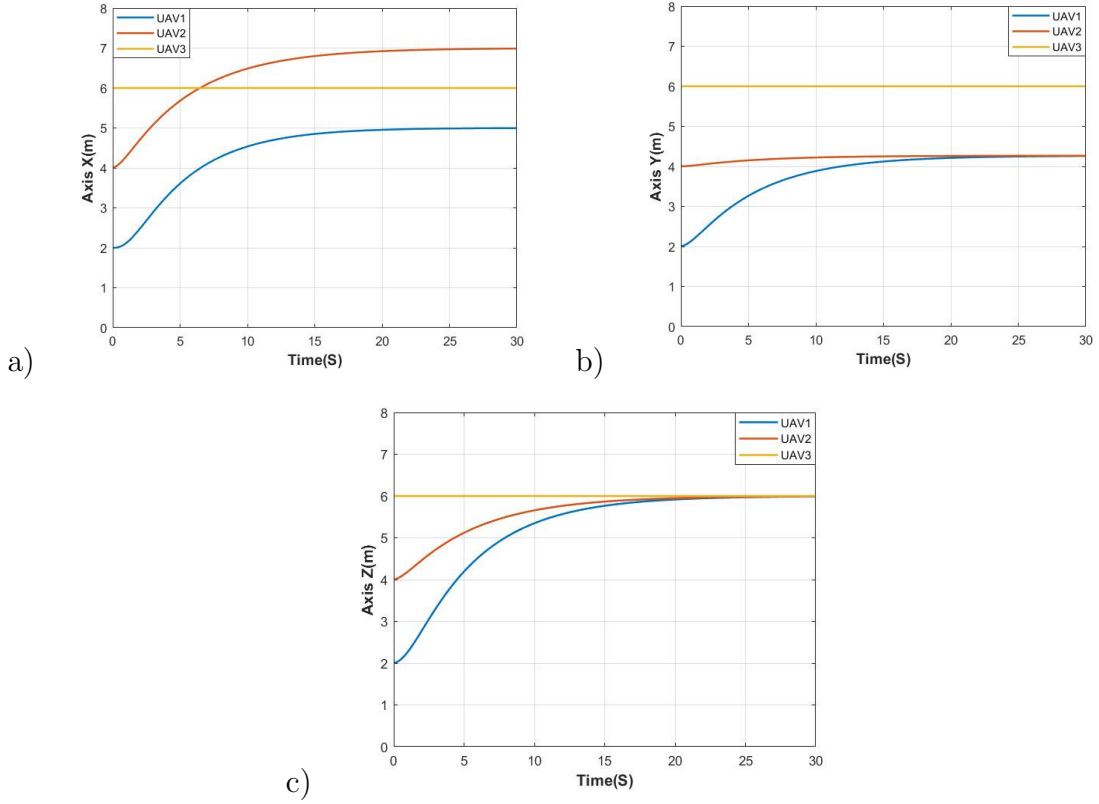


Figure 6.63: Axis of movement for 3 UAVs: a) axis of movement X, b) axis of movement Y, c) axis of movement Z.

The three-dimensional distances between the UAVs in the formation are shown in Figure 6.64. The figure presents the distances between UAV_1-UAV_2 , UAV_2-UAV_3 , and UAV_1-UAV_3 . It is evident that all distances converge to 2 m, indicating that the UAVs maintain the specified spacing for the desired formation. Figure 6.65 illustrates the convergence of distances along the Z axis, where all distances converge to zero, confirming that all UAVs are at the same altitude. Finally, Figure 6.66 provides a 3D perspective of the trajectory, showing the movements of UAV_1 and UAV_2 , while UAV_3 remains stationary, serving as the reference leader for the formation.

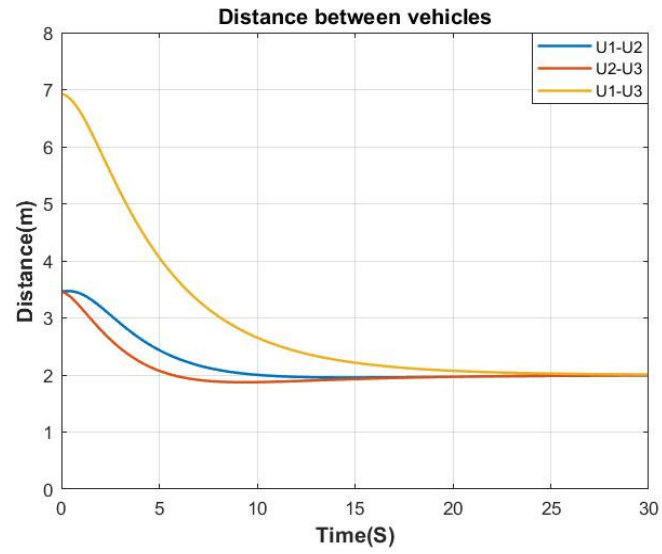


Figure 6.64: Distance between 3 UAVs for delta formation.

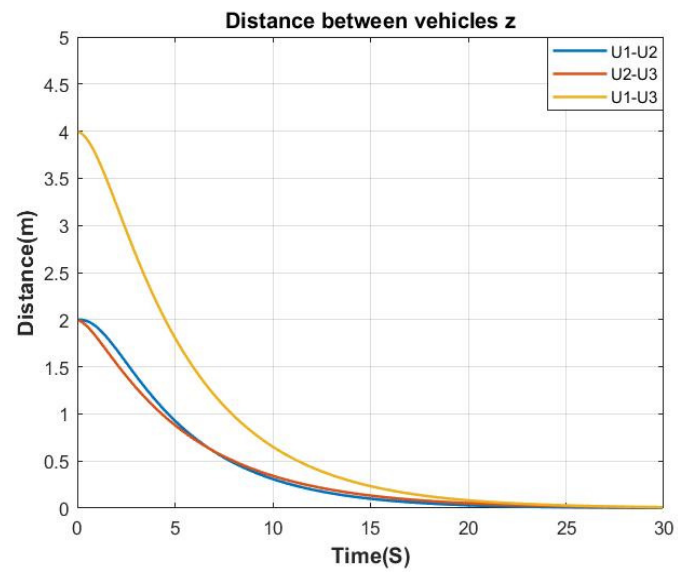


Figure 6.65: Distance between 3 UAVs for delta formation axis Z

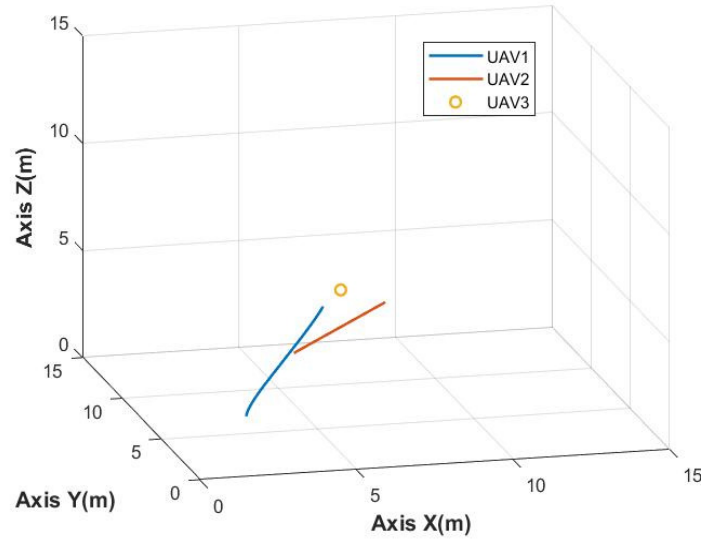


Figure 6.66: 3D Trajectory View of 3 UAVs.

6.6.2 FORMATION RESULTS WITH FIVE UAVS

The movement along the X , Y , and Z axes for the five UAVs is shown in Figures 6.67. These Figures demonstrate that the UAV positions converge to the desired formation positions. The graphs clearly illustrate how each UAV adjusts its trajectory to reach its specific target within the formation, showcasing the effectiveness of the implemented control algorithm.

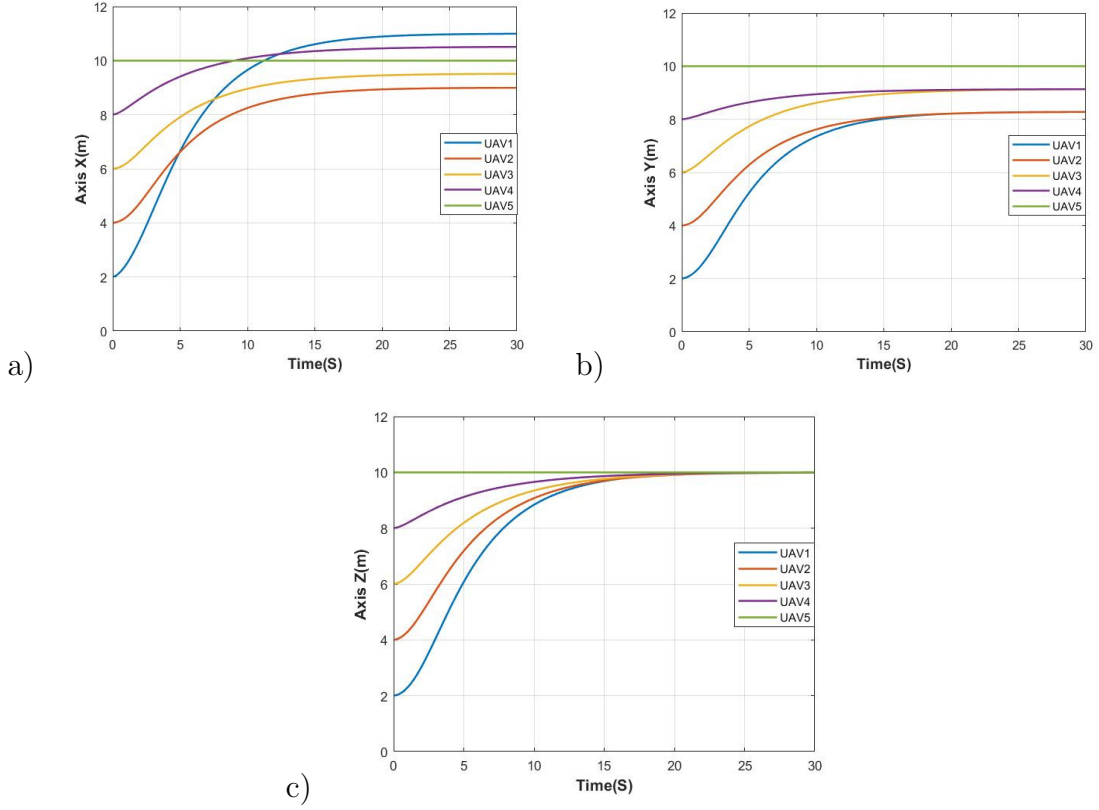


Figure 6.67: Axis of movement for 5 UAVs delta formation: a) axis of movement X, b) axis of movement Y, c) axis of movement Z.

In Figure 6.68, the distances between the UAVs in the delta formation of five UAVs are presented, as shown in Figure 5.3. The figure shows the distances with offsets of 1 m, indicating that all distances converge to 1 m, which shows that the control system successfully guides the UAVs to achieve the desired formation. Figure 6.69 displays the distances between the UAVs along the Z-axis, where it can be observed that all distances converge to zero, confirming that the UAVs navigate at the same altitude. Finally, the trajectory of vehicles in the 3D environment is shown in figure 6.70, where it is evident that UAV 5 remains stationary as it serves as the leader vehicle, with the other references moving relative to it.

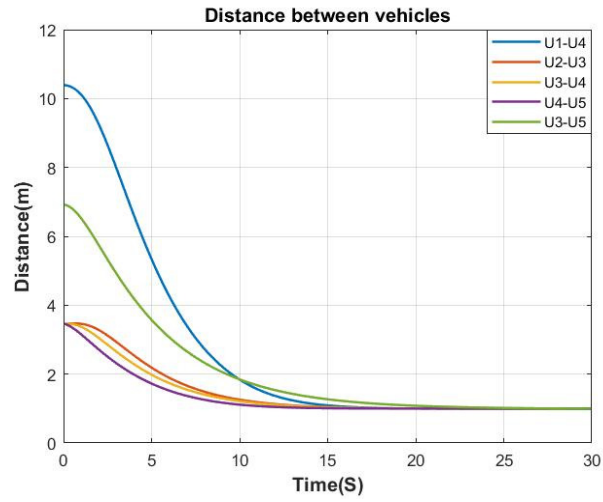


Figure 6.68: Distance between 5 UAVs for delta formation.

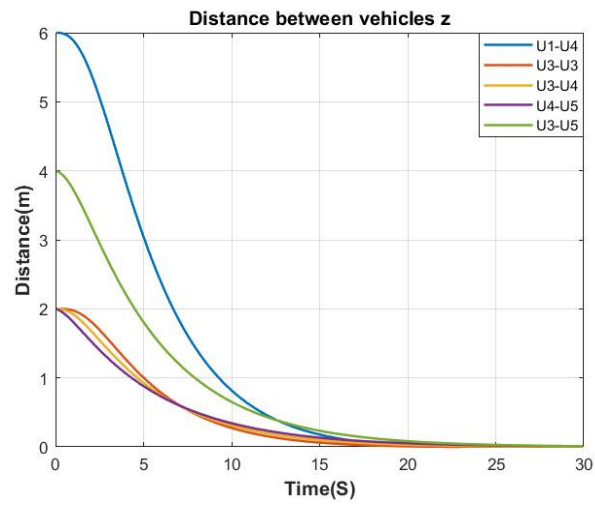


Figure 6.69: Distance between 5 UAVs for delta formation along the Z-axis.

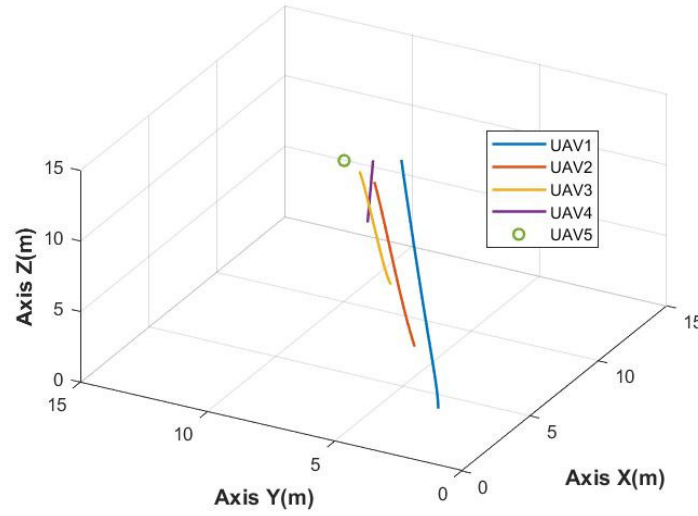


Figure 6.70: 3D Trajectory View of 5 UAVs delta formation.

6.6.3 FORMATION RESULTS WITH FIVE UAVS LINE-SHAPE

The movements along the X , Y , and Z axes for the five UAVs in a line formation are shown in Figure 6.71. It can be seen that all UAVs converge to their desired positions. Notably, on the Y -axis and Z -axis, they all converge to the same position, consistent with the configuration of the desired formation.

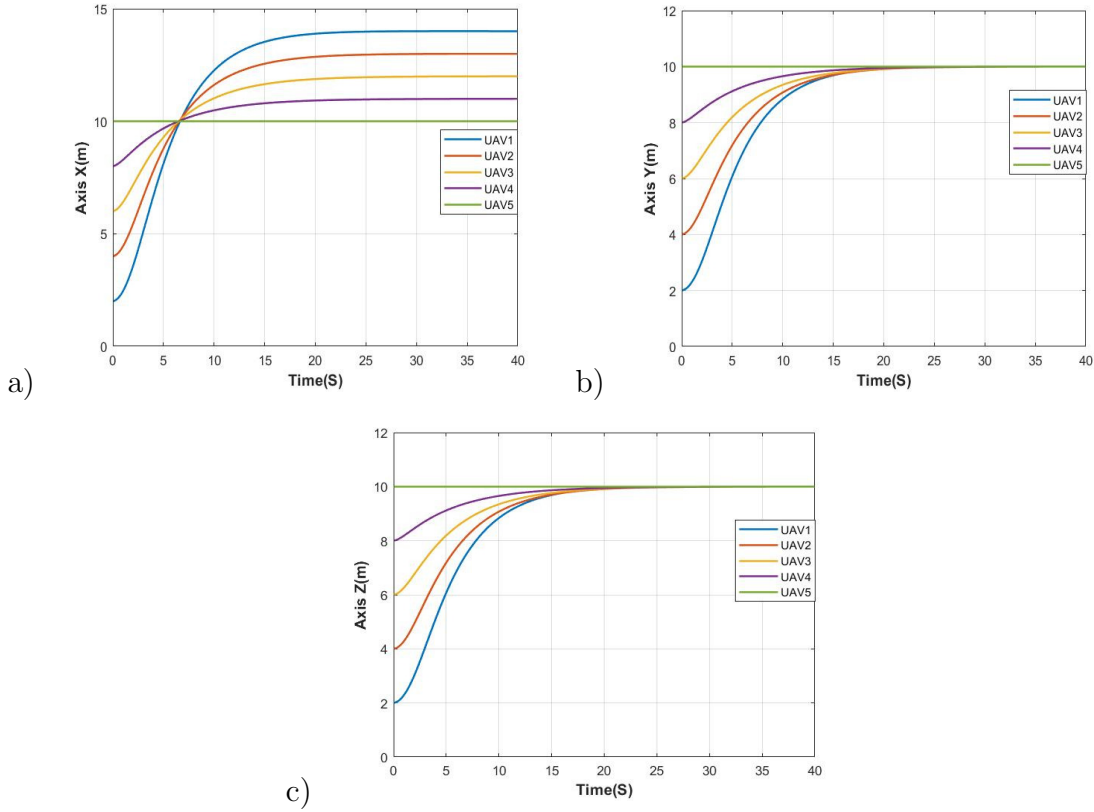


Figure 6.71: Axis of movement for 5 UAVs line formation: a) axis of movement X, b) axis of movement Y, c) axis of movement Z.

In figure 6.72, the distances between the vehicles are shown, where it can be observed that all distances converge to 1 m , confirming that they achieve straight line formation. The distances between the UAVs along the Z-axis are presented in figure 6.73, indicating that all distances converge to zero. Lastly, the 3D perspective of the movement is shown in Figure 6.74.

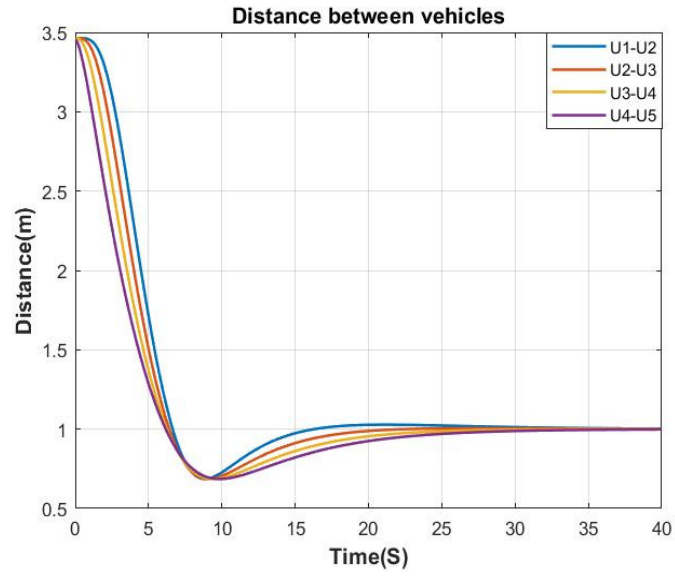


Figure 6.72: Distance between 5 UAVs for line formation.

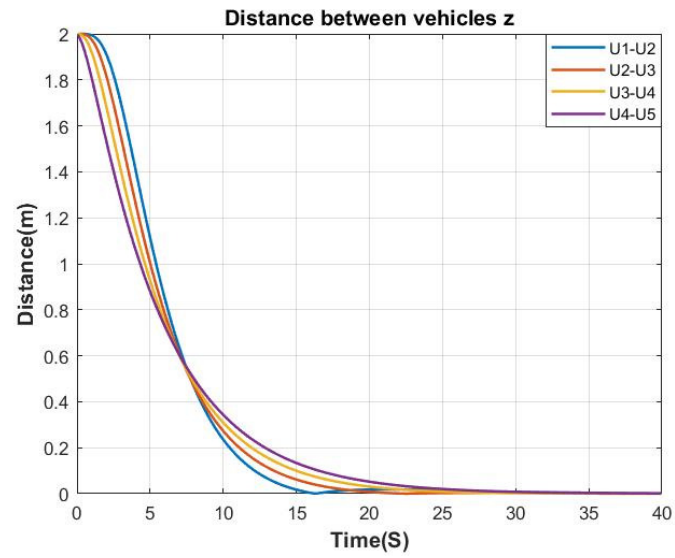


Figure 6.73: Distance between 5 UAVs for line formation along Z-axis.

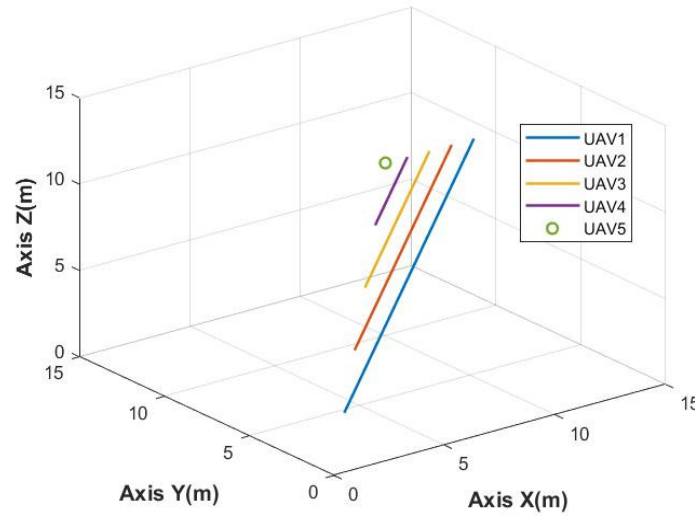


Figure 6.74: 3D Trajectory View of 5 UAVs line formation.

6.6.4 RESULTS CONSENSUS 3 UAVS WITH TRACKING TRAJECTORY AND OBSTACLE AVOIDANCE

In figure 6.75, the movement on the X , Y , Z axes of the UAVs and the guide trajectory generated by the genetic algorithm can be observed. The UAVs not only converge into a formation, but also need to follow the trajectory generated by the genetic algorithm. UAV 3 is responsible for following the trajectory, while UAV 2 follows UAV 3, and UAV 1 must follow UAV 2 with the necessary offsets to maintain a delta formation. The UAVs remain in formation while following the guide trajectory generated by the genetic algorithm.

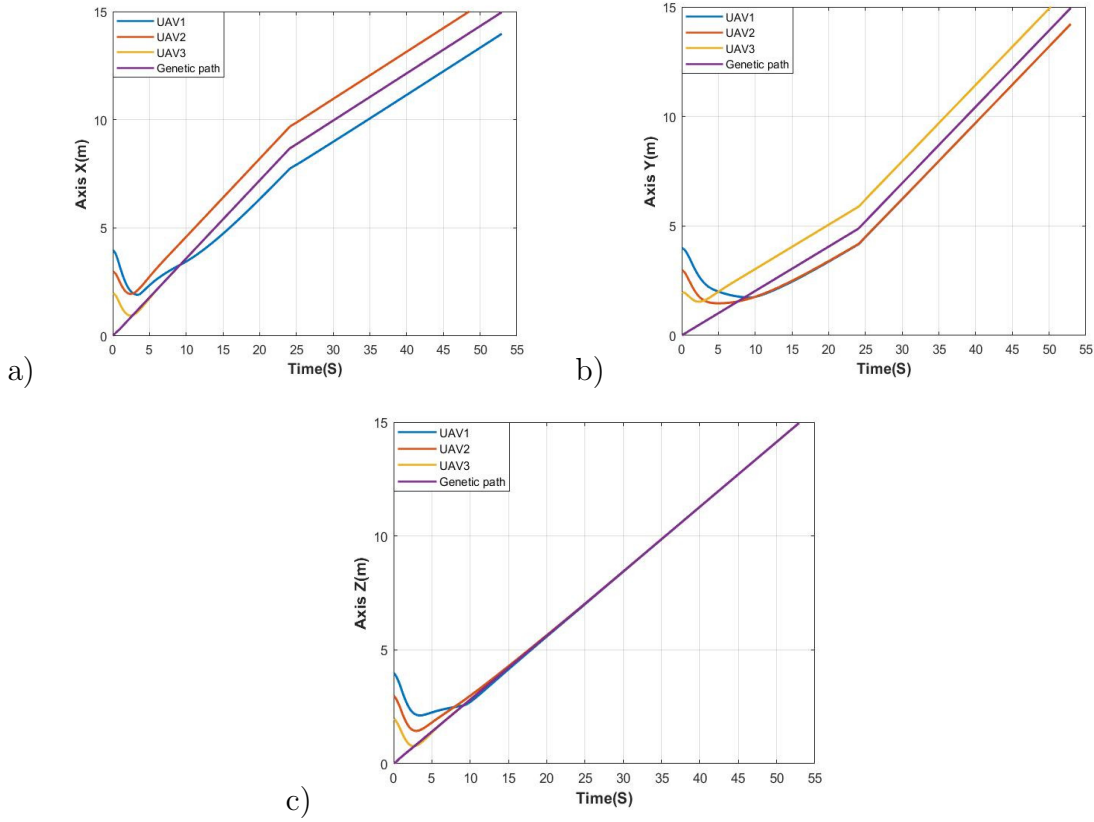


Figure 6.75: Axis of movement for 3 UAVs formation and genetic path: a) axis of movement X, b) axis of movement Y, c) axis of movement Z.

Figures 6.76 and 6.77 show the distance between the UAVs and the distance along the Z-axis, respectively. It can be seen that, in terms of formation, all UAVs converge to a distance of $2m$, demonstrating that formation is achieved. Along the Z-axis, the distance converges to zero, indicating that the UAVs are flying at the same height. Some sudden changes are observed around 10 seconds, due to a safety zone condition that required the UAVs to be no more than $1m$ apart. When the UAVs are closer than one meter, they repel each other, but they still manage to achieve formation after these adjustments.

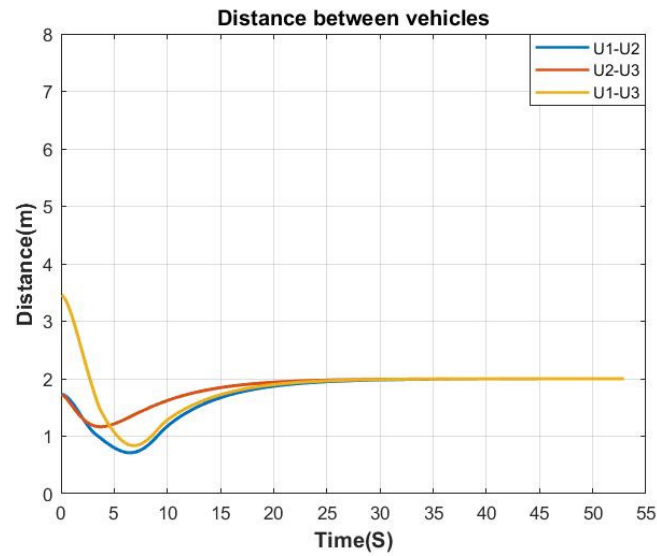


Figure 6.76: Distance between 3 UAVs for delta formation with tracking trajectory.

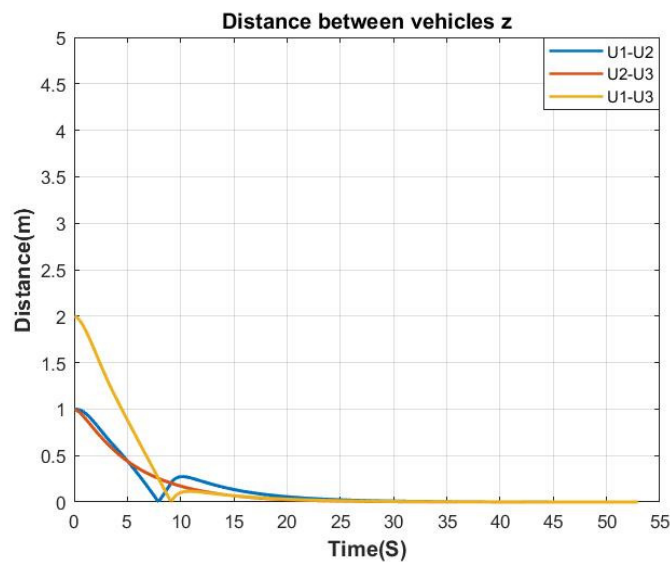


Figure 6.77: Distance between 3 UAVs delta for formation along Z-axis with tracking trajectory.

Finally, Figure 6.78 presents the fitness value graph, which illustrates the optimization process and the gradual decrease in the value of the objective function.

This steady decline is influenced by the speed set for the genetic trajectory. In Figure 6.79, the movement of the UAVs, the static obstacle and the genetic trajectory generated are shown.

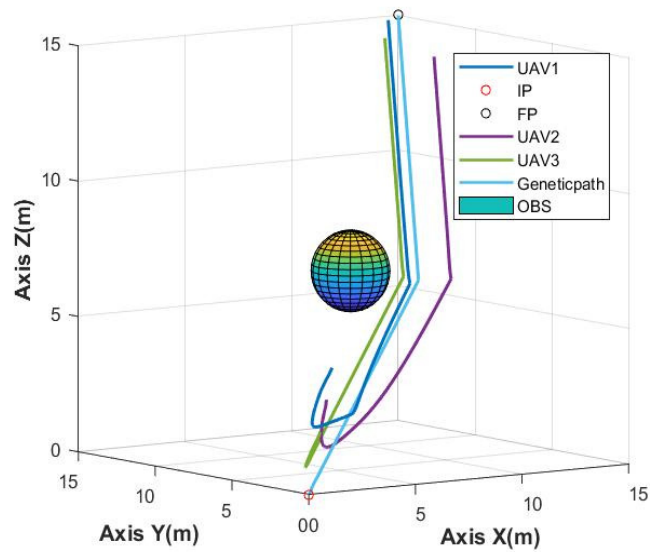


Figure 6.78: Distance between 3 UAVs for delta formation along Z-axis with tracking trajectory.

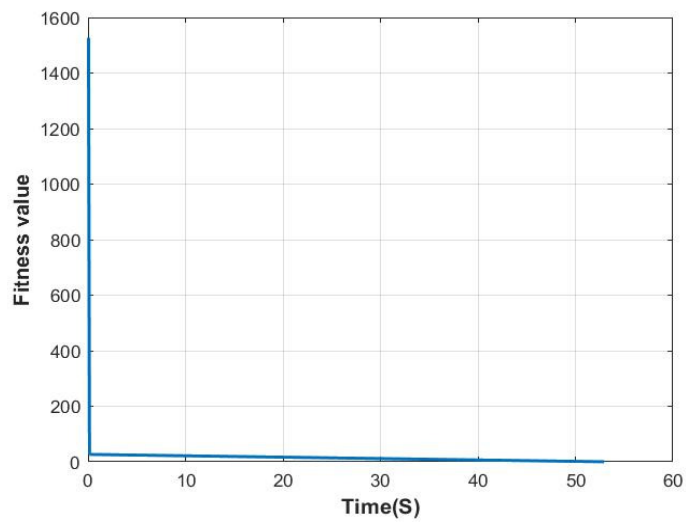


Figure 6.79: Fitness values for delta formation 3 UAVs for genetic path.

6.6.5 RESULTS CONSENSUS 5 UAVs WITH TRACKING TRAJECTORY AND OBSTACLE AVOIDANCE

In Figure 6.80, the movement along the X , Y , and Z axes is shown for the 5 UAVs. It can be seen that vehicles maintain their formation while following the trajectory generated by the genetic algorithm. Additionally, a change in the trajectory around the 25th second is evident, corresponding to the formation maneuvering to avoid the obstacle.

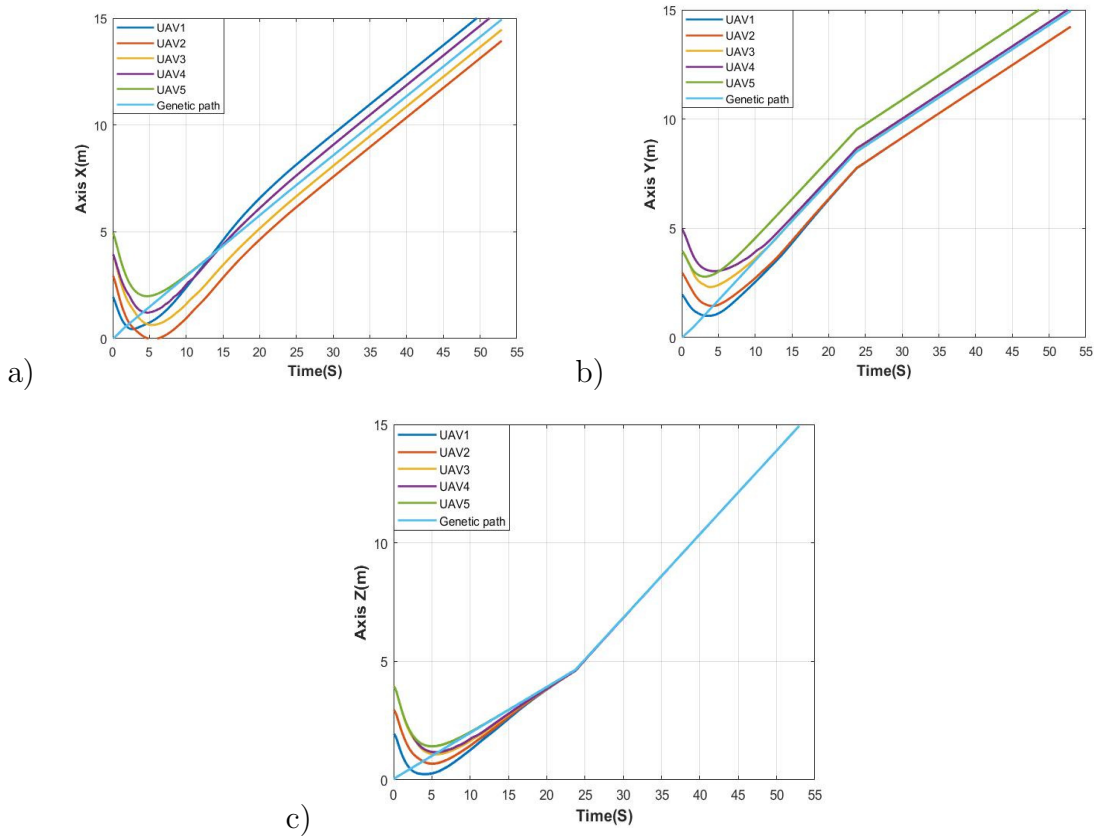


Figure 6.80: Axis of movement for 5 UAVs formation and genetic path: a) axis of movement X , b) axis of movement Y , c) axis of movement Z .

In Figure 6.81, the distance maintained between the UAVs is shown to achieve the formation of the delta with 5 UAVs. Before 5 seconds, the minimum established distance of 0.8m is breached between vehicles $U3$ and $U4$, triggering the

minimum distance criterion. The control system then acts to restore the formation and maintain the minimum distance, causing a repulsion effect between the UAVs and resulting in some vibration within the formation. Eventually, once there is no risk of breaching the safety distance, the distance between the UAVs stabilizes at 1 m. This behavior is also observed in Figure 6.82, which shows the distance along the Z-axis.

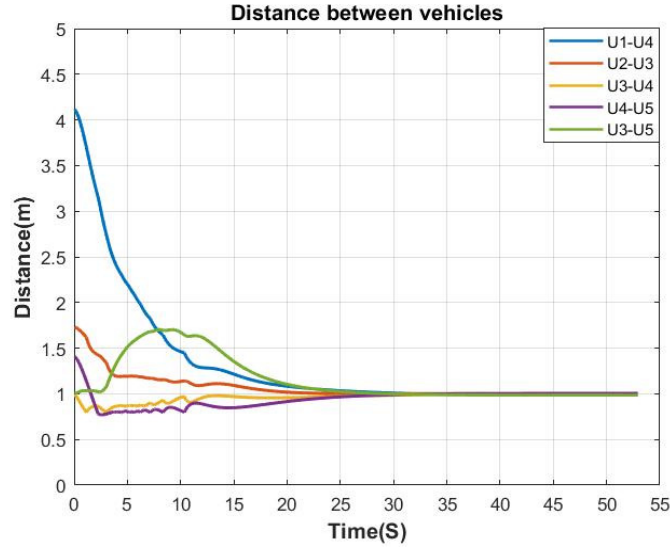


Figure 6.81: Distance between 5 UAVs for delta formation with tracking trajectory.

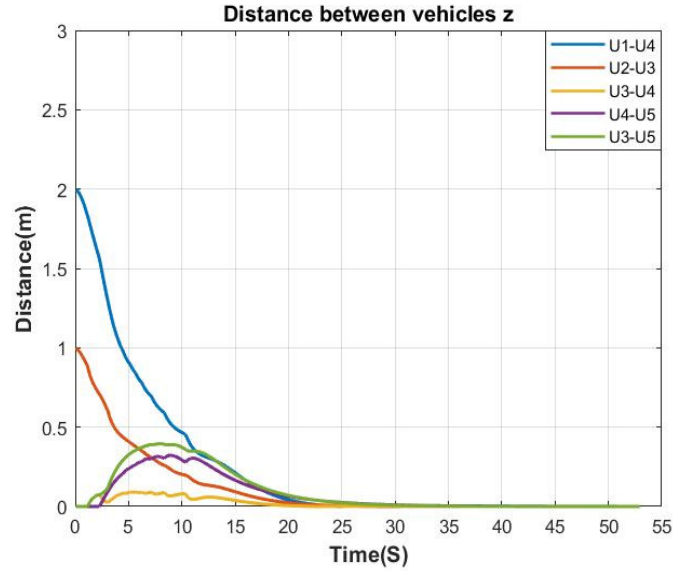


Figure 6.82: Distance between 5 UAVs for delta formation along Z-axis with tracking trajectory

Figure 6.83 displays the 3D trajectory executed by the 5 UAVs alongside the trajectory generated by the genetic algorithm. It can be observed how the formation is maintained successfully while following the genetic trajectory. In Figure 6.84, the fitness value graph of the generated trajectory is presented, demonstrating that from the initial iterations, the genetic algorithm effectively generates a path that avoids the obstacle.

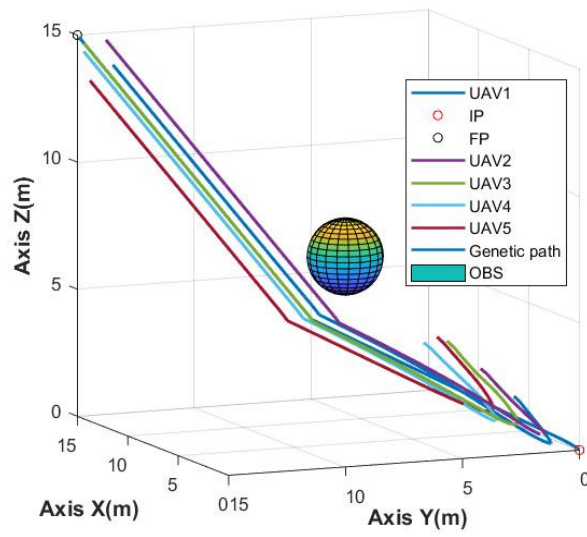


Figure 6.83: 3D Trajectory view of 5 UAVs for delta formation with tracking trajectory

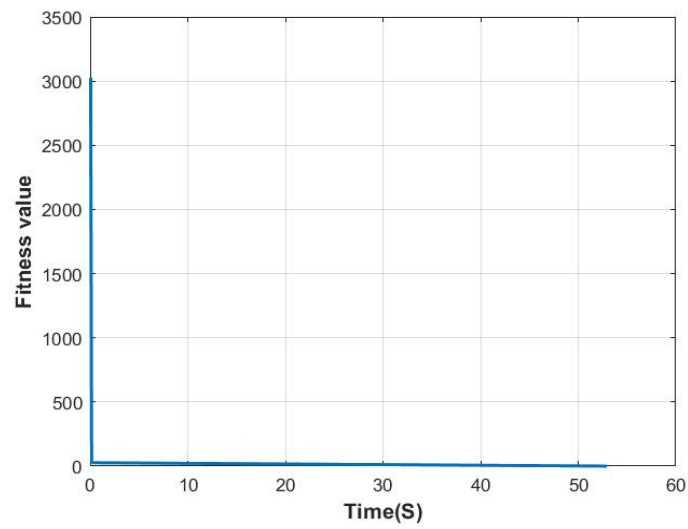


Figure 6.84: Fitness values for delta formation 5 UAVs for genetic path

CHAPTER 7

CONCLUSIONS

The initial tests focused on the application of genetic algorithms for maze solving, evaluating their performance in complex environments. The experiments showed that the algorithm was capable of finding movement sequences with the minimum number of steps possible. Subsequently, these steps were discretized, and continuous trajectories were generated using high-degree polynomials. Both simulations and real-time experiments confirmed that these trajectories can be followed by a quadrotor without any problems.

Another proposal of this research work is a genetic algorithm for path planning in both static and dynamic environments, with the capability to avoid moving obstacles. The population convergence criterion allows for a high mutation rate, maintaining diversity within the population without compromising convergence. In static environments, comparisons between the proposed algorithm, a genetic algorithm with low mutation, and another with high mutation but without population convergence have shown that the presented approach consistently achieves the best results in fewer iterations.

In dynamic environments where obstacles are in motion, the use of strategies such as repopulation, a high mutation rate, and the incorporation of the final point coordinates into the convergence population proved to be effective in responding to

changes in the environment and generating optimal trajectories. This was confirmed through simulations and real-time experiments.

Individual tests of the consensus algorithm demonstrated its efficiency in different scenarios. Initially, it was evaluated with a formation of three vehicles in a delta configuration and later with the coordination of five vehicles in different formations, such as line formation and delta formation. This allowed for verifying the effectiveness of the algorithm in formation management. Subsequently, the formation tracking was tested by incorporating the genetically generated trajectory. In this case, the centroid of the formation was used as the virtual leader, successfully combining the genetic strategy for trajectory planning and vehicle coordination using the consensus algorithm.

The numerical simulations demonstrated that the 5 vehicles were able to join the formation, follow the trajectory, and effectively evade the obstacle.

For future work, it is planned to incorporate an obstacle avoidance criterion for each vehicle individually, since in this case, the vehicles only avoided the obstacle by following the leader. This means that, when trying to join the formation, vehicles could potentially collide with obstacles because of the lack of direct interaction with them.

APPENDIX A

EQUATIONS OF MOTION OF A QUADROTOR

The model of the quadrotor considers an inertial fixed frame as $\mathcal{I}=\{x_{\mathcal{I}}, y_{\mathcal{I}}, z_{\mathcal{I}}\}$ and a body frame fixed attached to the center of gravity of the vehicle as $\mathcal{B}=\{x_{\mathcal{B}}, y_{\mathcal{B}}, z_{\mathcal{B}}\}$. The wind frame $\mathcal{A}=\{x_{\mathcal{A}}, y_{\mathcal{A}}, z_{\mathcal{A}}\}$ is considered during the forward flight, [50].

The Newton-Euler formulation, for rigid body, is used in order to obtain the dynamic model of the quadrotor

$$\dot{\xi} = V \tag{A.1}$$

$$m\dot{V} = Re_3(-T_T) + mge_3 + D_{\xi} \tag{A.2}$$

$$\dot{R} = R\hat{\Omega} \tag{A.3}$$

$$J\dot{\Omega} = -\Omega \times J\Omega + \tau_a + D_{\eta} \tag{A.4}$$

where

- $\xi = (x, y, z)^{\top} \in \mathbb{R}^3$ and $V = (\dot{v}_x, \dot{v}_y, \dot{v}_z)^{\top} \in \mathbb{R}^3$ are the position coordinates and translational velocity relative to the inertial frame.
- $\eta = (\phi, \theta, \psi)^{\top} \in \mathbb{R}^3$ describes the rotation coordinates where ϕ , θ and ψ represent the roll, pitch and yaw or heading, respectively.

- The rotation matrix, $R \in SO(3) : \mathcal{B} \rightarrow \mathcal{I}$, satisfies the $SO(3) = \{R | R \in \mathbb{R}^{3 \times 3}, \det[R] = 1, RR^\top = R^\top R = I\}$ and is parameterized by the Euler angles ϕ , θ and ψ . The rotation matrix is written as

$$R = \begin{pmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{pmatrix}$$

- $\Omega = (p, q, r)^\top \in \mathbb{R}^3$ is the angular velocity in $\mathcal{B}$ and $\hat{\Omega}$ denotes the skew-symmetric matrix of the vector which is given by

$$\hat{\Omega} = \begin{pmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{pmatrix}$$

- e_1 , e_2 , and e_3 are the vectors of the canonical basis of $\mathbb{R}^3$.
- $J \in \mathbb{R}^{3 \times 3}$ contains the moments of inertia and $m \in \mathbb{R}$ denotes the mass of the quadrotor UAV.
- $T_T \in \mathbb{R}^3$ is the total thrust and $\tau_a \in \mathbb{R}^3$ is the actuator moments of the quadrotor.
- $D_\xi = (d_{\xi_1}, d_{\xi_2}, d_{\xi_3})^\top \in \mathbb{R}^3$ and $D_\eta = (d_{\eta_1}, d_{\eta_2}, d_{\eta_3})^\top \in \mathbb{R}^3$ are smooth bounded disturbances, possibly time-varying and state-dependent.

The actuator moments τ_a generated by the four rotors are describe as follows

$$\tau_a = \begin{pmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{pmatrix} = \begin{pmatrix} d[(f_2 + f_4) - (f_1 + f_3)] \\ d[(f_3 + f_4) - (f_1 + f_2)] \\ Q_2 + Q_3 - Q_1 - Q_4 \end{pmatrix}$$

where d denotes the distance from the center of mass to the rotor axis and f_i is the lift force or thrust force of the propeller for $i = 1, 2, 3, 4$. $Q_i = \rho_a A_{d_i} \omega_i^2 r_i^3 c_{Q_i}$ is the reaction moment for the rotors with c_{Q_i} that denotes the rotor shaft moment

coefficient, ρ_a is the air density, A_{d_i} denotes the disk area, ω_i denotes the angular velocity of the rotor and r_i is the rotor radius for $i = 1, 2, 3, 4$.

Gyroscopic moment The gyroscope moment generated by the rotation of the airframe and the four rotors is described as

$$d_{\eta_{gyro}} = \sum_{i=1}^4 (-1)^{i+1} I_{r_i} [\Omega \times (\omega_i e_{b_3})]$$

where I_{r_i} is the moment of inertia of the rotor i and ω_i denotes the angular velocity of the rotor i , with $i=1, 2, 3, 4$.

A.0.0 DISTURBANCES DUE TO WIND

The *aerodynamic forces* produced during the flight are written as

$$D_{\xi}(t) = \begin{pmatrix} d_{\xi_1} \\ d_{\xi_2} \\ d_{\xi_3} \end{pmatrix} = RW^T \begin{pmatrix} D \\ Y \\ 0 \end{pmatrix}$$

with the rotation aerodynamic matrix $W : \mathcal{B} \rightarrow \mathcal{A}$ that transforms a force from the body frame to aerodynamic frame is described as

$$W = \begin{pmatrix} c_{\alpha}c_{\beta} & s_{\beta} & s_{\alpha}c_{\beta} \\ -c_{\alpha}s_{\beta} & c_{\beta} & -s_{\alpha}s_{\beta} \\ -s_{\alpha} & 0 & c_{\alpha} \end{pmatrix}$$

where α is the angle of attack and β are the sideslip angle. L , Y , and D are the aerodynamic forces: lift, side force, and drag, respectively, [51].

The *Aerodynamic moments* The aerodynamic moments generated during the flight are written as described as

$$d_{\eta_{aero}}(t) = \begin{pmatrix} \mathcal{L} \\ \mathcal{M} \\ \mathcal{N} \end{pmatrix}$$

where $\mathcal{L}$, $\mathcal{M}$ and $\mathcal{N}$ are the aerodynamic rolling, pitching and yawing moments, respectively, [52].

Rewriting the disturbances in the time-varying vectors $D_\xi(t)$ and $D_\eta(t)$, it results

$$D_\xi(t) = \begin{pmatrix} d_{\xi_1} \\ d_{\xi_2} \\ d_{\xi_3} \end{pmatrix}, \quad D_\eta(t) = d_{\eta_{gyro}}(t) + d_{\eta_{aero}}(t) = \begin{pmatrix} d_{\eta_1} \\ d_{\eta_2} \\ d_{\eta_3} \end{pmatrix}$$

which are continuously differentiable.

Thus, using Equations (A.2)–(A.4), a nonlinear set of equations is described as:

$$\begin{aligned} \dot{x} &= v_x \\ \dot{y} &= v_y \\ \dot{z} &= v_z \\ \dot{v}_x &= \frac{F_{xB}}{m} (c_\theta c_\psi) + \frac{F_{yB}}{m} (s_\phi s_\theta c_\psi - c_\phi s_\psi) + \frac{F_{zB}}{m} (c_\phi s_\theta c_\psi + s_\phi s_\psi) + d_{\xi_1} \\ \dot{v}_y &= \frac{F_{xB}}{m} (c_\theta s_\psi) + \frac{F_{yB}}{m} (s_\phi s_\theta s_\psi + c_\phi c_\psi) + \frac{F_{zB}}{m} (c_\phi s_\theta s_\psi - s_\phi c_\psi) + d_{\xi_2} \\ \dot{v}_z &= \frac{F_{xB}}{m} (-s_\theta) + \frac{F_{yB}}{m} (s_\phi c_\theta) + \frac{F_{zB}}{m} (c_\phi c_\theta) + g + d_{\xi_3} \\ \dot{\phi} &= p + q \sin(\phi) \tan(\theta) + r \cos(\phi) \tan(\theta) \\ \dot{\theta} &= q \cos(\phi) - r \sin(\phi) \\ \dot{\psi} &= q \sec(\theta) \sin(\phi) + r \sec(\theta) \cos(\phi) \\ \dot{p} &= \left(\frac{J_{yy} - J_{zz}}{J_{xx}} \right) qr + \left(\frac{1}{J_{xx}} \right) \tau_\phi + d_{\eta_1} \\ \dot{q} &= \left(\frac{J_{zz} - J_{xx}}{J_{yy}} \right) pr + \left(\frac{1}{J_{yy}} \right) \tau_\theta + d_{\eta_2} \\ \dot{r} &= \left(\frac{J_{xx} - J_{yy}}{J_{zz}} \right) pq + \left(\frac{1}{J_{zz}} \right) \tau_\psi + d_{\eta_3} \end{aligned} \tag{A.5}$$

BIBLIOGRAPHY

- [1] F. L. Lewis, H. Zhang, K. Hengster-Movric, and A. Das, *Cooperative Control of Multi-Agent Systems: Optimal and Adaptive Design Approaches*. Communications and Control Engineering, Springer, 2014.
- [2] B. Patle, G. Babu L, A. Pandey, D. Parhi, and A. Jagadeesh, “A review: On path planning strategies for navigation of mobile robot,” *Defence Technology*, vol. 15, no. 4, pp. 582–606, 2019.
- [3] J. R. Sánchez-Ibáñez, C. J. P. del Pulgar, and A. García-Cerezo, “Path planning for autonomous mobile robots: A review,” *Sensors*, vol. 21, no. 23, p. 7898, 2021.
- [4] S. Aggarwal and N. Kumar, “Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges,” *Computer Communications*, vol. 149, pp. 270–299, 2020.
- [5] A. Majeed and S. Lee, “A fast global flight path planning algorithm based on space circumscription and sparse visibility graph for unmanned aerial vehicle,” *Electronics Switzerland*, vol. 7, 2018.
- [6] H. Wang, W. Lyu, P. Yao, X. Liang, and C. Liu, “Three-dimensional path planning for unmanned aerial vehicle based on interfered fluid dynamical system,” *Chinese Journal of Aeronautics*, vol. 28, 2015.
- [7] Z. Yingkun, “Flight path planning of agriculture uav based on improved artificial potential field method,” *Proceedings of the 30th Chinese Control and Decision Conference, CCDC 2018*, 2018.

-
- [8] G. E. D. Flores, E. S. E. Quesada, S. S. Cruz, L. R. G. Carrillo, and R. Lozano, "Online uas local path planning algorithm for outdoors obstacle avoidance based on attractive and repulsive potential fields," *2016 International Conference on Unmanned Aircraft Systems, ICUAS 2016*, 2016.
 - [9] F. Samaniego, J. Sanchis, S. Garcia-Nieto, and R. Simarro, "Uav motion planning and obstacle avoidance based on adaptive 3d cell decomposition: Continuous space vs discrete space," *2017 IEEE 2nd Ecuador Technical Chapters Meeting, ETCM 2017*, vol. 2017-January, 2018.
 - [10] L. Blasi, E. D'Amato, M. Mattei, and I. Notaro, "Path planning and real-time collision avoidance based on the essential visibility graph," *Applied Sciences (Switzerland)*, vol. 10, 2020.
 - [11] Y. Xue and J.-Q. Sun, "Solving the path planning problem in mobile robotics with the multi-objective evolutionary algorithm," *Applied Sciences*, vol. 8, no. 9, 2018.
 - [12] S. A. F. Aghda and M. Mirfakhraei, "Improved routing in dynamic environments with moving obstacles using a hybrid fuzzy-genetic algorithm," *Future Generation Computer Systems*, vol. 112, 2020.
 - [13] S. Shao, Y. Peng, C. He, and Y. Du, "Efficient path planning for uav formation via comprehensively improved particle swarm optimization," *ISA Transactions*, vol. 97, 2020.
 - [14] H. Shiri, J. Park, and M. Bennis, "Massive autonomous uav path planning: A neural network based mean-field game theoretic approach," *arXiv*, 2019.
 - [15] S. Konatowski and P. Pawlowski, "Ant colony optimization algorithm for uav path planning," *14th International Conference on Advanced Trends in Radio-electronics, Telecommunications and Computer Engineering, TCSET 2018 - Proceedings*, vol. 2018-April, 2018.

- [16] B. K. Patle, D. R. Parhi, A. Jagadeesh, and S. K. Kashyap, "Matrix-binary codes based genetic algorithm for path planning of mobile robot," *Computers and Electrical Engineering*, vol. 67, 2018.
- [17] A. Atyabi and D. M. W. Powers, "Review of classical and heuristic-based navigation and path planning approaches," *International Journal of Advancements in Computing Technology (IJACT)*, vol. 5, 2013.
- [18] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press, 1975. second edition, 1992.
- [19] C. Lamini, S. Benhlila, and A. Elbekri, "Genetic algorithm based approach for autonomous mobile robot path planning," *Procedia Computer Science*, 2018.
- [20] Y. Wang and W. Chen, "Path planning and obstacle avoidance of unmanned aerial vehicle based on improved genetic algorithms," *Proceedings of the 33rd Chinese Control Conference, CCC 2014*, 2014.
- [21] M. Elhoseny, A. Shehab, and X. Yuan, "Optimizing robot path in dynamic environments using genetic algorithm and bezier curve," *Journal of Intelligent and Fuzzy Systems*, vol. 33, 2017.
- [22] V. Roberge, M. Tarbouchi, and G. Labonte, "Fast genetic algorithm path planner for fixed-wing military uav using gpu," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 54, 2018.
- [23] M. Elhoseny, A. Tharwat, and A. E. Hassanien, "Bezier curve based path planning in a dynamic field using modified genetic algorithm," *Journal of Computational Science*, vol. 25, pp. 339–350, 2018.
- [24] R. Shivgan and Z. Dong, "Energy-efficient drone coverage path planning using genetic algorithm," pp. 1–6, 2020.
- [25] M. d. S. Arantes, J. d. S. Arantes, C. F. M. Toledo, and B. C. Williams, "A hybrid multi-population genetic algorithm for uav path planning," p. 853–860, 2016.

- [26] Y. V. Pehlivanoglu and P. Pehlivanoglu, “An enhanced genetic algorithm for path planning of autonomous uav in target coverage problems,” *Applied Soft Computing*, vol. 112, p. 107796, 2021.
- [27] Y. V. Pehlivanoglu, “A new vibrational genetic algorithm enhanced with a voronoi diagram for path planning of autonomous uav,” *Aerospace Science and Technology*, vol. 16, no. 1, pp. 47–55, 2012.
- [28] C. Zhang, W. Zhou, W. Qin, and W. Tang, “A novel uav path planning approach: Heuristic crossing search and rescue optimization algorithm,” *Expert Systems with Applications*, vol. 215, p. 119243, 2023.
- [29] Y. Pan, Y. Yang, and W. Li, “A deep learning trained by genetic algorithm to improve the efficiency of path planning for data collection with multi-uav,” *IEEE Access*, vol. 9, pp. 7994–8005, 2021.
- [30] B. Chen, S. Lai, C. Chen, P. Shu, S. Chen, Z. Lai, and L. Xu, “Uav path planning based on improved genetic algorithm,” pp. 229–232, 2021.
- [31] L. Deng, H. Chen, X. Zhang, and H. Liu, “Three-dimensional path planning of uav based on improved particle swarm optimization,” *Mathematics*, vol. 11, no. 9, p. 1987, 2023.
- [32] O. Tutsoy, D. Asadi, K. Ahmadi, S. Y. Nabavi-Chashmi, and J. Iqbal, “Minimum distance and minimum time optimal path planning with bioinspired machine learning algorithms for faulty unmanned air vehicles,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 8, pp. 9069–9077, 2024.
- [33] Y. Zhou, Y. Su, A. Xie, and L. Kong, “A newly bio-inspired path planning algorithm for autonomous obstacle avoidance of uav,” *Chinese Journal of Aeronautics*, vol. 34, no. 9, pp. 172–186, 2021.
- [34] H. Duan, Q. Luo, Y. Shi, and G. Ma, “?hybrid particle swarm optimization and genetic algorithm for multi-uav formation reconfiguration,” *IEEE Computational Intelligence Magazine*, vol. 8, no. 3, pp. 16–27, 2013.

- [35] Y. Mohamed Ben Ali, “An augmented particle swarm model based bi-acceleration factor,” *International Journal of Intelligent Computing and Cybernetics*, vol. 4, no. 2, pp. 187–205, 2011.
- [36] D. Guo and Y. Zhang, “Novel recurrent neural network for time-varying problems solving [research frontier],” *IEEE Computational Intelligence Magazine*, vol. 7, no. 4, pp. 61–65, 2012.
- [37] M. Efe, “Neural network assisted computationally simple $pi^{\lambda}d^{\mu}$ control of a quadrotor uav,” *IEEE Transactions on Industrial Informatics*, vol. 7, no. 2, pp. 354–361, 2011.
- [38] S. Bansal, A. K. Akametalu, F. J. Jiang, F. Laine, and C. J. Tomlin, “Learning quadrotor dynamics using neural network for flight control,” pp. 4653–4660, 2016.
- [39] B. Zhang, Z. Mao, W. Liu, J. Liu, and Z. Zheng, “Cooperative and geometric learning for path planning of uavs,” pp. 69–78, 2013.
- [40] D. H. Stolfi and G. Danoy, “An evolutionary algorithm to optimise a distributed uav swarm formation system,” *Applied Sciences*, vol. 12, no. 20, 2022.
- [41] G. M. de Oliveira, R. G. Silva, L. R. do Amaral, and L. G. Martins, “An evolutionary-cooperative model based on cellular automata and genetic algorithms for the navigation of robots under formation control,” pp. 426–431, 2018.
- [42] D. E. Goldberg, “Genetic algorithms in search, optimization, and machine learning,” *Choice Reviews Online*, 1989.
- [43] A. Hassanat, K. Almohammadi, E. Alkafaween, E. Abunawas, A. Hammouri, and V. B. S. Prasath, “Choosing mutation and crossover ratios for genetic algorithms: A review with a new dynamic approach,” *Information*, vol. 10, no. 12, p. 390, 2019.
- [44] M. Golub and L. Budin, “An asynchronous model of global parallel genetic algorithms,” pp. 353–359, 2000.

- [45] C. W. Reynolds, “Flocks, herds, and schools: A distributed behavioral model,” pp. 25–34, 1987.
- [46] Bitcraze, “Crazyflie 2.1,” 2011. <https://www.bitcraze.io/products/crazyflie-2-1/>.
- [47] W. Giernacki, M. Skwierczynski, W. Witwicki, P. Wronski, and P. Kozierski, “Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering,” in *22nd International Conference on Methods and Models in Automation and Robotics (MMAR)*, pp. 37–42, 2017.
- [48] E. J. Ollervides-Vazquez, E. G. Rojo-Rodriguez, E. U. Rojo-Rodriguez, L. E. Cabriaes-Ramirez, and O. Garcia-Salazar, “Two-layer saturated pid controller for the trajectory tracking of a quadrotor uav,” in *2020 International Conference on Mechatronics, Electronics and Automotive Engineering (ICMEAE)*, pp. 85–91, 2020.
- [49] M. A. Gutierrez-Martinez, E. G. Rojo-Rodriguez, L. E. Cabriaes-Ramirez, L. A. Reyes-Osorio, P. Castillo, and O. Garcia-Salazar, “Collision-free path planning based on a genetic algorithm for quadrotor uavs,” in *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 948–957, 2020.
- [50] B. L. Stevens and F. L. Lewis, *Aircraft Control and Simulation*. USA: John Wiley and Sons, 1992.
- [51] J. G. Leishman, *Principles of Helicopter Aerodynamics*. USA: Cambridge University Press, 2006.
- [52] R. Lozano, *Unmanned Aerial Vehicles: Embedded Control*. USA: John Wiley-ISTE Ltd, 2010.

AUTOBIOGRAPHICAL SUMMARY

MANUEL ALEJANDRO GUTIERREZ MARTINEZ

Candidate for obtaining the degree of
Doctor of Science in Aeronautical Engineering

Universidad Autónoma de Nuevo León
Facultad de Ingeniería Mecánica y Eléctrica

Tesis:

DESIGN OF INTELLIGENT STRATEGIES FOR THE FORMATION
FLIGHT OF MULTIPLE UAVS BASED ON GENETIC ALGORITHMS

ABOUT THE AUTHOR

I, Manuel Alejandro Gutiérrez Martínez, am a candidate for the Doctorate in Aeronautical Engineering at the Facultad de Ingeniería Mecánica y Eléctrica(FIME) of the Universidad Autónoma de Nuevo León. In this section of my thesis, I would like to provide a brief overview of my background and personal motivations.

Although I consider myself an ordinary individual, I am deeply passionate about learning and research. I must admit that when learning becomes a mere obligation, it tends to lose part of its charm. However, when I have the freedom to explore topics that genuinely spark my curiosity, I find a profound sense of purpose and fulfillment. That intrinsic motivation—the drive to understand, question, and discover—defines an essential part of who I am.

Now, I will share a brief description of my academic background.

My academic journey began at Escuela Industrial y Preparatoria Técnica Álvaro Obregón, where I discovered a strong affinity for mathematics and science. This newfound ability became a personal challenge, motivating me to strive for constant improvement. Upon completing my preparatory studies, I graduated as a Technician in Mechatronics.

After this stage, I decided to enroll in the Facultad de Ingeniería Mecánica y Eléctrica at the Universidad Autónoma de Nuevo León. Initially, I was uncertain about which degree program to pursue. Eventually, I chose Mechatronics Engineering to continue the technical training I had already started.

During my undergraduate studies, I thoroughly enjoyed my classes and everything I was learning. I always felt excited to start a new course or topic; the idea of learning something new fascinated me. However, the prospect of eventually working in a corporate environment did not appeal to me at all. Fortunately, during the early semesters, that future still felt distant, allowing me to focus solely on studying and learning.

I should also mention that project management and administrative subjects, particularly those involving standards and protocols, were quite dull for me. Unlike many of my classmates, I found enjoyment in what were often considered “filler subjects,” such as psychology or art. During my undergraduate years, I also realized how much I appreciated taking classes taught by PhDs. Their mastery of the subject matter impressed me, and I felt I could learn more from them than anyone else. This was when the idea of pursuing a doctorate began to take shape in my mind—an academic life that would allow me to continue learning in the way I enjoyed most.

Although I was still in the early semesters at that time, I often expressed that I would eventually become a doctor, even though I didn’t know what field that would be in. I have always been someone who tends to look back. I’m not sure whether that’s good or bad; perhaps it’s a way of staying alert and always aware of what is behind me. But today, as I look back to write this text, I realize I am exactly where I always said I wanted to be.

Perhaps I have strayed a bit from the purpose of this autobiography, but in the end, it is my autobiography, and I can decide what to include. However, to maintain the formality required for such a document, I will now present a structured summary of my academic path.

I graduated as a Mechatronics Engineer in 2018. Subsequently, in 2019, I entered the Master’s program in Aeronautical Engineering at the same Facultad de Ingeniería Mecánica y Eléctrica, earning my degree as a Master of Engineering with a concentration in Aeronautical Structures in 2020. Finally, in 2021, I began my

Doctorate in Aeronautical Engineering, where I successfully completed the academic coursework and developed my doctoral thesis in 2023.

This thesis, titled Design of Intelligent Strategies for the Formation Flight of Multiple UAVs Based on Genetic Algorithms is the result of my dedication to both theoretical research and practical implementation. The project aims to contribute to the field of trajectory planning for unmanned aerial vehicles using artificial intelligence techniques and reflects my interest in connecting academic knowledge with real-world applications.

I am grateful for the support of my advisors, professors, and colleagues, who guided and encouraged me throughout this academic journey. I hope to continue my professional development, either by pursuing further postgraduate studies or by applying my knowledge in the industry to solve complex engineering challenges.