MIDCOURSE AND FINAL PHASE PATH PLANNING FOR AIR VEHICLES TO AVOID COUNTERMEASURES

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

BY

BERK ATAELİ

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

MARCH 2021

Approval of the thesis:

MIDCOURSE AND FINAL PHASE PATH PLANNING FOR AIR VEHICLES TO AVIOD COUNTERMEASURES

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

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

Prof. Dr. İlkay Ulusoy
Head of the Department, **Electrical and Electronic Eng.** _____

Prof. Dr. M. Kemal Leblebicioğlu
Supervisor, **Electrical and Electronic Eng., METU** _____

Examining Committee Members:

Prof. Dr. Ozan Tekinalp
Aerospace Engineering, METU _____

Prof. Dr. M. Kemal Leblebicioğlu
Electrical and Electronic Eng., METU _____

Prof. Dr. Klaus Werner Schmidt
Electrical and Electronic Eng., METU _____

Prof. Dr. Umut Orguner
Electrical and Electronic Eng., METU _____

Prof. Dr. Mehmet Önder Efe
Computer Engineering, Hacettepe University _____

Date: 25.03.2021

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

Name Last name: Berk Ataeli

Signature:

ABSTRACT

MIDCOURSE AND FINAL PHASE PATH PLANNING FOR AIR VEHICLES TO AVOID COUNTERMEASURES

Ataeli, Berk
Master of Science, Electrical and Electronic Engineering
Supervisor: Prof. Dr. M. Kemal Leblebicioğlu

March 2021, 98 pages

Path planning is a fundamental element of various common fields, such as robotics, computer animation, and wireless sensor networks. In this study, path planning for air vehicles is investigated in two phases. For the midcourse phase, the path with the minimum length, fuel consumption, and risk of being detected is sought between two particular points in 3D complicated environments. To achieve this aim, a Genetic Algorithm is utilized with a special feasible initial population creation method. Also, some problem dependent operators are defined in order to make the convergence faster and more accurate, which is very crucial when the environment involves complex obstacles. On the other hand, since there are typically air defense systems located close to targets, safety becomes more important in the final phase. In order to avoid these dangerous situations, evasive maneuvers must be carried out within short ranges near the target. Thus, the mathematical modeling of air vehicles, guidance systems, and Control Actuation Systems are integrated and several tactical strategies are examined against the Close-In Weapon System that is also modeled in this dissertation.

Keywords: Path Planning, Genetic Algorithm, Flight Mechanics, Evasive Maneuver, CIWS.

ÖZ

HAVA ARAÇLARININ KARŞI TEDBİRLERDEN KAÇINMA AMAÇLI ORTA VE SON FAZDAKİ YOL PLANLAMASI

Ataeli, Berk
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği
Tez Yöneticisi: Prof. Dr. M. Kemal Leblebicioğlu

Mart 2021, 98 sayfa

Yol planlama; robotik, bilgisayar animasyonu ve telsiz sensör ağları gibi çeşitli yaygın alanlarda kullanılan temel bir unsurdur. Bu çalışmada, hava araçlarının iki fazda yol planlaması incelenmiştir. Orta fazda üç boyutlu karmaşık ortamlarda; belirli iki nokta arasındaki en az uzunluk, yakıt tüketimi ve tespit edilme riski değerlerine sahip yol aranmaktadır. Bu amaca ulaşmak adına, özel bir makul başlangıç popülasyonu oluşturma methodu ile beraber bir Genetik Algoritma kullanılmıştır. Ayrıca, ortamda karmaşık engeller bulunduğu zaman çok daha önemli hale gelen yakınsamayı daha hassas ve hızlı yapabilmek için probleme özel birkaç operatör tanımlanmıştır. Öte yandan, genellikle hedeflerin yakınına hava savunma sistemleri konumlandırıldığı için son fazda güvenlik daha da önem kazanmaktadır. Bu tehlikeli sistemlerden sakınmak için kaçınma manevraları yapmak zorunlu hale gelir. Bu sebeple, hava araçlarının matematiksel modeli, güdüm ve CAS sistemleri entegre edilmiş ve bu çalışmada ayrıca modellenmiş olan CIWS adı verilen bir hava savunma sistemine karşı çeşitli kaçınma manevrası stratejileri incelenmiştir.

Anahtar Kelimeler: Yol Planlaması, Genetik Algoritma, Uçuş Mekaniği, Kaçınma Manevrası, CIWS

To my dearest family

ACKNOWLEDGMENTS

First, I would like to thank my supervisor Prof. Dr. Kemal Leblebicioğlu for his guidance and advice throughout the research. Besides, I appreciate Dr. Özgür Ateşoğlu for his valuable comments and contributions on this work.

The technical assistance of my colleagues at Roketsan Inc., Mehmet Ufuk Deniz, Emre Dede, and especially Abdulkadir Ercan are gratefully acknowledged. Also, I sincerely appreciate my company for encouraging postgraduate education.

Special thanks to my dearest friends Banu Çiçek Büyüker and Murat Bayraktar for their encouragement, support, and bringing new ideas to my perspective.

I cannot truly exaggerate my gratitude for my parents for their support, encouragement, and love. With my graduation, all of their children have got master's degrees so I want to congratulate them.

My sweetest sister Beyza and brother Anıl deserve the greatest gratitude for their endless love and support. None of my accomplishments would be possible without them.

This work is financially supported by the Scientific and Technological Research Council of Turkey under BİDEB 2210-A program.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF ALGORITHMS

ALGORITHMS

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS

AAA	Antiaircraft Artillery Systems
CAS	Control Actuation Systems
CIWS	Closed-In Weapon Systems
CG	Center of Gravity
DOF	Degrees of Freedom
GA	Genetic Algorithm
GPS	Global Positioning System
IADS	Integrated Air Defense Systems
LOS	Line of Sight
PNG	Proportional Navigation Guidance
PSO	Particle Swarm Optimization
SAM	Surface-to-Air Missiles
SPEA	Strength Pareto Evolutionary Algorithm

LIST OF SYMBOLS

SYMBOLS

u	Velocity in x-axis of body frame
v	Velocity in y-axis of body frame
w	Velocity in z-axis of body frame
X	Force in x-axis of body frame
Y	Force in y-axis of body frame
Z	Force in z-axis of body frame
p	Angular velocity in x-axis of body frame
q	Angular velocity in y-axis of body frame
r	Angular velocity in z-axis of body frame
L	Rolling moment
M	Pitching moment
N	Yawing moment
ϕ	Roll angle
θ	Pitch angle
ψ	Yaw angle
δ_e	Elevator deflection angle
δ_a	Aileron deflection angle
δ_r	Rudder deflection angle
T	Thrust
m	Mass
g	Gravity
I	Inertia matrix
ρ	Air density
S	Surface of wing
d	Reference length

α	Angle of attack
β	Sideslip angle
J	Cost function
λ	LOS angle

CHAPTER 1

INTRODUCTION

Optimizing the flight trajectory of the air vehicles is essential to minimize flight time, fuel consumption and maximize safety. The overall trajectory can be examined in two parts, midcourse and final phases.

In this study, the optimal paths are investigated in complicated 3D environments using optimization algorithms for the midcourse phase. There are classic and modern intelligent algorithms in the literature that have been studied so far to solve this problem. In classic algorithms, a generalization of the A* algorithm for sampling-based motion planning [1], an improved sparse A* algorithm [2] are discussed earlier and they enhance the planning efficiency. Nevertheless, these algorithms do not give acceptable results in complicated environments. Among the modern intelligent algorithms, there are artificial neural networks, evolutionary algorithms [3], intelligent swarm algorithms [4], and simulated annealing techniques [5]. From the intelligent swarms part, path planning based on ant colony algorithm with A* heuristic method [4] has efficient searching capabilities, which makes it effective and fast in dealing with complicated environments. Since the environments are modeled in 3D space in this work, the algorithm should be fast, powerful, and compatible with the requirements. Among all the alternatives, the Genetic Algorithm is selected to find the optimal paths, which meets all the necessities. Lately, it is significantly used for solving optimization problems thanks to its capability of fast and effective searching. A genetic algorithm is applied to path planning for autonomous mobile robots in [6], where the environments are modeled as 2D grid maps. A fitness function is proposed and the effect of different environments is examined. Also, [7] presents a genetic algorithm with fixed length chromosomes,

which improves computational power in the 2D path planning problem. In addition, one significant advantage of the Genetic Algorithm is that problem dependent operators can be defined specific to the problem. Likewise, extra evolutionary operators are utilized in [8] to find precise and effective paths. Also, it compares GA with another commonly used optimization algorithm PSO.

For the final phase of the flight trajectory, the aim is to avoid air defense systems and reach the target. Performing evasive maneuvers is a solution against air defense systems and it is made possible by controlling the position of the air vehicle. Mathematical modeling of air vehicles is necessary to get the relationship between forces/moments and positions/velocities of the air vehicles [9]. Also, guidance systems must be developed to guide the air vehicles towards the target. Then, the Control Actuation Systems makes the aircraft follow the command signals produced by the guidance systems. All three subsystems are integrated and the resulting system makes sure that the air vehicle does not miss the target when it tries to avoid the air defense systems. Likewise, mathematical modeling of an air defense system is required in order to simulate the contest between the models. Therefore, a gun-based CIWS is modeled and used for defending against missiles in this study.

Air defense systems are designed to destroy incoming threats by detecting, tracking, and attacking their targets. They are categorized in [11] as antiaircraft artillery, surface-to-air missiles, and Integrated Air Defense System (IADS). In brief, Surface-to-Air Missiles (SAM) have guidance systems that manage the movable control surfaces on the missile so that they can track the targets. Integrated Air Defense System is an assembly of several defense mechanisms assigned to joint areas. Antiaircraft Artillery systems (AAA) fire at air targets, which are usually within short ranges such as three kilometers. Close-in weapon systems (CIWS) are examined in the AAA category and defend a limited area around a critical place. They are also called point-defense weapon systems. Figure 1.1 shows a CIWS on a ship that can detect the threats within 5.5 km and destroy them within 1.5 km.

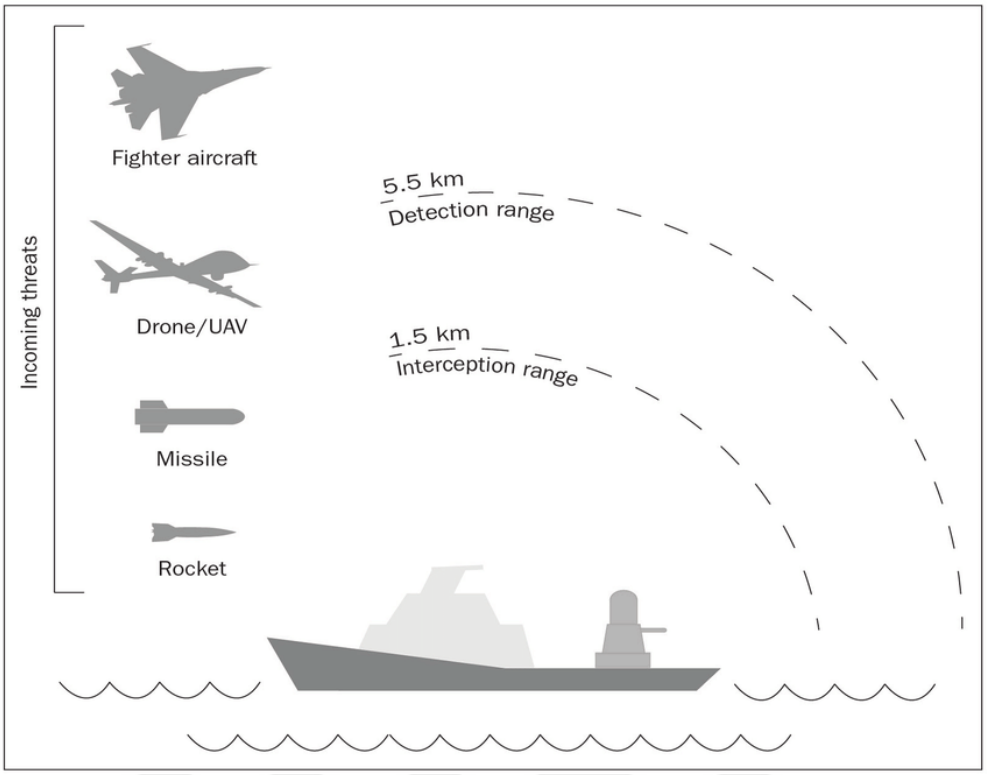

Figure 1.1. Close-In Weapon System [10]

[12] explains the simulation flowchart of general simulation software for CIWS. A closed-loop fire correction algorithm can boost shooting accuracy by using a Kalman Filter, which is shown in [13].

Evasive maneuvers are crucial for the survivability of fighter aircraft and missiles against advanced anti-air defense systems. The maneuver types have been discussed for over 30 years. The weaving maneuver [14] provides an enhancement in the performance of the missiles to avoid the defense systems. Another prevalent type, barrel-roll maneuver [15], continuously changes the maneuver direction and produces very useful trajectories for missiles. It is also a complex problem because the missiles should reach the target besides having the correct maneuver. Therefore, the evasive maneuver command diminishes as the missile gets closer to the target, not to get far away from and miss the target.

1.1 Thesis Contributions

A genetic algorithm structure is used to solve the 3D path planning problem in this study. Most of the researches has investigated the problem for 2D environments. When the environment becomes three-dimensional, the problem gets more complex. Therefore, a fast and solid algorithm is proposed to handle the complexity of the problem. Also, there are several improvements in genetic algorithm stages as follows.

- Instead of generating the initial population by only using random number generators, a novel method is developed in which feasible initial populations are created.
- Besides the standard operators, novel problem dependent operators specific to the path planning problem are developed for fast convergence.

On the other hand, since the aim is to avoid anti-aircraft systems in the final phase, an anti-aircraft system, CIWS is modeled elaboratively enough to measure the performance of the air vehicle's maneuvers. This enables examining different strategies and answer the question "Which strategy should be applied for maneuvers to avoid air defense systems?". In other words, an air vehicle model and an anti-aircraft system model are simulated simultaneously and their competition is investigated.

CHAPTER 2

MATHEMATICAL MODEL OF AIR VEHICLES

Analysis of motion of air vehicles is very complicated because they have six degrees of freedom. Fundamental knowledge of flight dynamics is required in order to build mathematical models of air vehicles. Therefore, necessary information such as the notation, axes systems and transformations are given in the following sections.

2.1 Reference Frames

The frame of reference is a particular point of view to describe a motion. There are several reference frames used in flight mechanics for specifying the points and the vectors effectively, which are defined below.

2.1.1 Inertial Frame of Reference

An inertial frame of reference is a fixed frame that is accepted as relative to the remote starts. Newton’s second law is valid in this frame.

2.1.2 Earth-Fixed Frame of Reference

The Earth-fixed frame is fixed to the Earth, and the z-axis is directed to the center of the Earth. Therefore, the xy plane is tangent to the Earth’s surface. In flight mechanics, the Earth's rotation can be neglected relative to the inertial frame, so any Earth-fixed frame of reference can be accepted as an inertial frame.

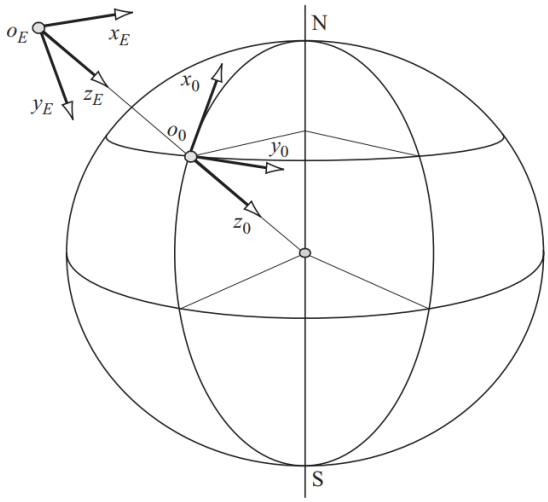

Figure 2.1. Earth-Fixed Frame [16]

2.1.3 Vehicle-Carried Vertical Frame of Reference

In this reference frame, the origin is attached to the vehicle (generally to the center of mass), and it moves with the vehicle as its name suggests. The z-axis is in the direction of the local gravitation vector. Conveniently, x-axis points to the vehicle's geographical North and y-axis points to its geographical East.

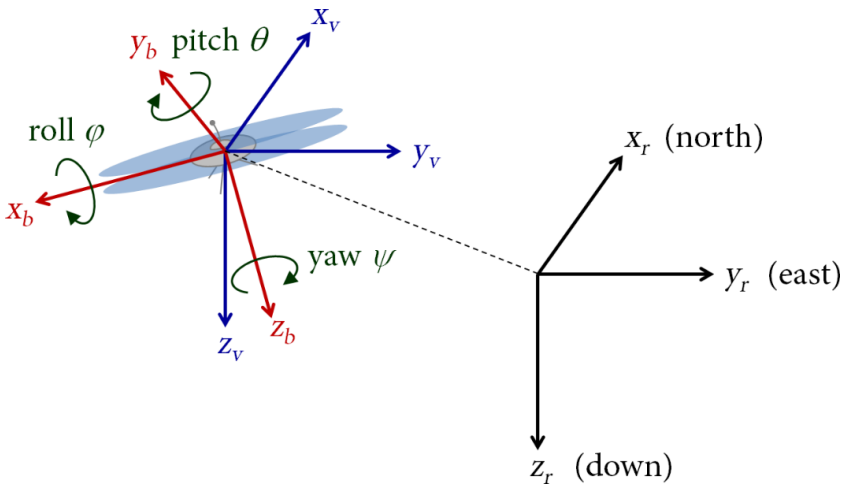

Figure 2.2. Vehicle-Carried Vertical Frame [17]

2.1.4 Body-Fixed Frame of Reference

This frame is linked to the vehicle body, and the origin is generally the center of mass. The body-fixed frame also moves with the body, and the x-axis is directed to the front, which is called the fuselage axis. Conveniently, the z-axis points downward of the vehicle and the y-axis points to the vehicle's right side. Figure 2.3 shows the frame fixed to the vehicle body.

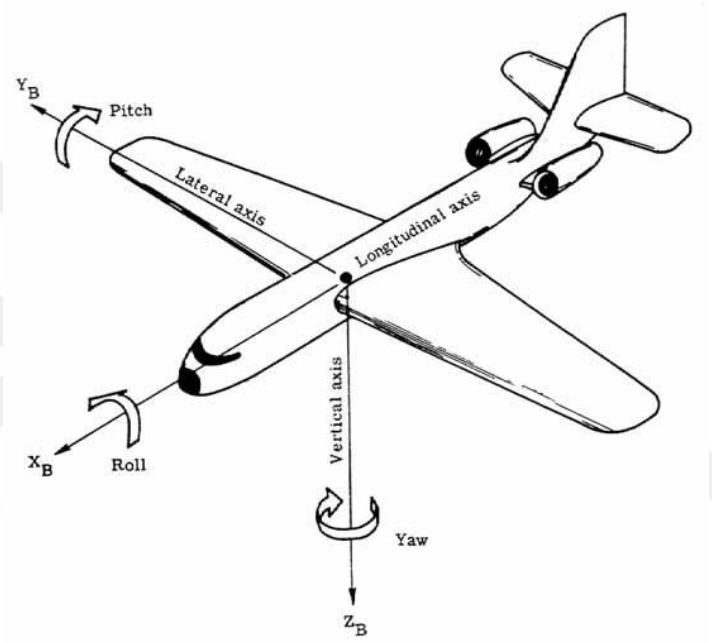

Figure 2.3. Body-Fixed Frame [18]

2.1.5 Aerodynamic Frame of Reference

In this reference frame, the x-axis is directed to the aerodynamic velocity of the vehicle. The z-axis is in the symmetrical plane of the vehicle, and the y-axis is chosen conveniently (orthogonally). This frame of reference becomes crucial in the necessity of modeling wind.

2.2 Euler Angles and Aircraft Attitude

The angles defined by the right-handed rotation about the three axes of a right-handed system are called Euler angles (ϕ, θ, ψ). These angles describe the orientation of any reference frame relative to another frame. Similarly, the attitude of an aircraft is defined as the angular orientation of the airframe axis with respect to the Earth-fixed axis. Therefore, Euler angles transform a vector expressed in a frame with respect to another frame by having consecutive rotations about the axes x, y, z.

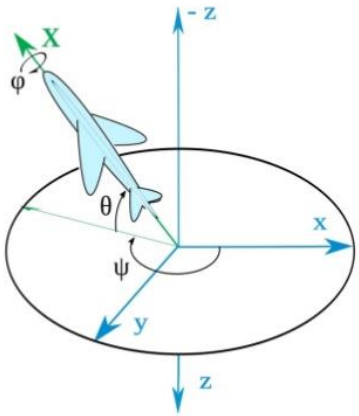

Figure 2.4. Euler Angles [17]

2.3 Axis Transformation

It is necessary to transform the motion variables expressed in reference axes to aircraft fixed axes. The transformation occurs in three consecutive rotations around the axes z, y, x, respectively, which is given in Table 2.1.

Table 2.1. Order of the Consecutive Rotations Needed for the Transformation

1st rotation	ψ roll angle	about axis z	1st intermediate axes
2nd rotation	θ pitch angle	about axis y	2nd intermediate axes
3rd rotation	ϕ yaw angle	about axis x	aircraft fixed axes

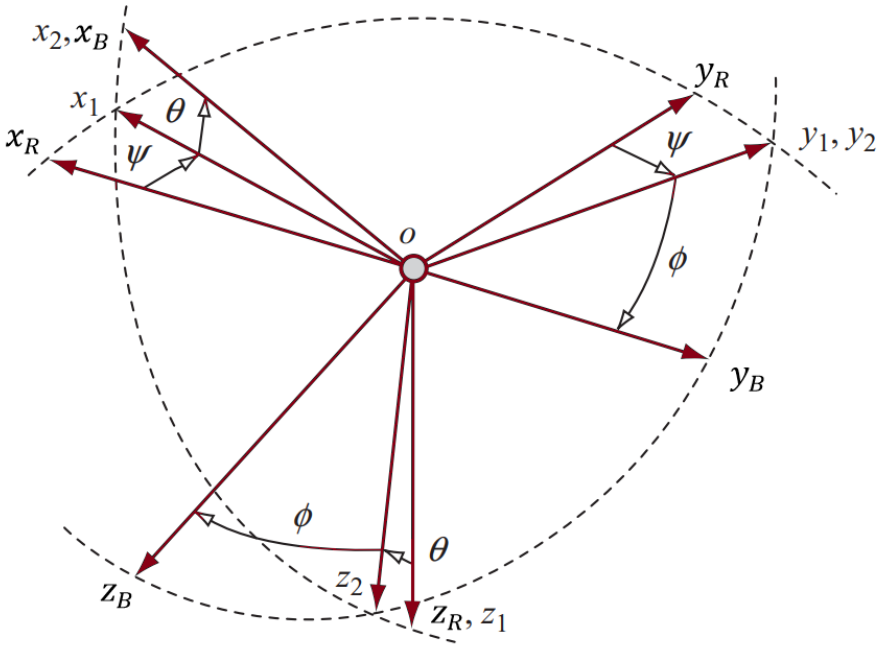

Figure 2.5. Consecutive Rotations of the Transformation

First intermediate axes ($OX_1Y_1Z_1$) are obtained after rotating the reference axes ($OX_RY_RZ_R$) by angle ψ about the z-axis of the reference axes. The transformation matrix for this single transformation is given below.

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = \begin{bmatrix} cos\psi & sin\psi & 0 \\ -sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_R \\ y_R \\ z_R \end{bmatrix} \tag{2.1}$$

Rotating the first intermediate axes by angle θ about the y-axis of the currently obtained first intermediate axes gives the second intermediate axes. The transformation matrix for this step is given below.

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} cos\theta & 0 & -sin\theta \\ 0 & 1 & 0 \\ sin\theta & 0 & cos\theta \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \tag{2.2}$$

Finally, aircraft fixed axes are obtained after the rotation of the x-axis of the second intermediate axes by angle ϕ. The transformation matrix for this step is given below.

$$\begin{bmatrix} x_B \\ y_B \\ z_B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & sin\phi \\ 0 & -sin\phi & cos\phi \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} \tag{2.3}$$

The transformation matrix between the reference axes and the aircraft fixed axes can be calculated by multiplying the three matrices found above.

$$\begin{bmatrix} x_B \\ y_B \\ z_B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & sin\phi \\ 0 & -sin\phi & cos\phi \end{bmatrix} \begin{bmatrix} cos\theta & 0 & -sin\theta \\ 0 & 1 & 0 \\ sin\theta & 0 & cos\theta \end{bmatrix} \begin{bmatrix} cos\psi & sin\psi & 0 \\ -sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_R \\ y_R \\ z_R \end{bmatrix} \tag{2.4}$$

$$T = \begin{bmatrix} cos\theta cos\psi & cos\theta sin\psi & -sin\theta \\ sin\phi sin\theta cos\psi - cos\phi sin\psi & sin\phi sin\theta sin\psi + cos\phi cos\psi & sin\phi cos\theta \\ cos\phi sin\theta cos\psi + sin\phi sin\psi & cos\phi sin\theta sin\psi - sin\phi cos\psi & cos\phi cos\theta \end{bmatrix}$$

2.4 Notations

Forces, moments, displacements, and velocities in the body frame are shown with their special letters in flight dynamics since the equations of motion are written in this coordinate system whose origin is fixed to the CG of the air vehicle. The flight dynamics variables are listed below in order of x, y, z directions.

- Linear velocity vector components are u, v, w.
- Force vector components are X, Y, Z.
- Angular velocity vector components are p, q, r.
- Moment vector components are L, M, N.

All components of the vectors are shown in Figure 2.6.

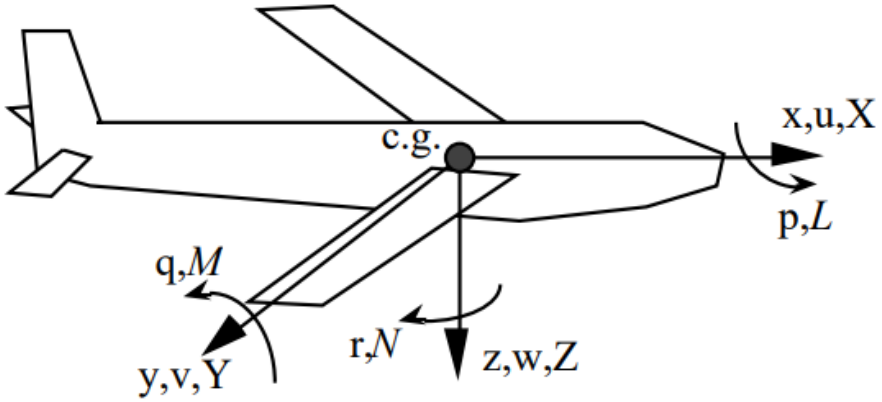

Figure 2.6. Flight Dynamics Variables [19]

2.5 General Equation of Motion of a Symmetric Air Vehicle

The general equations of motion are written in body-fixed coordinate system depending upon some assumptions, which are given below [20].

- The air vehicle is accepted as a rigid body, so the elastic deformation is ignored.
- It has a perfect symmetry about the longitudinal symmetry plane.
- The effects of spinning rotors are ignored.
- The effect of the Earth's rotation on the acceleration is neglected.

which can be categorized as translational and rotational dynamics.

2.5.1 General Force Equations

Translational equations of motion are derived based on Newton's second law. The net force acting on the rigid body is the sum of the aerodynamic, propulsive (thrust) and gravitational forces. Aerodynamic forces are exerted on the body by the air, which balances the weight so that the vehicle can fly smoothly. Propulsive force is

the force applied by the engine of air vehicles and this force is in the direction of x_B. Namely, these vectors are not in the same reference frame as illustrated in Figure 2.7, so the weight vector is transformed to the body frame by multiplying with the transformation matrix T_{BR} in order to bring them into the same reference frame. After transforming the weight vector, the net force is calculated by adding the vectors as Equation 2.5.

$$\sum F = ma$$

$$\sum F = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + T_{BR} \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + \begin{bmatrix} Thrust \\ 0 \\ 0 \end{bmatrix} \tag{2.5}$$

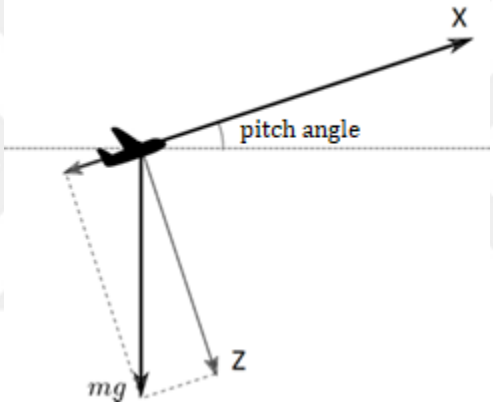

Figure 2.7. Forces Applied on the Rigid Body

The acceleration of the body is not directly equal to the derivatives of the linear velocities because the vehicle has a rotary motion besides. In this case, Coriolis Theorem suggests Equation 2.6 for the calculation of the acceleration.

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} + \begin{bmatrix} p \\ q \\ r \end{bmatrix} x \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{2.6}$$

Hence, Newton's second law can be stated as in Equation 2.7.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + T_{BR} \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} = m\left(\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} + \begin{bmatrix} p \\ q \\ r \end{bmatrix} x \begin{bmatrix} u \\ v \\ w \end{bmatrix}\right)$$
$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + mg \begin{bmatrix} -\sin\theta \\ \cos\theta \sin\phi \\ \cos\theta \cos\phi \end{bmatrix} = m \begin{bmatrix} \dot{u} - rv + qw \\ \dot{v} + ru - pw \\ \dot{w} + pv - qu \end{bmatrix} \quad (2.7)$$

Therefore, the translational dynamic equations are given in Equation 2.8.

$$\begin{aligned} X - mg\sin\theta &= m(\dot{u} - rv + qw) \\ Y + mg\cos\theta\sin\phi &= m(\dot{v} + ru - pw) \\ Z + mg\cos\theta\cos\phi &= m(\dot{w} + pv - qu) \end{aligned} \quad (2.8)$$

2.5.2 General Moment Equations

The rotational equations of motion are derived based on the rotational form of Newton's second law (Equation 2.9), where G is the moment, H is the angular momentum of the mass. Similarly, the Coriolis theorem suggests Equation 2.10 for the calculation of moment [20].

$$\sum G = \frac{\partial H}{\partial t} \quad (2.9)$$

$$\begin{bmatrix} L \\ M \\ N \end{bmatrix} = I \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} + \begin{bmatrix} p \\ q \\ r \end{bmatrix} + Ix \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad (2.10)$$

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{zz} \end{bmatrix} \quad (2.11)$$

The moment equations depend highly on the inertia matrix I_{3x3} (Equation 2.11) which is a feature of the air vehicle. If Equation 2.10 is extracted, the moment equations are obtained before simplification.

$$
\begin{aligned}
\mathrm{L} &= I_{xx}\dot{p} - I_{xz}(\dot{r} + pq) - \left(I_{yy} - I_{zz}\right)qr - I_{yz}(q^2 - r^2) - I_{xy}(\dot{q} - rp) \\
\mathrm{M} &= I_{yy}\dot{q} - I_{xz}(r^2 - p^2) - (I_{zz} - I_{xx})rp - I_{xz}(\dot{p} + qr) - I_{yz}(\dot{r} - pq) \\
\mathrm{N} &= I_{zz}\dot{r} - I_{xz}(\dot{p} - qr) - \left(I_{xx} - I_{yy}\right)pq - I_{xy}(p^2 - q^2) - I_{yz}(\dot{q} + rp)
\end{aligned}
\tag{2.12}
$$

For a symmetric airplane, $I_{xy} = I_{yz} = 0$ and $I_{xz} \ll I_z$ [16]. Thus, if the related terms are assumed to be 0 and canceled, the simplified moment vector components can be calculated using Equation 2.13.

$$
\begin{aligned}
\mathrm{L} &= I_{xx}\dot{p} - \left(I_{yy} - I_{zz}\right)qr \\
\mathrm{M} &= I_{yy}\dot{q} - (I_{zz} - I_{xx})rp \\
\mathrm{N} &= I_{zz}\dot{r} - \left(I_{xx} - I_{yy}\right)pq
\end{aligned}
\tag{2.13}
$$

2.6 Aerodynamic Model

The aerodynamic forces and moments in the equations of motion of a symmetric air vehicle given by Equations 2.8 and 2.13 depend on the translational, rotational velocities and their derivatives. The general equations are given below [23].

$$
\begin{aligned}
Force &= \frac{1}{2}\rho S V^2 C_F \\
Moment &= \frac{1}{2}\rho S b V^2 C_M \\
V &= \sqrt{u^2 + v^2 + w^2}
\end{aligned}
\tag{2.14}
$$

ρ is the air density, which changes according to the altitude, S is the reference surface (generally the surface of the wing), b is the span of the wing (the vehicle diameter). C_F and C_M are nondimensional aerodynamic coefficient vectors, and their components are called coefficients of lift force, drag force, rolling moment, pitching moment and yawing moment. Note that C_D and C_L are negative because drag and lift components of the aerodynamic forces are in the reverse direction of the axes x and

z of the body frame, respectively. Drag reduces the speed by opposing the velocity vector, and lift tries to balance the weight in the physical meaning.

$$C_F = \begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix} = \begin{bmatrix} \cos\alpha\cos\beta & \sin\beta & \sin\alpha\cos\beta \\ -\cos\alpha\sin\beta & \cos\beta & -\sin\alpha\sin\beta \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} \begin{bmatrix} -C_D \\ C_y \\ -C_L \end{bmatrix}$$
$$C_M = \begin{bmatrix} C_l \\ C_m \\ C_n \end{bmatrix} \quad (2.15)$$

These aerodynamic coefficients are affected remarkably by the aerodynamic angles α (angles of attack) and β (sideslip angle), defined as the orientation of the aerodynamic velocity with respect to the air vehicle. They are formulated in Equation 2.16.

$$\alpha = \tan^{-1}\frac{w}{u}$$
$$\beta = \tan^{-1}\frac{v}{V} \quad (2.16)$$

The geometry of the air vehicle has a massive impact on the aerodynamic coefficients. Therefore, the air vehicles are designed by taking these coefficients into account. The aerodynamic coefficients are functions of angular velocities (p, q, r), aerodynamic angles (α, β) and control surface deflections $(\delta_a, \delta_e, \delta_r)$. The coefficients depending on angular velocities and aerodynamic angles are called stability derivatives. On the other hand, air vehicles have control surfaces to carry out turnings, and these surfaces are controlled by deflection angles. In flight dynamics, $\delta_a, \delta_e, \delta_r$ are aileron, elevator and rudder deflections and their coefficients are called control derivates. Once for all, the aerodynamic coefficients are given in terms of stability and control derivatives in Equation 2.17. In this work, they are accepted as linear functions, and the coefficient values used in the simulations are given in Appendix C.

$$
\begin{aligned}
C_x &= -C_D \\
C_y &= C_{y_\beta}\beta + C_{y_{\delta_r}}\delta_r \\
C_z &= -C_{L_\alpha}\alpha + C_{L_{\delta_e}}\delta_e \\
C_l &= C_{l_p}\frac{d}{2V}p + C_{l_{\delta_a}}\delta_a \\
C_m &= C_{m_\alpha}\alpha + C_{m_q}\frac{d}{2V}q + C_{m_{\delta_e}}\delta_e \\
C_n &= C_{n_\beta}\beta + C_{n_r}\frac{d}{2V}r + C_{n_{\delta_r}}\delta_r
\end{aligned}
\tag{2.17}
$$

2.7 3D Projection of Proportional Navigation Guidance

In this study, 3D Proportional Navigation Guidance (PNG) law is used to guide the air vehicle. PNG is a trendy and effective guidance method in homing missiles. Theoretically, PNG generates acceleration commands which are perpendicular to the line-of-sight (LOS), proportional to the line-of-sight rate and closing velocity [21]. The formula for acceleration commands is given in Equation 2.18, where n_c is acceleration command, N is effective navigation ratio, V_c is closing velocity and $\dot{\lambda}$ is LOS rate.

$$
n_c = NV_c\dot{\lambda} \tag{2.18}
$$

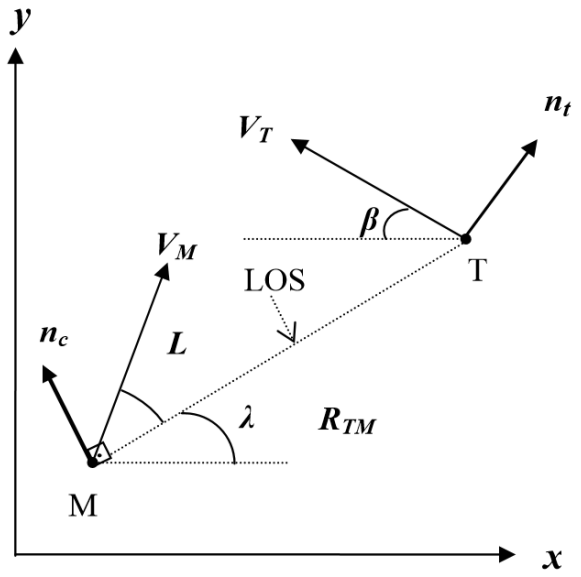

Figure 2.8. 2D Proportional Navigation [22]

3D Proportional Navigation is obtained by projecting the 2D proportional navigation onto three separate planes (S_{xy}, S_{xz}, S_{yz}). In other words, the fundamental idea is the same as in 2D PNG [21]. For instance, the LOS rate is calculated for three planes separately for 3D PNG [22]. Note that positions, velocities and the other vectors have three components from now on. In short, the closing velocity V_c and the LOS rate $\dot{\lambda}$ is calculated to be able to complete the items in Equation 2.18.

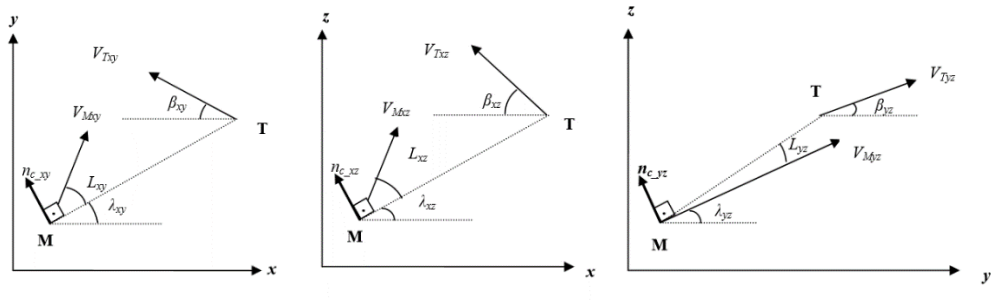

Figure 2.9. Proportional Navigation Projections onto the Planes S_{xy}, S_{xz}, S_{yz} [22]

Firstly, the distance between the air vehicle and the target is calculated.

$$R = \sqrt{{P_{TM}}_x^2 + {P_{TM}}_y^2 + {P_{TM}}_z^2} \tag{2.19}$$

The LOS angles are calculated for each plane in Equation 2.20.

$$\begin{aligned} \lambda_{xy} &= \tan^{-1} \frac{P_{TM_y}}{P_{TM_x}} \\ \lambda_{xz} &= \tan^{-1} \frac{P_{TM_z}}{P_{TM_x}} \\ \lambda_{yz} &= \tan^{-1} \frac{P_{TM_z}}{P_{TM_y}} \end{aligned} \tag{2.20}$$

The LOS rates are derived from Equation 2.21.

$$\begin{aligned} \dot{\lambda_{xy}} &= \frac{P_{TM_x} V_{TM_y} - P_{TM_y} V_{TM_x}}{P_{TM_x}^2 + P_{TM_y}^2} \\ \dot{\lambda_{xz}} &= \frac{P_{TM_x} V_{TM_z} - P_{TM_z} V_{TM_x}}{P_{TM_x}^2 + P_{TM_z}^2} \\ \dot{\lambda_{yz}} &= \frac{P_{TM_y} V_{TM_z} - P_{TM_z} V_{TM_y}}{P_{TM_y}^2 + P_{TM_z}^2} \end{aligned} \tag{2.21}$$

The closing velocities are calculated for the planes S_{xy}, S_{xz}, S_{yz}.

$$\begin{aligned} V_{c_{xy}} &= -\frac{P_{TM_x} V_{TM_x} + P_{TM_y} V_{TM_y}}{\sqrt{P_{TM_x}^2 + P_{TM_y}^2}} \\ V_{c_{xz}} &= -\frac{P_{TM_x} V_{TM_x} + P_{TM_z} V_{TM_z}}{\sqrt{P_{TM_x}^2 + P_{TM_z}^2}} \\ V_{c_{yz}} &= -\frac{P_{TM_y} V_{TM_y} + P_{TM_z} V_{TM_z}}{\sqrt{P_{TM_y}^2 + P_{TM_z}^2}} \end{aligned} \tag{2.22}$$

Thereby, the acceleration commands perpendicular to the corresponding LOS for the three planes are expressed below.

$$\begin{aligned} n_{c_{xy}} &= NV_{c_{xy}}\dot{\lambda_{xy}} \\ n_{c_{xz}} &= NV_{c_{xz}}\dot{\lambda_{xz}} \\ n_{c_{yz}} &= NV_{c_{yz}}\dot{\lambda_{yz}} \end{aligned} \tag{2.23}$$

The acceleration commands in x, y, z axes can be computed from the combinations of the accelerations perpendicular to the corresponding LOS. The geometric relationship can be seen in Figure 2.9.

$$\begin{aligned} a_x &= -n_{c_{xy}} \sin\lambda_{xy} - n_{c_{xz}} \sin\lambda_{xz} \\ a_y &= n_{c_{xy}} \sin\lambda_{xy} - n_{c_{yz}} \sin\lambda_{yz} \\ a_z &= n_{c_{xz}} \sin\lambda_{xz} + n_{c_{yz}} \sin\lambda_{yz} \end{aligned} \tag{2.24}$$

2.8 Control Actuation System

Control Actuation Systems (CAS) are used in flight mechanics to direct the air vehicles by controlling the fin (control surfaces) positions. The guidance system takes the position and velocity of the target and the interceptor and generates the command acceleration. The reference signal to the control unit is the difference between the command and the actual acceleration. At this moment, the system controls the control surface deflections ($\delta_a, \delta_e, \delta_r$) and the thrust force ($\delta_t$) which is coming from the engine in order to make the air vehicle go towards the target. The fin angles significantly impact aerodynamic forces, and the effects are included in Equation 2.17. The whole system is illustrated in Figure 2.10.

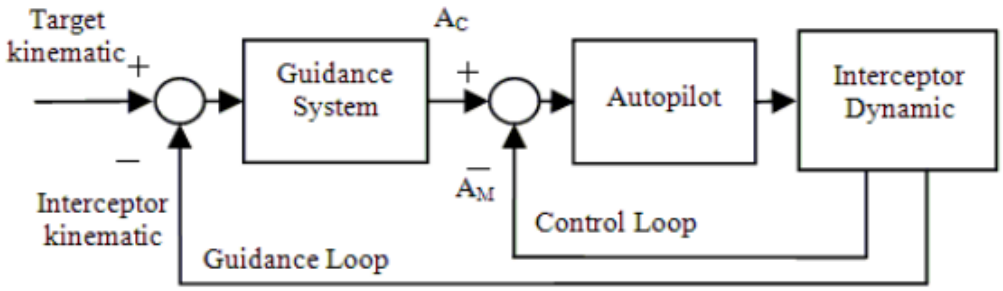

Figure 2.10. Guidance and Control Loops in CAS [24]

The states $X = [x, y, z, u, v, w, \phi, \theta, \psi, p, q, r]$ and the control variables $\delta = [\delta_e, \delta_t, \delta_a, \delta_r]$ define the system dynamics, which can be formulated as Equation 2.25. There are four separate control loops for each control variable.

$$\dot{X} = f(X, \delta) \tag{2.25}$$

Whereas the guidance system generates the desired accelerations, the actual accelerations are calculated by dividing the net force by the weight using Equation 2.5. The difference between them creates the error signal to the control unit, and the control variables are manipulated accordingly.

2.9 Demonstration of the Model

The mathematical model of an air vehicle, guidance and control actuation systems are integrated and built on MATLAB Simulink. The overall model and the subsystems are examined in this section, and some experiments are conducted in order to show the interactions between the subsystems operate correctly. The overall Simulink model can be seen in Figure A.1. The model's upper side represents the mathematical model of an air vehicle, which interacts with the guidance system and the control actuation system blocks.

The guidance system generates the command accelerations and gives them to the control unit. Then, the control actuation system produces necessary control variables to follow the command signals and gives them to the air vehicle, as shown in Figure 2.11.

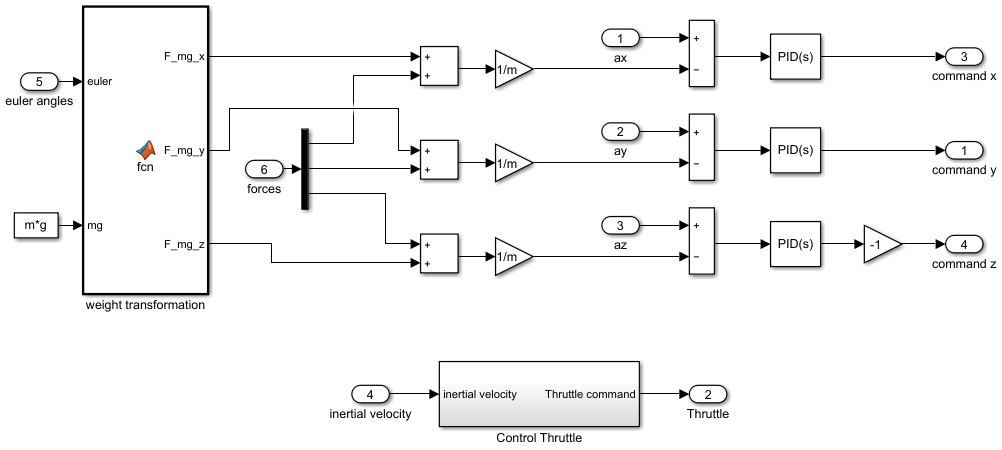

Figure 2.11. Control Actuation System Block

A basic scenario is simulated and the outputs are examined in order to see if the trajectory is compatible with the outputs. The air vehicle parameters used in this simulation are shown in Appendix C. A target is located in $(0, 0, 0)$, and the air vehicle starts moving from $(-5000,\ 2000,\ 1000)$ with an initial velocity in x direction. The air vehicle is supposed to reach the target making the correct maneuver. As a result of the simulation, the flight trajectory shown in Figure 2.12 is obtained. The figure shows xy and xz views of the trajectory besides the 3D view, and it can be seen that the air vehicle reaches the target successfully.

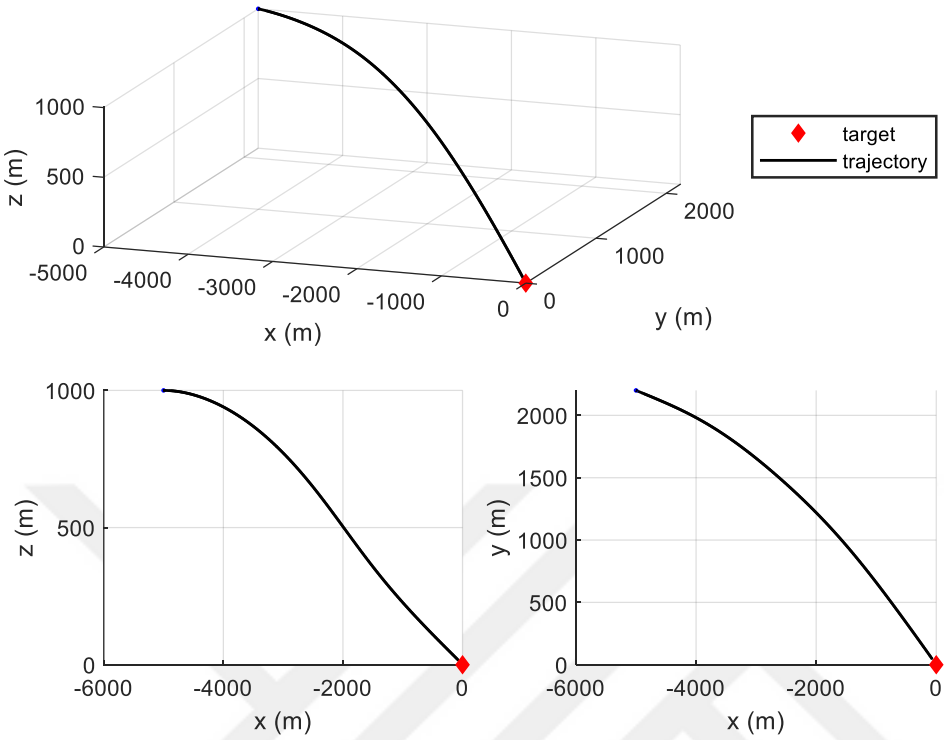

Figure 2.12. The flight trajectory of the air vehicle

The Euler angles of the air vehicle are given in Figure 2.13. According to the figure, the roll angle is almost zero throughout the flight, which is the desired condition for this simulation. In other words, the aileron is controlled in a way that the roll angle remains zero. The pitch angle has negative values throughout the flight, and it makes sense because the air vehicle is going down towards the target, which can be seen in xz view of the trajectory. The air vehicle is also turning slightly right side towards the target, and the yaw angle seems compatible with this movement.

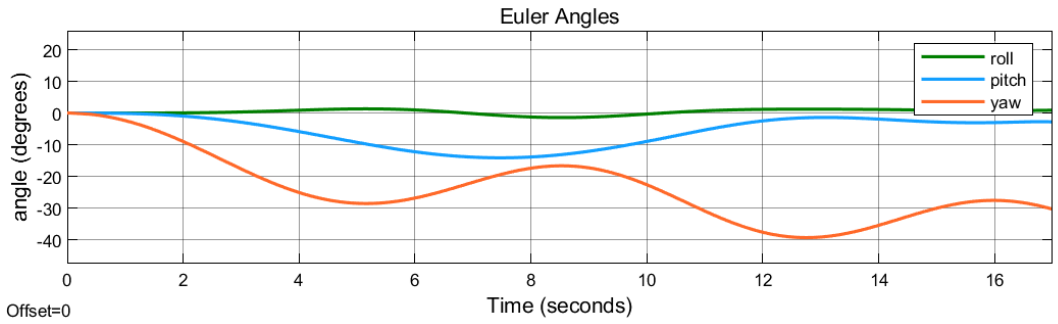

Figure 2.13. Euler Angles of the Air Vehicle

As stated above, the guidance system produces the command accelerations, and the CAS makes the actual accelerations follow these signals. Figures 2.14 and 2.15 show actual and desired accelerations for y and z axes. The figures show that the actual accelerations are pretty following the command accelerations so that the air vehicle reaches the target successfully.

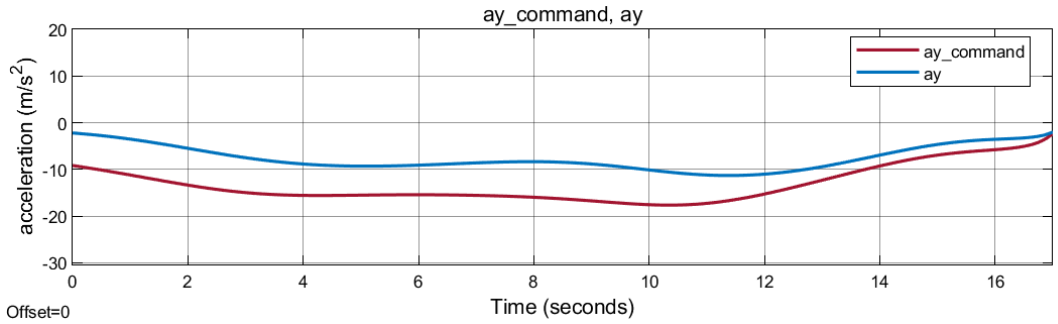

Figure 2.14. The Actual and Desired Accelerations in the y-axis

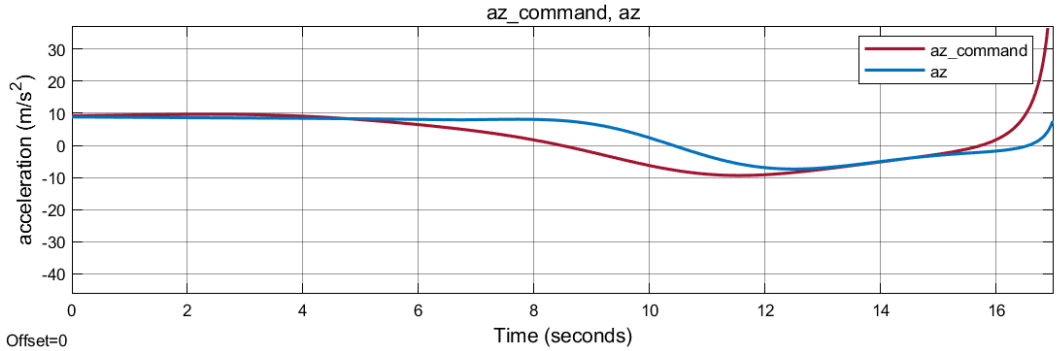

Figure 2.15. The Actual and Desired Accelerations in the z-axis

CHAPTER 3

MIDCOURSE PATH PLANNING

3.1 Problem Definition

The midcourse path planning problem is to find the optimum path from the starting point to the target point in a 3D environment under some constraints. Finding a continuous path is an infinite-dimensional problem in which it is aimed to get a space curve that is defined by three functions in 3D space. These functions are x(t), y(t), z(t) and the main task is to find these functions. Finding the best path according to some objectives is substantially a multi-objective optimization problem which is formulated as follows:

minimize

$$w_1 * J_1(x(t), y(t), z(t)) + w_2 * J_2(x(t), y(t), z(t)) + w_3 * J_3(x(t), y(t), z(t))$$

such that

$$J_1 = \int_{t=t_0}^{t=t_f} \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2 + \left(\frac{dz}{dt}\right)^2}\, dt$$

$$J_2 = \int_{t=t_0}^{t=t_f} \left(\left|\frac{d^2x}{dt^2}\right| + \left|\frac{d^2y}{dt^2}\right| + \alpha(\ddot{z}) \left|\frac{d^2z}{dt^2}\right| \right) dt$$

$$J_3 = \int_{t=t_0}^{t=t_f} f(z)\, dt; \quad f(z) = \begin{cases} 0, & z < h \\ z^2, & z \geq h \end{cases}$$

$$\alpha(\ddot{z}) = \begin{cases} 0, & \ddot{z} < 0 \\ a, & \ddot{z} \geq 0 \end{cases}$$

$$0 \leq w_1, w_2, w_3 \leq 1$$

$$w_1 + w_2 + w_3 = 1$$

$$d_{start}(x(t), y(t), z(t)) \geq d_{min_{start}}$$

$$d_{turn}(x(t), y(t), z(t)) \geq d_{min_{turn}}$$

$$d_{final}(x(t), y(t), z(t)) \geq d_{min_{final}}$$

$$\theta(x(t), y(t), z(t)) \leq \theta_{maxturn}$$

$$d_{climb}(x(t), y(t), z(t)) \geq d_{min_{climb}}$$

$$d_{over}(x(t), y(t), z(t)) \geq d_{min_{over}}$$

$$d_{dive}(x(t), y(t), z(t)) \geq d_{min_{dive}}$$

where

J_1 is the length of the path

J_2 is the fuel consumption

J_3 is the risk of being detected

The objective functions J_1, J_2 and J_3 are length, fuel consumption and risk of being detected, and they are mathematically defined above. J_1 simply calculates the arc length of a continuous path. J_2 estimates the fuel consumption of an air vehicle from the accelerations. The risk of being detected of an air vehicle is estimated as J_3 which increases with the probability of being caught by radars. Piecewise integral brings the condition that there is no risk of being detected cost until the height gets to a certain value h. In the meantime, angle and distance constraints are in the nature of the path planning problem, stated in the formulation. Air vehicles should not make very sharp turns that their dynamics cannot handle, which brings an angle constraint. Also, they should not have any maneuver until it goes a predefined straight distance. All the air vehicle subsystems start to work properly after the turning, climbing, and diving maneuvers. Also, since the problem is 3D path planning so that the trajectory might pass over obstacles, the climbing and diving parts bring some extra constraints. The constraints that might make a path infeasible in this work are listed below:

1) *Distance constraint on the first waypoint:* The minimum distance that it has to go straight after the starting point until the first movement is $d_{min_{start}}$.
2) *Distance constraint between turnings:* The minimum distance between two turnings is $d_{min_{turn}}$.
3) *Distance constraint on the final waypoint:* The minimum distance that it has to go straight before the target point is $d_{min_{final}}$.
4) *Angle constraint on turnings:* The maximum turning angle is $\theta_{maxturn}$. This constraint is not checked in the Initial Population Creation part because the turning angles are generated between the angle limits calculated according to this constraint when generating a path. However, this constraint might be violated during crossover and mutation operations.
5) *Distance constraint before climbing:* The minimum distance that it has to go straight before the climbing starts is $d_{min_{climb}}$.
6) *Distance constraint over an obstacle:* The minimum distance that it has to go straight over the obstacle after finishing climbing and before starting diving is $d_{min_{over}}$.
7) *Distance constraint after diving:* The minimum distance that it has to go straight after the diving ends is $d_{min_{dive}}$.

3.2 Discretization of the Problem

The midcourse path planning problem is infinite-dimensional. Computers cannot solve it in finite time without discretizing the problem, so a finite-dimensional approximation is needed. In order to make it a finite-dimensional problem, a path is considered as a sequence of a number of points that defines the trajectory between the starting point and target point. These points are called waypoints, and each of them has x, y, z coordinates to state a position in 3D space. Figure 3.1 shows a

continuous path and many waypoints on it, in which the sequence of the waypoints expresses the same trajectory as the continuous path.

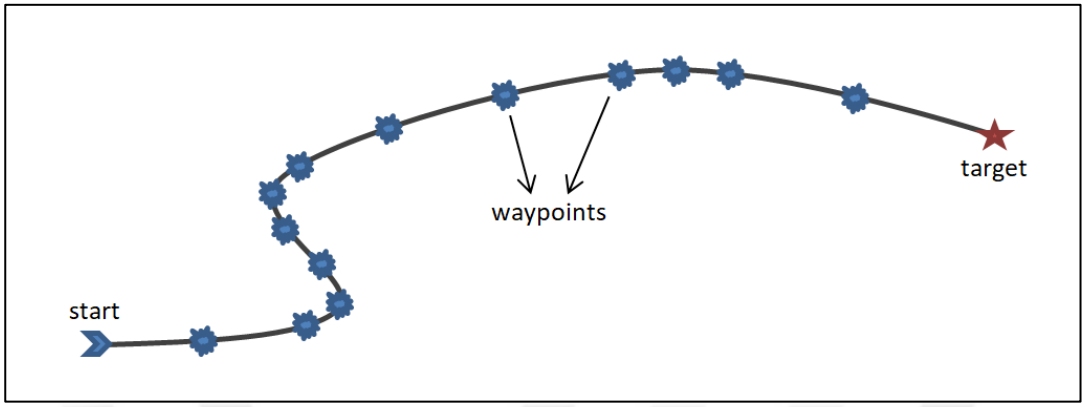

Figure 3.1. A Path that is Expressed with a Sequence of Waypoints

It is important to express a path with the minimum number of waypoints. Notice that more waypoints are located on the curved parts of the path, and fewer waypoints are enough to describe straight parts of the path in Figure 3.1. Also, the number of waypoints is not constant. Therefore, coordinate positions of each of the varying number of waypoints are the variables to be optimized in this problem.

3.3 Finite-Dimensional Problem

In the mathematical representation of the infinite-dimensional problem, integrals are approximated using trapezoid method of numerical integration, and derivates are replaced with finite-difference operators for discretization. Note that the intervals in trapezoid integral approximation are not uniform, so all the items in equations are multiplied with the lengths of specific intervals. After that, the aim becomes to find the positions of the waypoints which minimize the weighted sum of the objective functions, where t_f is still an unknown. The formulation of the optimization problem is given below.

given

$$q = < x_1, y_1, z_1, x_2, y_2, z_2, \dots, x_n, y_n, z_n, t_f >$$

minimize

$$w_1 * J_{1d}(q) + w_2 * J_{2d}(q) + w_3 * J_{3d}(q)$$

such that

$$J_{1d} = \frac{1}{2}\Big[(t_1 - t_0)\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2 + (z_1 - z_0)^2} + 2(t_2 - t_1)\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} + 2(t_3 - t_2)\sqrt{(x_3 - x_2)^2 + (y_3 - y_2)^2 + (z_3 - z_2)^2} + \cdots + (t_n - t_{n-1})\sqrt{(x_n - x_{n-1})^2 + (y_n - y_{n-1})^2 + (z_n - z_{n-1})^2}\Big]$$

$$J_{2d} = \tfrac{1}{2}[(t_1 - t_0)(|x_2 - 2x_1 + x_0| + |y_2 - 2y_1 + y_0| + \alpha(a_z)|z_2 - 2z_1 + z_0|) + 2(t_2 - t_1)(|x_3 - 2x_2 + x_1| + |y_3 - 2y_2 + y_1| + \alpha(a_z)|z_3 - 2z_2 + z_1|) + 2(t_3 - t_2)(|x_4 - 2x_3 + x_2| + |y_4 - 2y_3 + y_2| + \alpha(a_z)|z_4 - 2z_3 + z_2|) + \cdots + (t_{n-1} - t_{n-2})(|x_n - 2x_{n-1} + x_{n-2}| + |y_n - 2y_{n-1} + y_{n-2}| + \alpha(a_z)|z_n - 2z_{n-1} + z_{n-2}|)]$$

$$J_{3d} = \frac{1}{2}[(t_1 - t_0)f(z_1) + 2(t_2 - t_1)f(z_2) + 2(t_3 - t_2)f(z_3) + \cdots + (t_n - t_{n-1})f(z_n)]; \quad f(z) = \begin{cases} 0, & z < h \\ z^2, & z \geq h \end{cases}$$

$$\alpha(a_z) = \begin{cases} 0, & a_z < 0 \\ a, & a_z \geq 0 \end{cases}$$

$$0 \leq w_1, w_2, w_3 \leq 1$$

$$w_1 + w_2 + w_3 = 1$$

$$d_{start}(q) \geq d_{min_{start}}$$

$$d_{turn}(q) \geq d_{min_{turn}}$$

$$d_{final}(q) \geq d_{min_{final}}$$

$$\theta(q) \leq \theta_{maxturn}$$

$$d_{climb}(q) \geq d_{min_{climb}}$$

$$d_{over}(q) \geq d_{min_{over}}$$

$$d_{dive}(q) \geq d_{min_{dive}}$$

Paths might have turnings, climbings and divings, and many waypoints are needed to describe such detailed maneuvers. In this work, points on the maneuvers are not counted as waypoints to have less variables in the optimization process, enhancing the convergence time drastically. The transitions between waypoints are filled with realistic maneuvers and stored in the waypoint structures programmatically. For example, climbing and diving maneuvers are required if there is an obstacle between two waypoints. Therefore, the turning, climbing and diving maneuvers are calculated according to the locations of the adjacent waypoints and all the constraints of these maneuvers given in the formula are checked at every stage of the algorithm. Calculations of the maneuvers are elaborated in the next sections.

3.4 Calculation of the Maneuvers of 3D Paths

3.4.1 Calculation of the Turning Maneuvers

Since the shortest transition between two points is looked for, a variant of the Dubins [25] path approach is used in this work. In geometry, Dubins path typically refers to the shortest curve that connects two points [26]. It is a very fast geometric and analytical method, which simplifies the problem. It is quite good that it makes the calculations fast since so many calculations are required in the optimization process. All the calculation steps are given in detail in this section.

Suppose three waypoints are united with two straight lines. In that case, the resulting path is not enough to indicate a complex trajectory, and the turning in the middle is not realistic to any existing air vehicle, which is shown in Figure 3.2. Therefore, smooth transitions are crucial in such maneuvers. Turning maneuvers are calculated by placing circles between two waypoints. The circle's radius might be adaptive so

that it changes up to the distances between waypoints, but it is taken constant in this work. The air vehicle dynamics are taken into account when selecting the radius of the circle because G-force that the air vehicles are exposed on turning maneuvers will be affected when the radius changes. In other words, the radius cannot be smaller than a threshold value that the dynamics cannot handle, so it should be selected carefully.

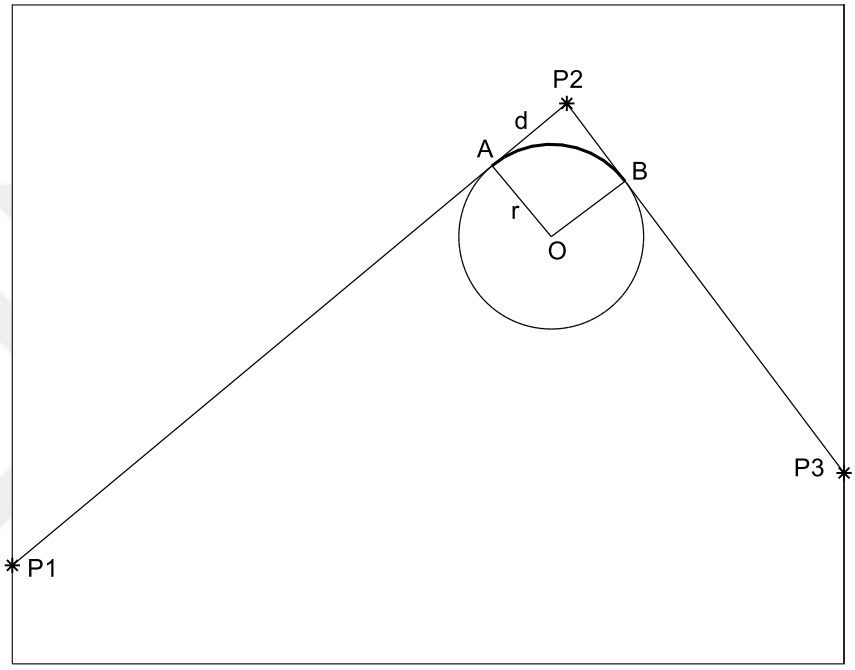

Figure 3.2. Placing a Circle with a Constant Radius Between Waypoints

Besides, most air vehicles have GPS to understand their exact positions, and it is crucial to follow a desired path. However, there is always an error in the position that GPS gives. The effect of the GPS error should be modeled to generate paths that can be followed safely in any case. Therefore, the path has widened the length of maximum GPS error in the lateral direction, and the error polygon of the path is created. For instance, if the actual path does not collide with an obstacle but the polygon does, that is accepted as a collision because of the GPS error.

In Figure 3.2, the points on the arc are calculated, and they indicate the smooth turning between two waypoints instead of the single waypoint. The calculation steps are explained below.

$$\begin{aligned} V_{in} &= P_2 - P_1 \\ V_{out} &= P_3 - P_2 \\ \theta_{difference} &= \text{atan}(V_{out}) - \text{atan}(V_{in}) \end{aligned} \tag{3.1}$$

The angle θ, which is $\measuredangle P_1 P_2 O$, is used to calculate the distance d since $|AO|$ is perpendicular to V_{in} in Equation 3.2.

$$\begin{aligned} \theta &= \frac{180 - \theta_{difference}}{2} \\ d &= \frac{r}{\tan\theta} \end{aligned} \tag{3.2}$$

Position A is calculated by subtracting the unit vector of V_{in} with magnitude d from the position P_2.

$$\begin{aligned} u_{V_{in}} &= \frac{V_{in}}{|V_{in}|} \\ A &= P_2 - d * u_{V_{in}} \end{aligned} \tag{3.3}$$

In order to get the position of the center O, the vector $u_{V_{in}}$ is rotated by 90 degrees and multiplied by the radius, and subtracted from position A.

$$\begin{aligned} n &= \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} * u_{V_{in}} \\ O &= A - r * n \end{aligned} \tag{3.4}$$

The points on the arc can be calculated by rotating the vector n by the step angle α and adding these vectors with magnitude r to the center of the circle. If the number

of divisions is constant, the step angle is calculated by dividing the arc angle by the division number N. As the number N increases, the resolution of the arc increases.

$$\alpha = \frac{\theta_{difference}}{N} * k, \;\; k = 1,2,3, \ldots, \mathrm{N}$$
$$arcpoints = O + r * \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} * n \tag{3.5}$$

In order to create the error polygon, only a minor part of the last step is altered. The distance r multiplied by the rotation matrix in Equation 3.5 is modified, and the points of two counter sides are obtained as expressed in Equation 3.6. Inner and outer arc points of the error polygon are depicted in Figure 3.3.

$$\alpha = \frac{\theta_{difference}}{N} * k, \;\; k = 1,2,3, \ldots, \mathrm{N}$$
$$outer\ arcpnts = O + (r + error\ distance) * \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} * n \tag{3.6}$$
$$inner\ arcpnts = O + (r - error\ distance) * \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} * n$$

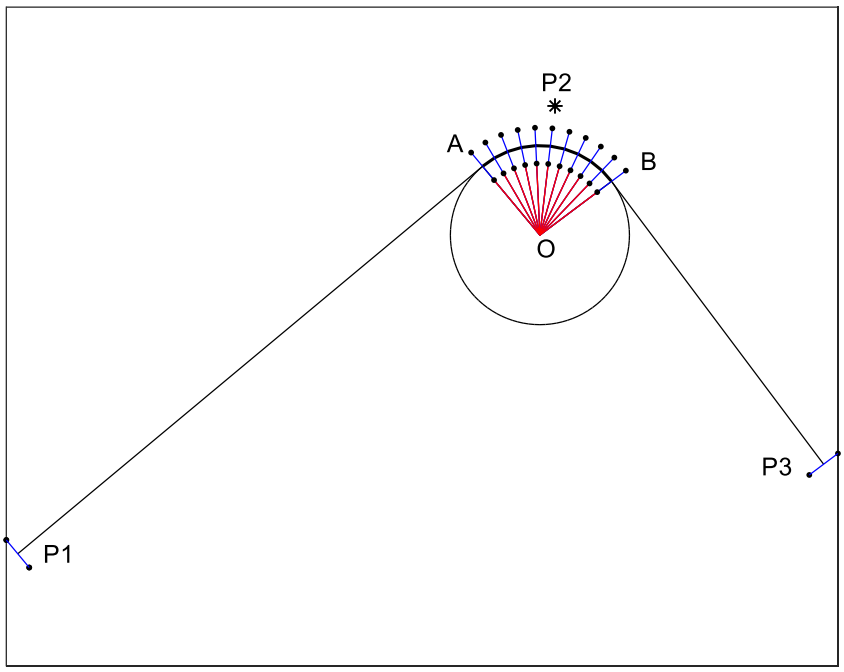

Figure 3.3. Calculating Arc Points of the Error Polygon

Also, initial and final pairs of the error polygon are calculated using a similar procedure. Figure 3.4 shows the resulting error polygon.

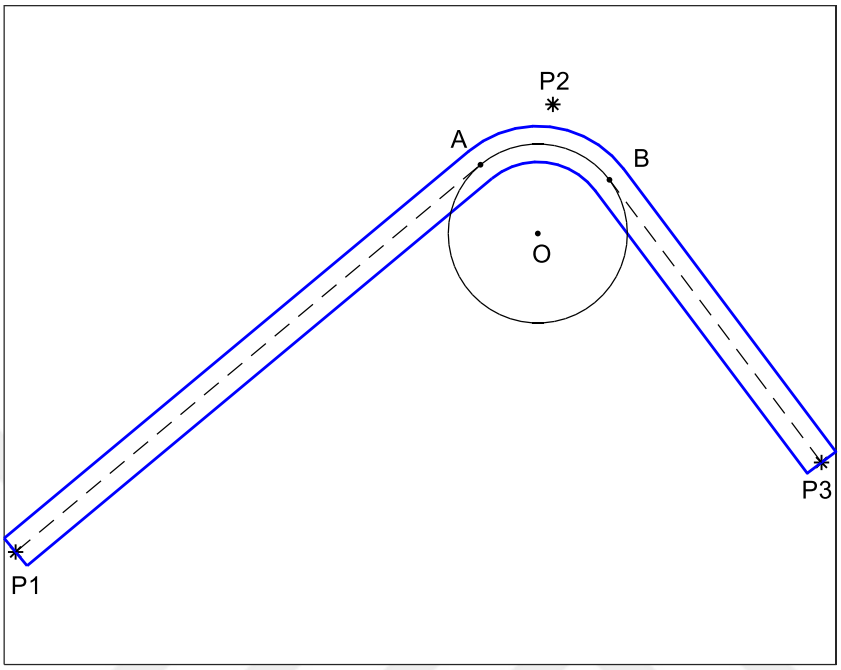

Figure 3.4. Error Polygon with Smooth Turning Between Two Waypoints

3.4.2 Calculation of the Climbing and Diving Points

If there are some places that the polygon intersects with an obstacle, the part including the obstacle is found by calculating the edges of the intersection in the direction of V_{base} and the area between these edges is accepted as an obstacle.

$$V_{base} = P_3 - P_2$$
$$\theta_{base} = \text{atan}\,(V_{base}) \tag{3.7}$$

All points that define the intersection area are projected into the vector V_{base} and the nearest and the outmost points to P_2 are found by comparing all the distances. If the

end of the error polygon is already inside the obstacle, the error polygon is extended in the same direction until getting out of the obstacle.

$$\begin{aligned} V_i &= intersectionpoint - P_2 \\ \theta_i &= \text{atan}\,(V_i) \\ distance &= |V_i| * \cos\,(\theta_{base} - \theta_i) \end{aligned} \tag{3.8}$$

After the minimum and the maximum distances are selected, the nearest and the outmost positions are calculated using geometry in Equation 3.9.

$$\begin{aligned} u_{V_{base}} &= \frac{V_{base}}{|V_{base}|} \\ nearest &= P_2 + mindistance * u_{V_{base}} \\ outmost &= P_2 + maxdistance * u_{V_{base}} \end{aligned} \tag{3.9}$$

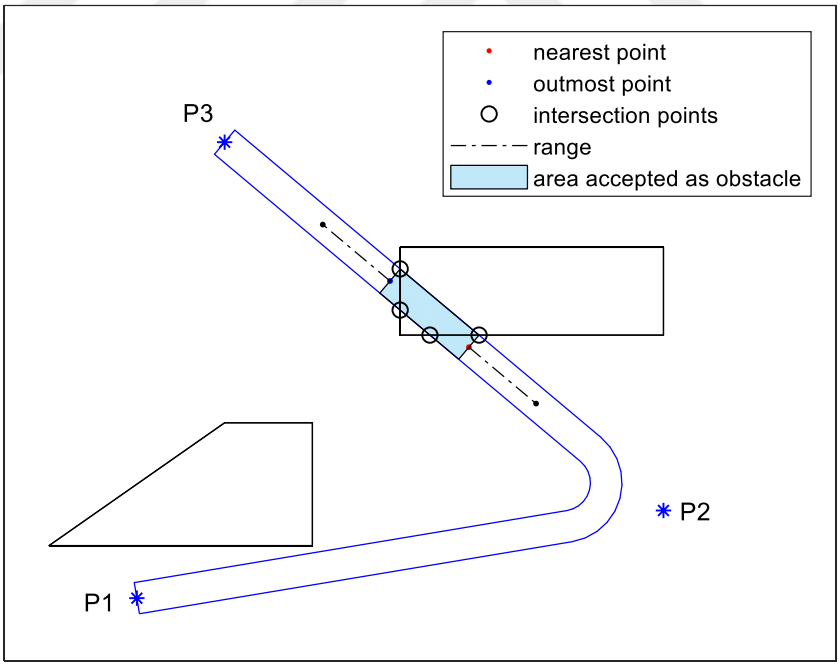

Figure 3.5. Calculation of Climbing and Diving Points

Climbing and diving angles are assumed to be constant in this work. If the altitude of the obstacle that the trajectory passes over is known, the distance it takes when climbing and diving in 2D can be calculated using the following formula. If the obstacle shapes are detailed, the maximum altitude in the accepted area as the obstacle is taken into account.

$$range = \frac{h}{\tan \theta_{climb}} \tag{3.10}$$

Simply, the nearest and the outmost points are climb-end and dive-start points, respectively. The point that climbing starts and the point that diving ends can be calculated by adding $range$ to the nearest and the outmost points in the direction of V_{base}, as shown in Equation 3.10. All these points are depicted in Figure 3.5. The resulting climbing and diving points are illustrated in Figure 3.6.

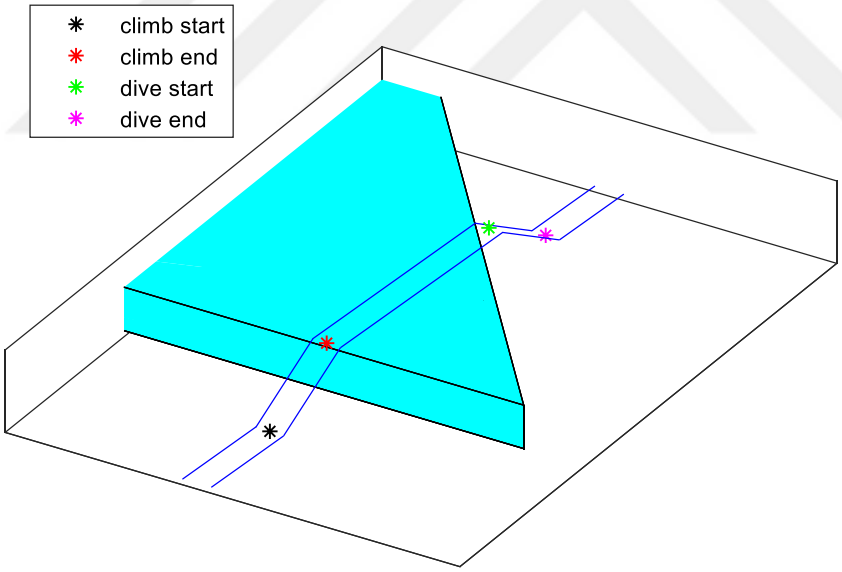

Figure 3.6. Climbing and Diving over an Obstacle

3.5 Optimization Method

Choosing an optimization method is not an easy task because there are plenty of them in the literature. The necessities of the problem should be examined carefully in order to choose the correct method. There are many variables to be optimized in the path planning problem, so global optimality is crucial since the risk of getting stuck in local minima is pretty high. Also, it requires flexibility since the number of optimization variables is not constant in this problem. Therefore, one of the most famous algorithms, the Real Coded Genetic Algorithm, is used in this work. It is very robust with respect to local minima and maxima. Problem dependent operators can be defined in GA. It is one of the most significant advantages because it means the optimization can be programmed specifically to the problem. Also, that's a stochastic algorithm that works well on complicated problems such as path planning, which highly depends on the shape and location of the obstacles in the environment. Apart from these, it supports multi-objective optimization, which is also used in this work.

3.6 Genetic Algorithm on Path Planning Problems

In a genetic algorithm, population, chromosome and gene should be made compatible with the problem. Simply, each chromosome represents one path so that the population has many paths in path planning problems. Every single coordinate variable is a gene in the algorithm. Since each waypoint has three coordinate variables, the number of genes is equal to three times the number of waypoints. By optimizing all of the coordinate variables of waypoints, the best path is obtained.

Feasible initial population creation and crossover, mutation, reduction, link, repair operators are the genetic algorithm operators. They all will be explained in detail in the following sections.

3.6.1 Feasible Initial Population Creation

There has to be an initial population in a genetic algorithm to create better populations by evolving it. Generally, the initial population is created arbitrarily, but most of the paths would have unrealistic movements such as extreme turning angle, maneuverability, etc. It is wise to create feasible paths at the beginning of the algorithm because it decreases the overall GA convergence time. It does not spend time making the paths feasible by crossover, mutation, and other operators. Therefore, feasible initial population creation is aimed at this work since it is reasonable in solving path planning problems by genetic algorithms.

A predefined number of feasible paths are generated to create the initial population of the genetic algorithm. Starting with the initial point, the following nodes are added to the trajectory until it reaches the target node without collision. Possible following nodes are generated as candidates by considering the angle limits, and one of them is picked among them. The angle limits are determined as follows:

- It should go towards the target node so that it does not deviate the direction to the target more than the maximum deviation angle $\theta_{maxdeviation}$.
- It should not make the turn very sharply that dynamics of the air vehicle cannot provide. Therefore, the turning angle should be smaller than the maximum turning angle $\theta_{maxturn}$.

The limits θ_{lower} and θ_{upper} are calculated in Equation 3.11.

$$\begin{aligned} \theta_{target} &= \text{atan}\,(target - current) \\ \theta_{current} &= \text{atan}\,(current - previous) \\ \theta_{lower} &= \max(\theta_{current} - \theta_{maxturn},\ \theta_{target} - \theta_{maxdeviation}) \\ \theta_{upper} &= \min(\theta_{current} + \theta_{maxturn},\ \theta_{target} + \theta_{maxdeviation}) \end{aligned} \tag{3.11}$$

Candidate path angles are generated by dividing the interval between lower and upper limits equally by the step angle θ_{step}. The position of the candidate point is calculated by shifting the current position in the direction of the randomly selected candidate angle. The shifting length might be randomly changing or adaptive, but it is taken constant in this work. All these processes are shown in Figure 3.7.

$$candidate = current + length * \begin{bmatrix} \cos\theta_{candidate} \\ \sin\theta_{candidate} \end{bmatrix} \tag{3.12}$$

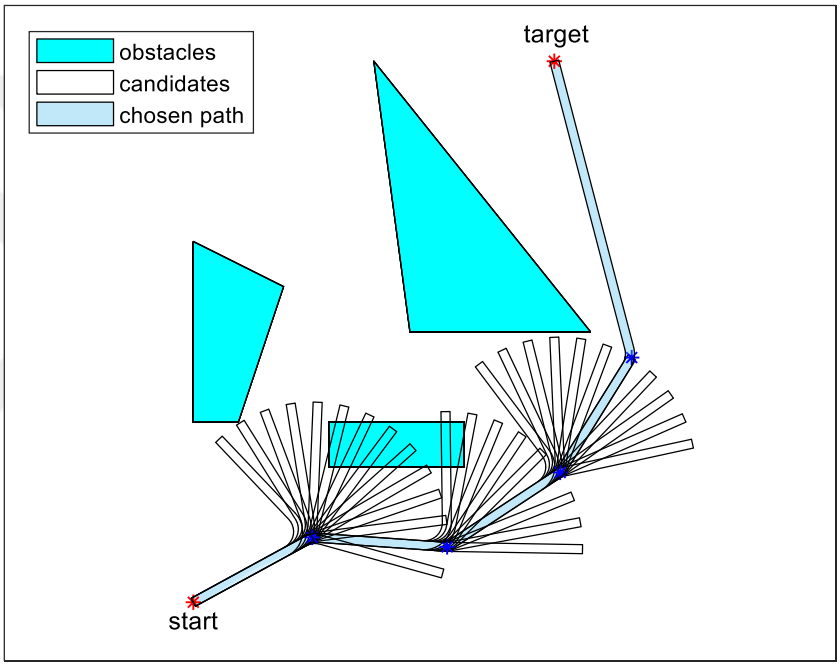

Figure 3.7. Proceeding over Randomly Chosen Candidate Paths

If a candidate polygon coincides with an obstacle, the climbing and diving points are calculated, which is explained in Section 3.4.2, since it needs to go over the obstacle in the 3D path planning problem. The climbing, diving points, and the resulting path is an example scenario that can be seen in Figure 3.8.

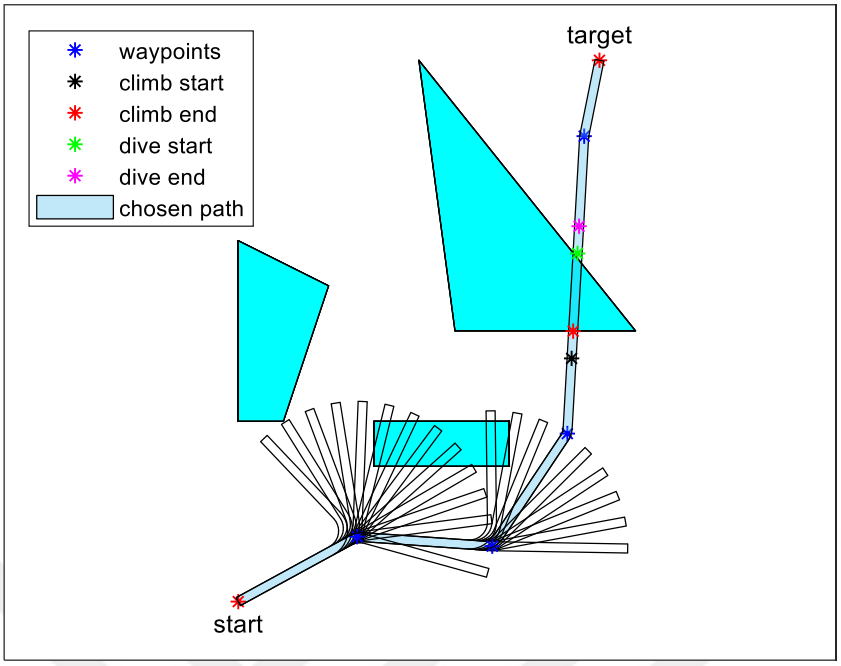

Figure 3.8. Going over a Candidate Path that Coincides with an Obstacle

Pseudocode 3.1. Feasible Initial Population Creation

```
add initial waypoint to path
while true
    if the last waypoint of the path directly connects to target without collision
        check constraint 3, start over in case of any violations
        add target to path
        break
    else
        select a candidate angle randomly and calculate the candidate position
        check constraint 2, start over in case of any violations
        if the error polygon of candidate intersects with an obstacle
            if end of the error polygon is already inside the obstacle
                extend the polygon until it gets outside of the obstacle
            check constraints 5 and 6, start over in case of any violations
            place candidate according to constraint 7 after obstacle
            add currently located candidate to path
        else
            add candidate to path
```

Note that there is no need to check angle constraints during creating the initial population because the candidate angle limits are calculated according to these constraints.

3.6.2 Main Operators of Genetic Algorithm

3.6.2.1 Crossover

In a genetic algorithm, crossover is a genetic operator used to combine the genetic information of two parents to generate a new individual, which is called offspring. Operator stages are given below.

3.6.2.1.1 Selection

Two individuals from the population are selected as parents in order to crossover the parent genes. The selection process is significant for converging to the global optimum point. It should be done so that the better paths have a higher chance to be picked by calculating the cumulative probability vector of the population. In order to calculate the cumulative probability vector, a fitness vector is obtained by figuring the cost of each individual.

$$fitness(j) = \frac{1}{1 + getCost(population(j))}$$
$$cumulative(j) = cumulative(j-1) + \frac{fitness(j)}{sum(fitness)} \quad (3.13)$$
$$\text{where j } = 1,2,3,\ldots,\text{population size}$$

The selection algorithm starts with generating a random number between 0 and 1. The individual with the maximum cumulative probability smaller than this random number is found using binary search and selected as the first parent. Similarly, the second parent is selected by following the same procedure, but if the first and the second parents are identical, the second parent should be reselected. Besides, if a parent consists of only three waypoints, it already has the minimum number of waypoints, so the parent cannot be crossed. In this case, the parent is added to the population and the probability of having a similar solution to the parent increases. This parent will also have promising potential in the mutation stage.

3.6.2.1.2 Crossover Methods

After selecting the parents, chromosomes are disintegrated, and the pieces are swapped between the parents, which is illustrated in Figure 3.9. There is only one crossover point on the left side so that chromosomes are divided from one point. On the right side, there are two crossover points so that chromosomes are divided from multiple points. The number of fragmentations determines the method.

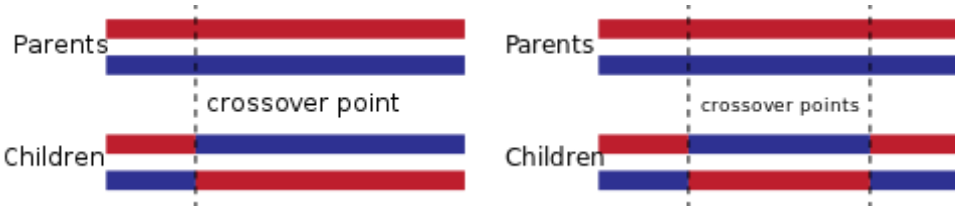

Figure 3.9. Single Point Crossover and Multi-Point Crossover [27]

Notice that the lengths of the chromosomes in Figure 3.9 are equal to each other. Whereas the lengths of the chromosomes are generally constant in a genetic algorithm, it is varying in path planning problems since the number of waypoints is not constant. Therefore, different crossover points must be chosen for each path.

3.6.2.1.2.1 Single-point crossover

In the single-point crossover, one point fragmentation of the parents is used. In other words, two incomplete path pieces are connected, and a new path is created. The first path piece is the piece of the first parent, which is between the initial waypoint and the crossover point of the first parent. Similarly, the second path piece is the piece of the second parent, which is between the crossover point of the second parent and the target point. Both crossover points are randomly selected, and every waypoint cannot be chosen as a crossover point. Logically, initial and target waypoints cannot be crossover points. Next to the last waypoint of the first parent cannot be a crossover point because it would not connect to the second path piece when it can connect directly to the target. Similarly, the second waypoint of the second parent cannot be a crossover point. After crossover points of the parents are randomly selected among their appropriate waypoint candidates, the second piece is simply attached to the end of the first piece. The new connection of the resulting individual is checked according to the related angle and distance constraints. If there are no violations, currently obtained feasible offspring is put into the population. The single point crossover algorithm is given below.

Pseudocode 3.2. Crossover

```
calculate fitness and cumulative probability vectors of the population
for i = 1 to population size
    select parents in regard to cumulative probability vector of the population
    if any parent consists of only three waypoints
        add the parent with three waypoints into the population
        continue
    select crossover points of the parents
    offspring = parent1(1:first crssver point)&parent2(second crssver point:end)
    check necessary ones of constraints 1 to 5; repair in case of any violations
    if there is an obstacle between crossover points
        check constraints 5,6 and 7; repair in case of any violations
        calculate the climbing and diving pnts, store in the offspring structure
    add offspring to crossed population
```

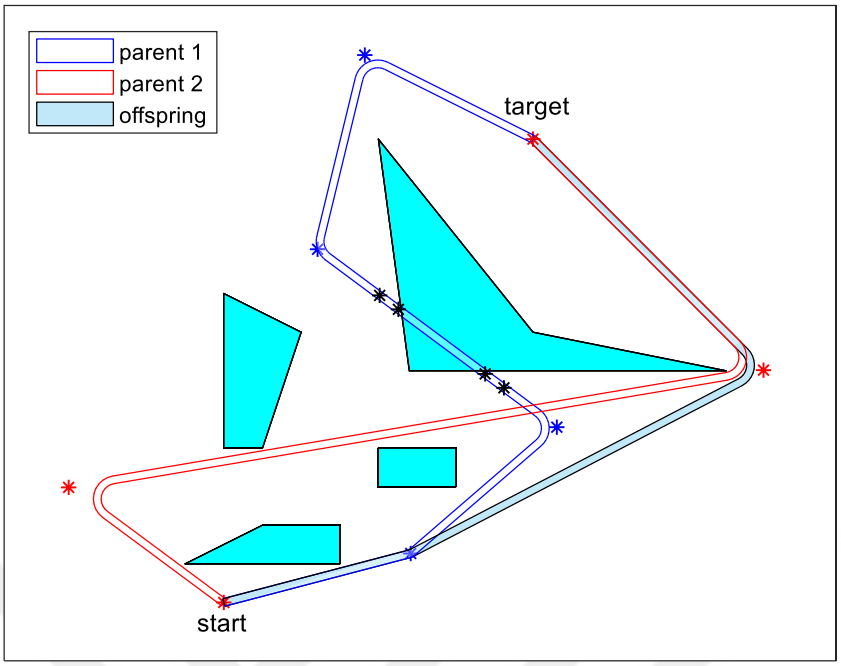

Figure 3.10. Single-Point Crossover

Figure 3.10 illustrates the single-point crossover operation. When the crossover point candidates of parent 1 are waypoints 2,3 and 4, the first crossover point is selected as the second waypoint. Similarly, since the only crossover point candidate of parent 2 is waypoint 3, the second crossover point is selected as the third waypoint. The offspring is obtained by connecting two pieces and shown with the blue shaded path in the figure. This is a very good example of the effectiveness of crossover in genetic algorithm. It discovers a new path that is better than its parents.

3.6.2.1.2.2 Multi-point crossover

Multi-point crossover is a generalization of the single-point crossover, which uses multiple fragmentations. Similarly, two crossover points for each individual are selected randomly, and the pieces are swapped between each other.

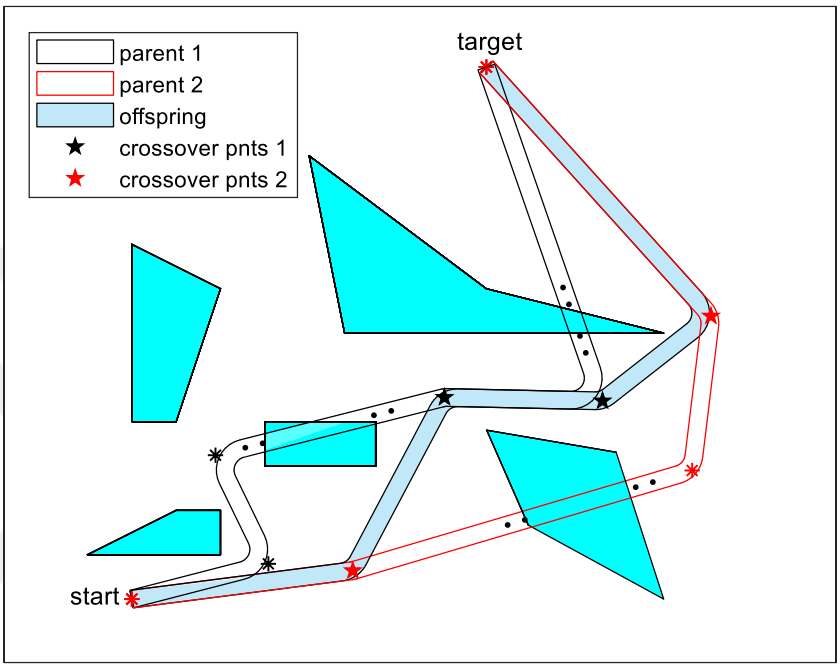

Figure 3.11. Two-Point Crossover

In Figure 3.11, the crossover points of the parents are indicated by stars. The inner part of crossover points 1 and the outer parts of crossover points 2 are connected. Whereas the parents have some climbings, the offspring has no climbings, and it's better than its parents with respect to all objectives.

3.6.2.2 Mutation

In a real-coded genetic algorithm, simple static Gaussian mutation is used generally. The i-th parameter z_i of an individual is mutated by $z_i = z_i + N(0, \sigma_i)$ with mutation rate p, where $N(0, \sigma_i)$ is an independent random Gaussian number with a mean of

zero and the standard deviation σ_i [28]. The probability density function of Gaussian distribution has a graph whose value is decreasing as it gets away from zero. It means mutated individuals will be close to the original ones. A similar approach is used in this work. Instead of updating position coordinates separately, the point is relocated randomly in a mutation area around the point. Mutation operator stages are given below.

3.6.2.2.1 Deciding if the mutation takes place

A random number is generated for each individual in the population to decide if they mutate. If the random number is smaller than the mutation rate, the individual mutates.

3.6.2.2.2 Waypoint selection

In mutation, one of the waypoints of a path is randomly selected in order to change its location. Simply, initial and final waypoints cannot be selected for mutation because their locations are unalterable.

3.6.2.2.3 Mutation takes place

The selected waypoint is relocated randomly inside a circle centered at the selected waypoint, illustrated in Figure 3.12. The point is simply shifted by the distance d_R in the direction of the angle θ_R.

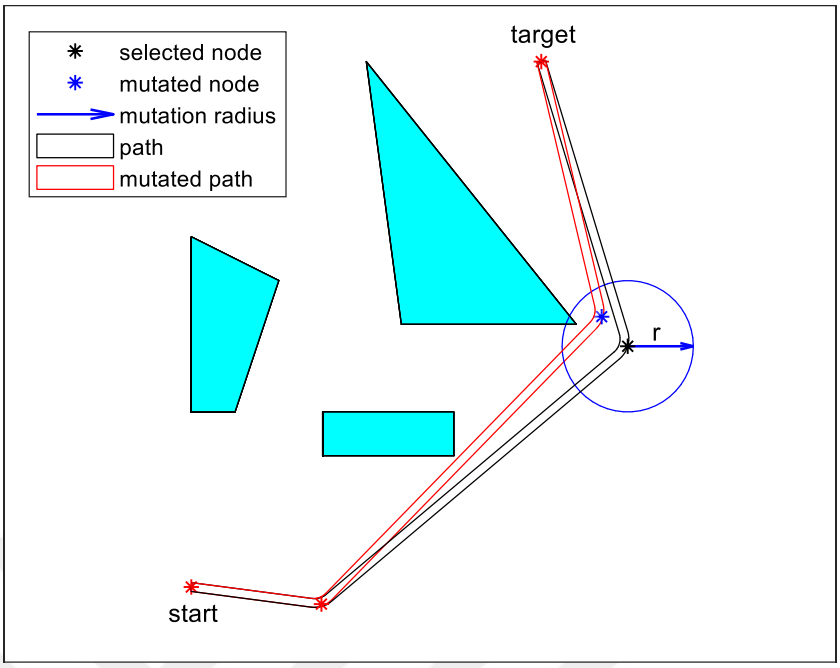

Figure 3.12. Mutation

In order to calculate the new location of the selected waypoint;

- the distance d_R is randomized between 0 and mutation radius, which defines the distance between the original waypoint and the mutated waypoint.
- the angle θ_R is randomized between 0 and 2π, which defines the direction that the point is shifted.

The calculation of the new position is given in Equation 3.14.

$$\begin{bmatrix} x_m \\ y_m \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + d_R * \begin{bmatrix} \cos\theta_R \\ \sin\theta_R \end{bmatrix} \tag{3.14}$$

The new waypoint is located somewhere in the circle, and the new path is created. If the shifted point lies within an obstacle or the related angle and distance constraints fail, the resulting path must be repaired. Else, the mutated path is added to the population. The mutation algorithm is given in Pseudocode 3.

Pseudocode 3.3. Mutation

```
for i = 1 to population size
    generate a random number between 0 and 1
    if random number > mutation rate
        continue
    path = population(i)
    select a random waypoint from the path except initial and target points
    generate a random distance between 0 and a predefined constant
    generate a random angle between 0 and 2π
    update the position of the selected waypoint
    get the new path and update the turning and climbing informations
    check necessary ones of constraints 1 to 5; repair in case of any violations
    add mutated path to mutated population
```

The mutation is very beneficial to find the global optimum solution. Mutated chromosomes are added to the population even if they are worse than their original states because it increases the diversity of the population. Diversity is a necessity in any genetic algorithm in order to avoid getting stuck in local minima. Also, mutation enhances the possibility of convergence to the global optimum solution significantly.

3.6.3 Problem Dependent Operators

Genetic algorithm is a flexible algorithm that allows defining problem dependent operators. They can be seen as special mutations, and they make the algorithm capable of converging faster. Three operators specific to the problem are defined in this work.

3.6.3.1 Reduction Operator

A path is defined using a set of waypoints in order to have a fast and neat algorithm in this work. However, suppose there is a redundant waypoint between two

waypoints that can be directly connected to each other without colliding with any obstacles. In that case, the redundant waypoint should be removed from the path. Because it increases the length of the path needlessly, and it brings unnecessary turnings to the path, which increases the energy consumption. Also, the most significant advantage of this operator is that it speeds up the convergence process a lot. For example, mutation of a redundant waypoint does not contribute to optimization, so reduction operator avoids wasting time with the redundant waypoints.

In reduction operator, all of the two waypoint combinations with one or more waypoints between them are checked if they can be connected directly to each other without colliding with an obstacle and all the related angle and distance constraints are satisfied. If reduction is possible, two waypoints are connected. Reduction operator is needed after creating the feasible path, crossover operator and mutation operator because redundant points might show up in all these stages. All the combinations should be checked after creating the feasible path, and the algorithm is given in Pseudocode 4.

Pseudocode 3.4. Reduction Operator

```
len = length of the path
for i = len:-1:3
    for j = 1 to i-2
        if waypoint j directly connects to waypoint i
            check related constraints, continue in case of any violations
            return the shortened path
```

Create feasible path algorithm creates new paths by proceeding towards random directions until arriving at the target point. Therefore, the resulting paths always have zigzag shapes, and such paths generally contain unnecessary movements, which is illustrated in Figure 3.13. Reduction operator removes redundant waypoints, and a better path is obtained.

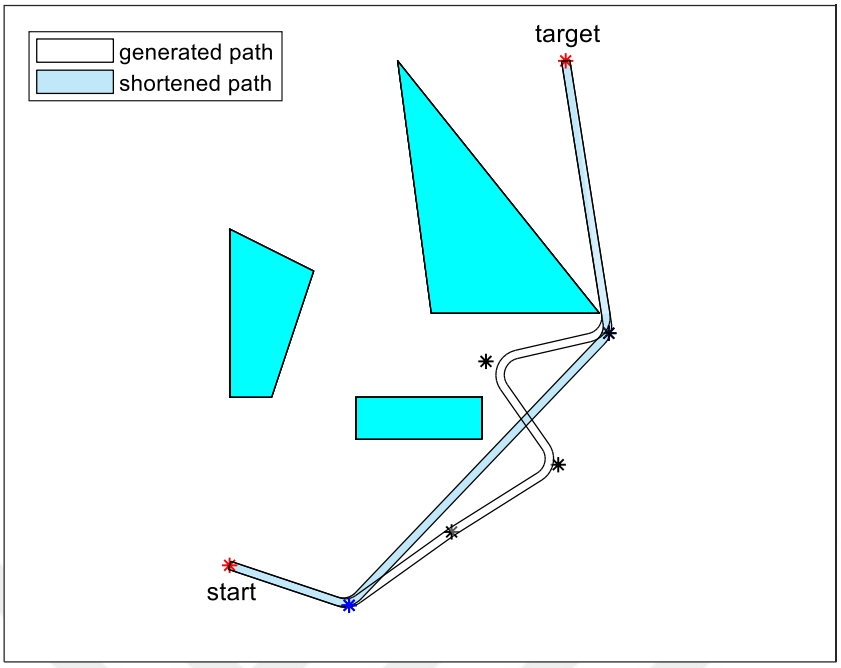

Figure 3.13. Reduction Operator After Creating Feasible Path

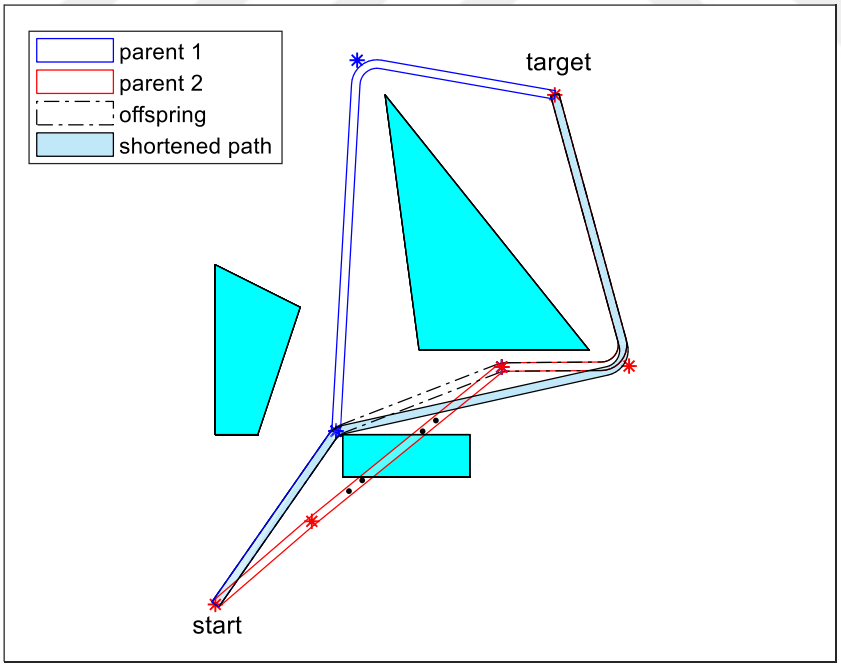

Figure 3.14. Reduction Operator After Crossover

In reduction operator after crossover, only the waypoints of the first parent are tried to connect to the waypoints of the second parent since the parents are already shortened separately. In Figure 3.14, offspring is obtained after crossing parent 1 and parent 2. Reduction operator gets involved after crossover, and the unnecessary waypoint is removed from the path.

In reduction operator after mutation, only the mutated waypoint is tried to connect to the other waypoints because the only change in the path is the mutated waypoint. Figure 3.15 shows that a path mutates, and the mutated path becomes suitable to be shortened. The mutated waypoint directly connects to the starting point by skipping the middle waypoint, and a much better shortened path is obtained.

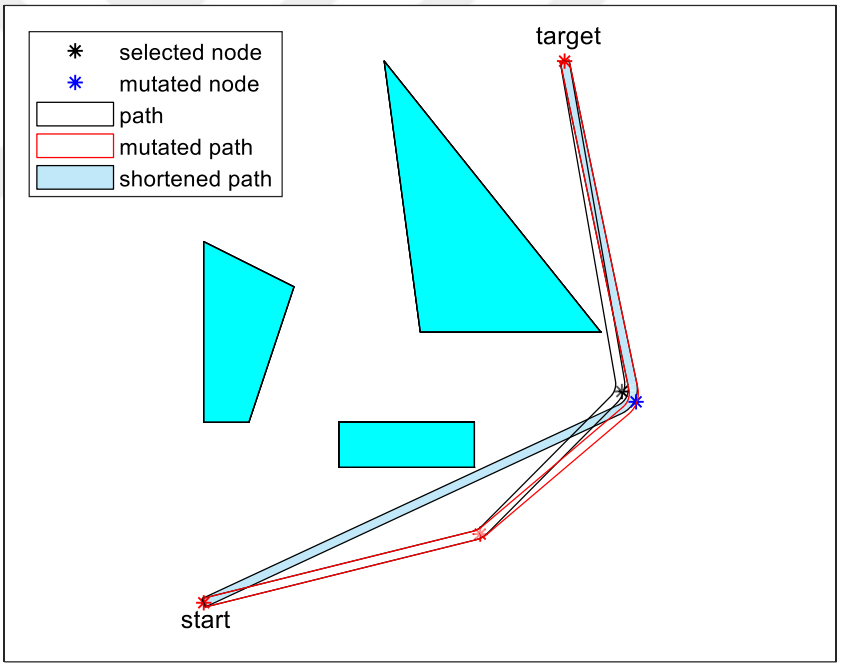

Figure 3.15. Reduction Operator After Mutation

3.6.3.2 Link Operator

A randomly selected waypoint is directly linked to another randomly selected waypoint that is not adjacent to the first one in link operator. By this way, better paths might be obtained according to the problem's objectives by exploring new paths in the environment. This operator boosts local optimization and fastens the convergence process. Also, link operator rate should be chosen carefully because it might reduce the diversity of the population.

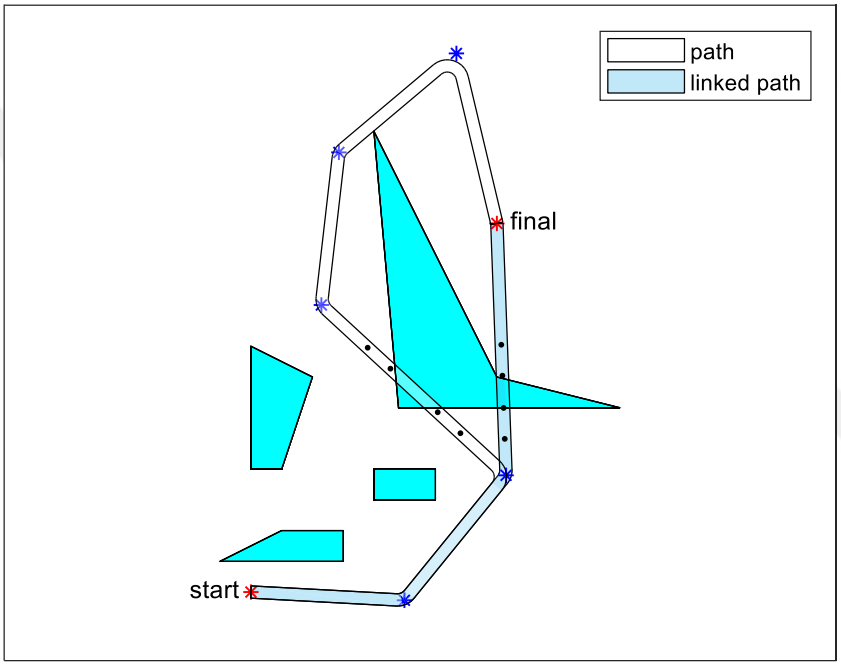

Figure 3.16. Link Operator

In Figure 3.16, the target waypoint of the path is linked to the third waypoint, so the resulting path is much shorter than the original path. The operator discovers a new route that goes over another region of the same obstacle. Also, the linked path seems the local optimum solution for this path, as promised above.

3.6.3.3 Repair Operator

The aim is to find the best solutions by exploring the new ones in genetic algorithm. The new individuals might not satisfy the constraints of the problem, and throwing such individuals out of the population is the general attitude. However, if the violation is removed by fixing the problem, the individual is added to the population instead of killing it, so it simply helps diversity and convergence time. Repair operator is defined for such cases in order not to waste diverse solutions, which is applied after creating feasible path, crossover, mutation and link operators.

In Figure 3.17a, link operator is applied to the slightly different path from the path in Figure 3.16, and the resulting path does not satisfy the minimum distance constraint before climbing. In physical meaning, the faulty waypoint, which is marked in Figure 3.17b, is very close to the obstacle to start climbing, so locating it far away from the obstacle might be a wise strategy. Therefore, repair operator shifts the faulty waypoint in the climbing direction and obtains a new waypoint.

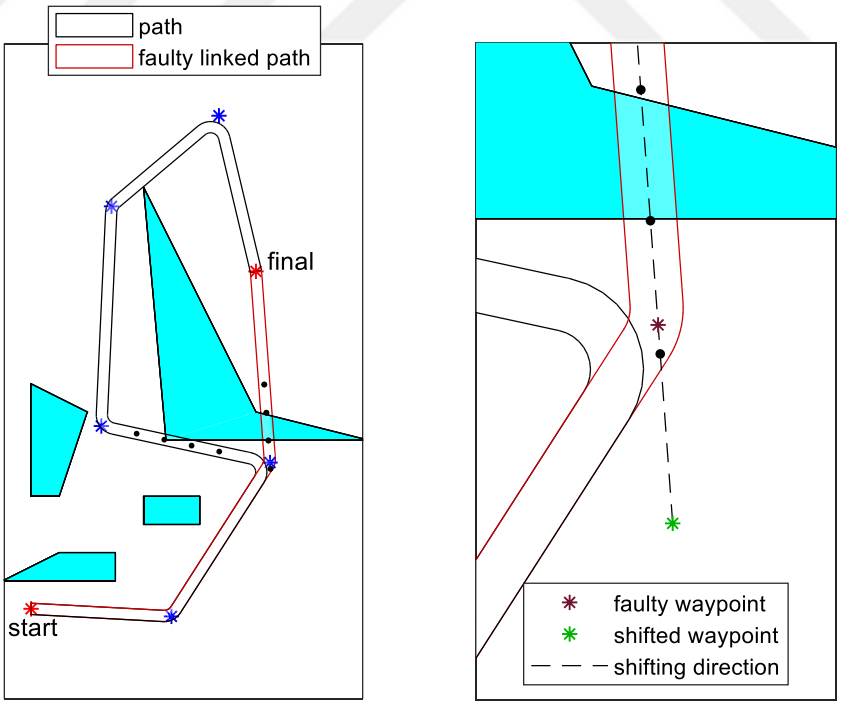

Figure 3.17. Repairing Faulty Linked Path by Repair Operator

The repaired path with the new waypoint is shown in Figure 3.18.

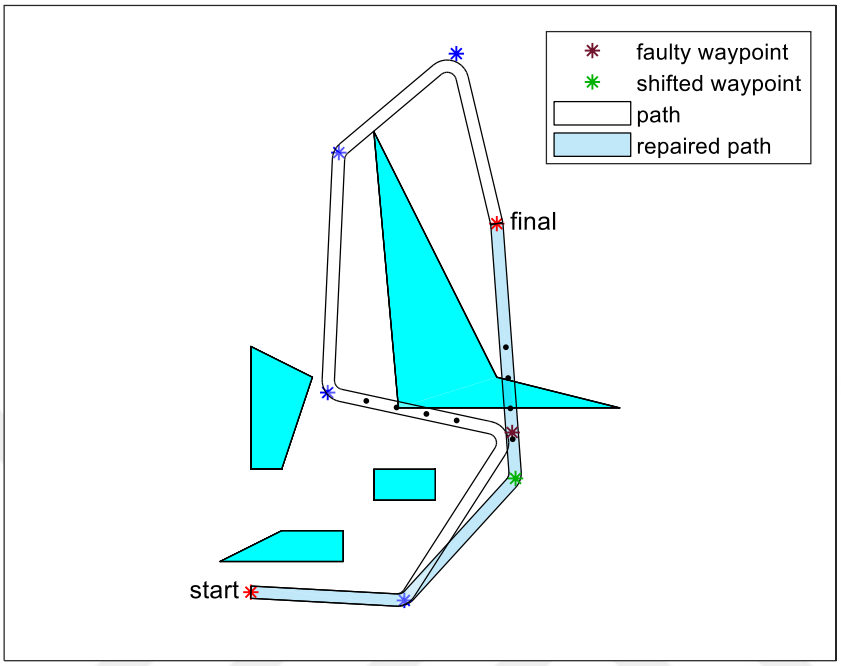

Figure 3.18. Repair Operator

3.6.4 Overall Genetic Algorithm for 3D Path Planning

After generating a feasible initial population, all the operators are applied to the population at every generation until the maximum number of generations is reached. The structure of the main algorithm is given in the flowchart below.

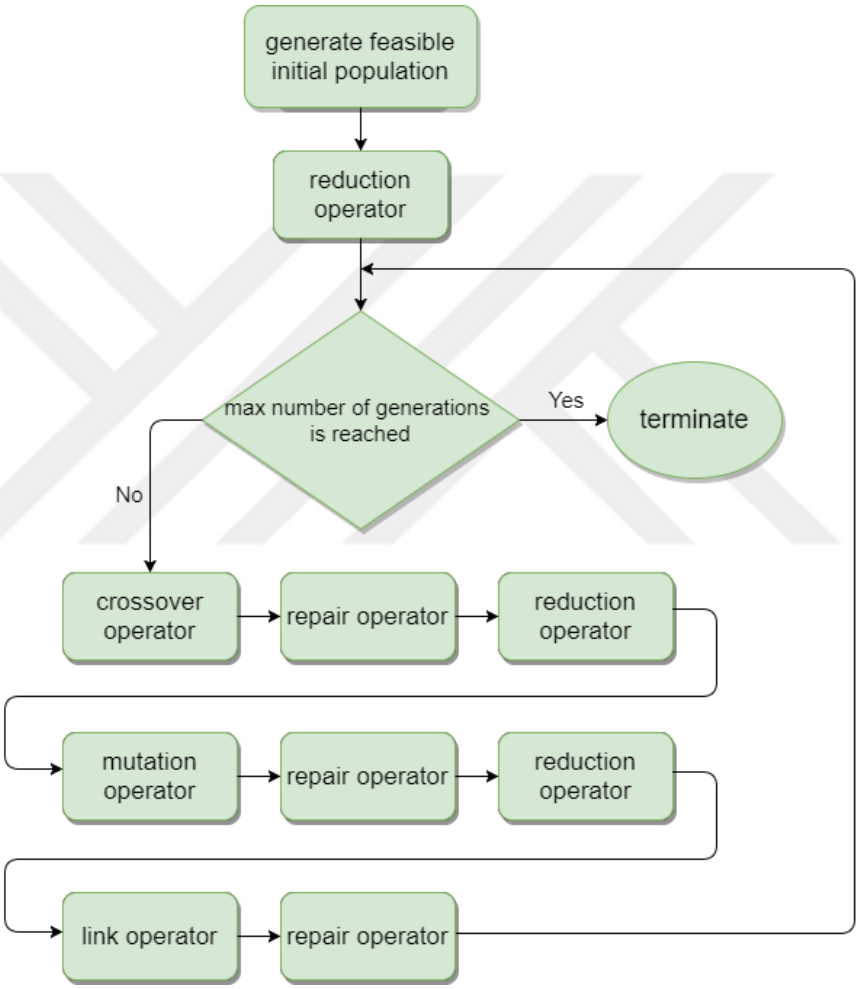

Figure 3.19. The Main Algorithm

3.6.5 Simulations

In these simulations, the aim is to see if the results seem intuitively correct and observe how different parts of the cost function affect the result. The starting point, the target point and the obstacles are inputs of the algorithm. The constraint values given in Appendix B are the configuration parameters. When they are all set, the best path would be found by the algorithm. Figures from 3.20 to 3.25 show the effect of the obstacle locations and dimensions. Simulations are conducted when the number of generations is 50. The cost function is a weighted sum of three objectives, which are length, fuel consumption and risk of being detected. Thus, these are single objective optimization solutions. Note that the paths found in the scenarios below might change if the weights of the objectives are changed. Multi-objective optimization methods get rid of adjusting weights, which is explained in the next section. The simulations last 20 seconds on average on a 2.80 GHz CPU and 8GB RAM system. All the functions are implemented on MATLAB 2017b.

3.6.5.1 Experiment I

This experiment is conducted to see if the paths found look like global optimum solutions intuitively. Scenario I shows that the path found by the algorithm looks like the best path according to the three objectives. Scenario II shows that the map is slightly changed in a way that the top corner of the uppermost obstacle is shifted upwards so that it blocks the way it passed by. This time, the algorithm finds a different path that looks like the best path on the new map. Logically, the cost has increased since the path gets longer. This simulation verifies that the algorithm works well intuitively.

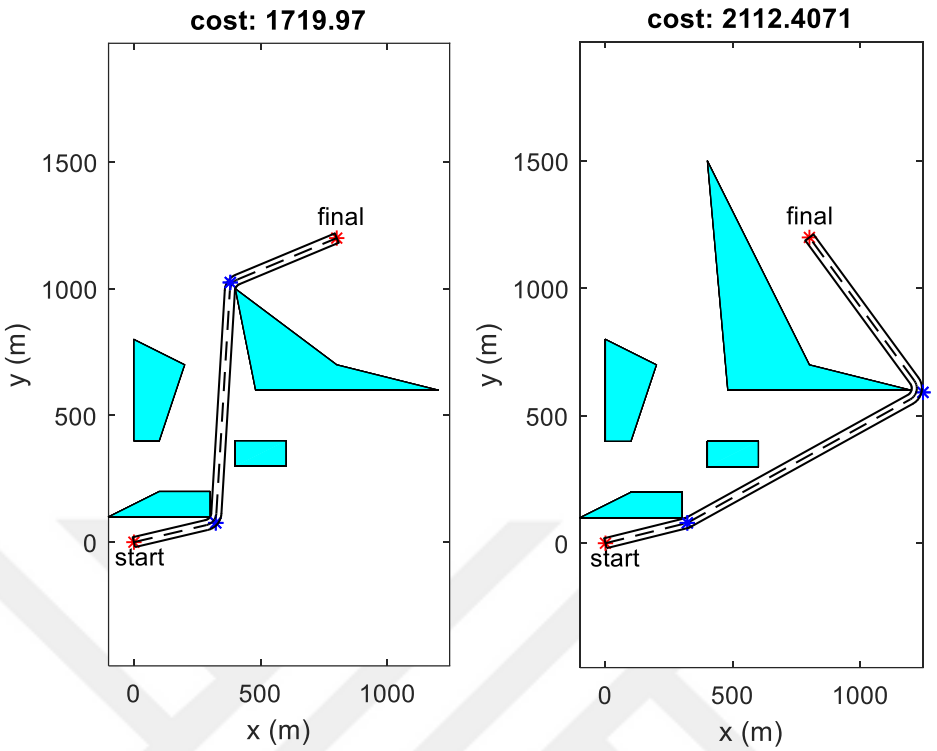

Figure 3.20. Scenario I and Scenario II

3.6.5.2 Experiment II

In Figures 3.21 and 3.22, simulations are conducted to see the effect of the heights of obstacles. Scenario III shows that the path found goes over the uppermost obstacle and reaches the target point. When the height of the obstacle is triplicated, the path shown in Figure 3.22 is obtained. As expected, the algorithm finds a less costly path that gets around the obstacle instead of climbing over the very high obstacle. Since it covers a longer distance, the second path is much costly, but it is still the best path on the map in Scenario IV.

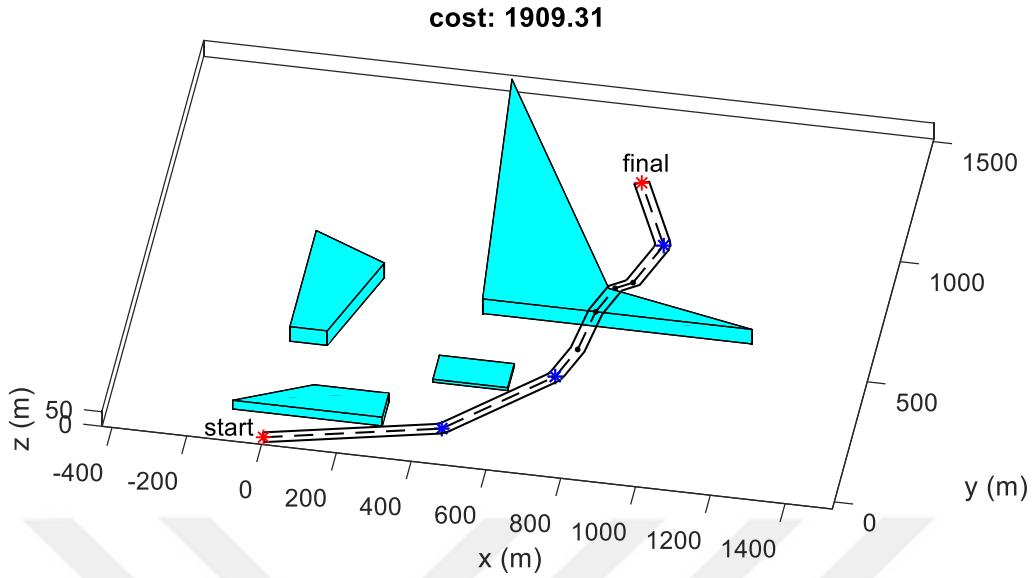

Figure 3.21. Scenario III

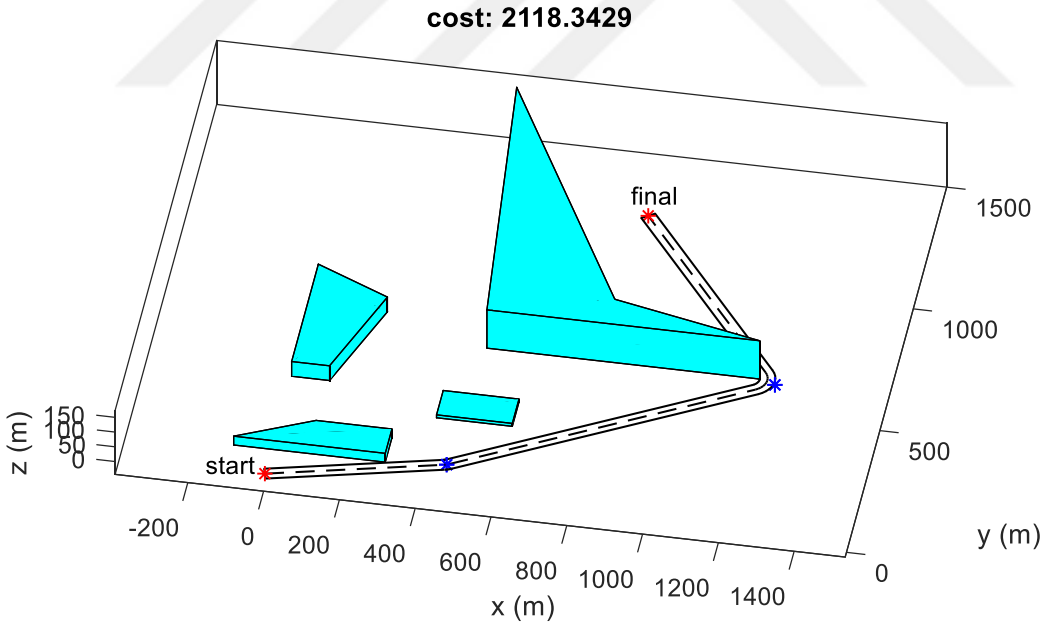

Figure 3.22. Scenario IV

3.6.5.3 Experiment III

In this experiment, the aim is to see if the algorithm works well in complicated environments. Figure 3.23 shows a path founded in a complicated environment. Notice that starting point and target point are very close to the obstacles, which increases the probability of having infeasible paths. However, thanks to the problem dependent operators defined in this work, the algorithm handled the infeasible paths and found the best path successfully.

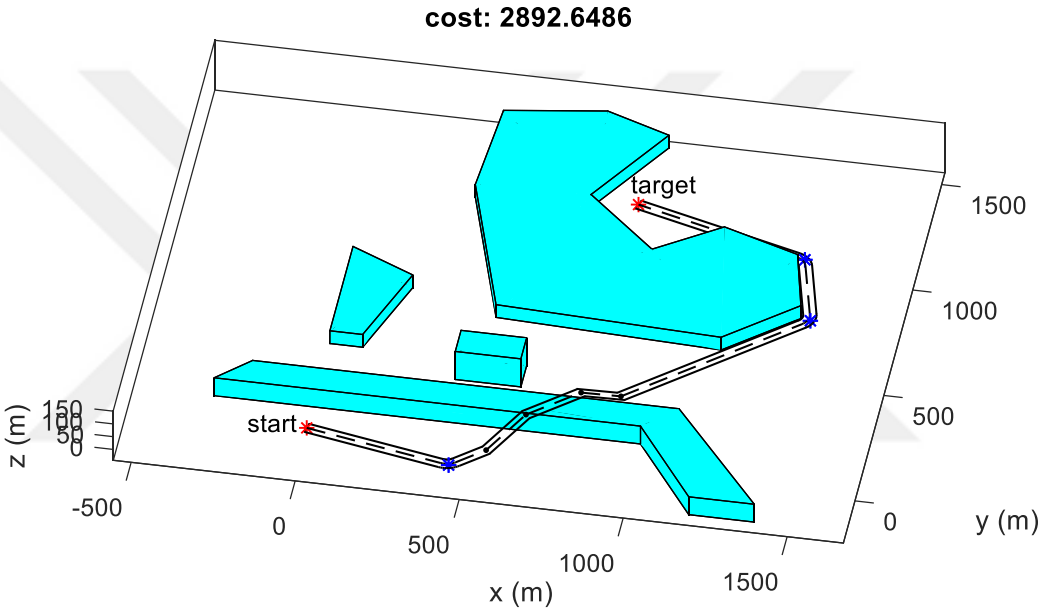

Figure 3.23. Scenario V

If there is a narrow canal in the middle of the uppermost obstacle, the algorithm would be tested in a more challenging case illustrated in Figures 3.24 and 3.25. For such a case, finding the best path for once is not enough to verify the algorithm. The success rate is important. Therefore, a Monte Carlo simulation is performed in order to eliminate the randomness of the algorithm. Monte Carlo simulation results are shown in Table 3.1. According to this, the algorithm finds the best path, which goes from the canals mostly. In failure cases, it gets around the obstacle as if there is no

canal, i.e., it finds the path shown in Figure 3.23. Also, the success rate increases as the number of generations increases, which can be seen from the table below.

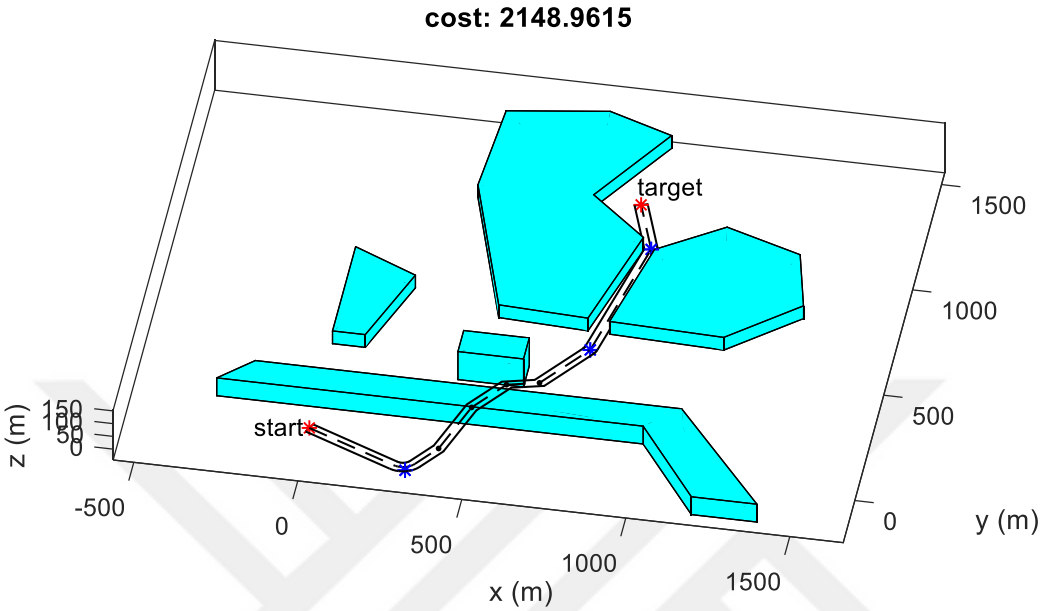

Figure 3.24. Scenario VI

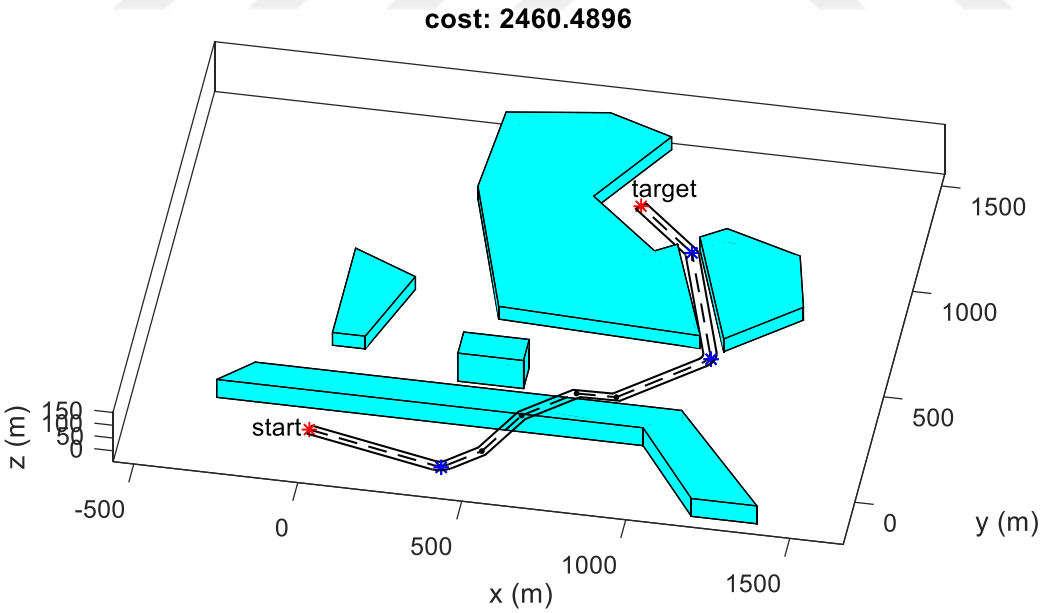

Figure 3.25. Scenario VII

Table 3.1. Monte Carlo Simulation Results

Scenario	number of generations	canal is found	canal is not found	number of trials
VI	50	17	3	20
VII	50	15	5	20
VI	100	20	0	20
VII	100	18	2	20

3.6.6 Multi-Objective Optimization in Path Planning Problem

In order to decide if one path is better than another path, their cost values are calculated according to an objective function. This objective can be the length of the path, fuel consumption, total maneuver angle or visibility for the path planning problem. One objective or weighted sum of multiple objectives can be taken as one function and better solutions are determined by comparing their objective function values in single-objective optimization problems. The weighted sum of multiple objectives is taken as a single objective with the predefined weight vector in Equation 3.15. The weight vector is chosen in proportion to the importance of the objectives, and this is a straightforward method of combining different objectives.

$$F(x) = \sum_{k=1}^{K} w_k * f_k(x) \tag{3.15}$$

For example, the length of the path and fuel consumption functions are added to the total objective function with different weights in Equation 3.16. Apparently, having a shorter path is more important than consuming less fuel according to the weights in the total objective function below.

$$f_{total}(x) = 0.75 * f_{length}(x) + 0.25 * f_{fuel}(x) \tag{3.16}$$

However, it is tough to set the weight vector in the weighted sum of multiple objectives method, and it does not provide a set of alternative solutions by changing the weights of objectives. In multi-objective optimization problems, two or more objective functions are minimized or maximized simultaneously. The desired output is generally a set of non-dominated solutions rather than one solution. It is a remote possibility that a single solution that optimizes all of the objectives exists. Non-dominated solutions are simply the solutions that are not dominated by any other solutions. They need to satisfy the following rules:

- They are not worse than any other solutions in all objectives.
- They are better than others in at least one objective.

Such solutions are also called Pareto-optimal solutions, and they are considered equally good, i.e., none of them are superior over other solutions.

Multi-objective genetic algorithms are advantageous over traditional methods because genetic algorithms fundamentally operate on a set of solutions when the output of multi-objective optimization is a set of solutions. There are various multi-objective genetic algorithms, and the Strength Pareto Evolutionary Algorithm [29] is used in this work because it is very efficient and powerful.

3.6.6.1 Strength Pareto Evalutionary Algorithm (SPEA)

SPEA is a mixture of multi-objective optimization techniques and evolutionary algorithms in order to find non-dominated solutions. It stores all the non-dominated solutions found so far in an external population. This external population is updated in every generation of genetic algorithm. The outline of the algorithm is illustrated in Figure 3.26.

All non-dominated individuals of the population are found, and they are added to the external population, which is called Pareto set in Figure 3.26. After newly found non-dominated solutions are compared with the existing Pareto set, all non-dominated solutions are stored in extended Pareto set, whereas dominated ones are

deleted. If the size of the resulting extended Pareto set is greater than a predefined value, the size is reduced by the clustering technique. The clustering technique reduces the size in a way that it eliminates the solutions if similar of them already exist in the Pareto set. Therefore, it preserves the characteristics of the Pareto-optimal solutions. The reduced Pareto set is united with the population and goes into the selection process. After crossover and mutation are applied to selected individuals, the next population is generated. This loop continues along with generations until the stopping condition is reached. After all, the Pareto-optimal solution set, which is the output of the algorithm, is obtained.

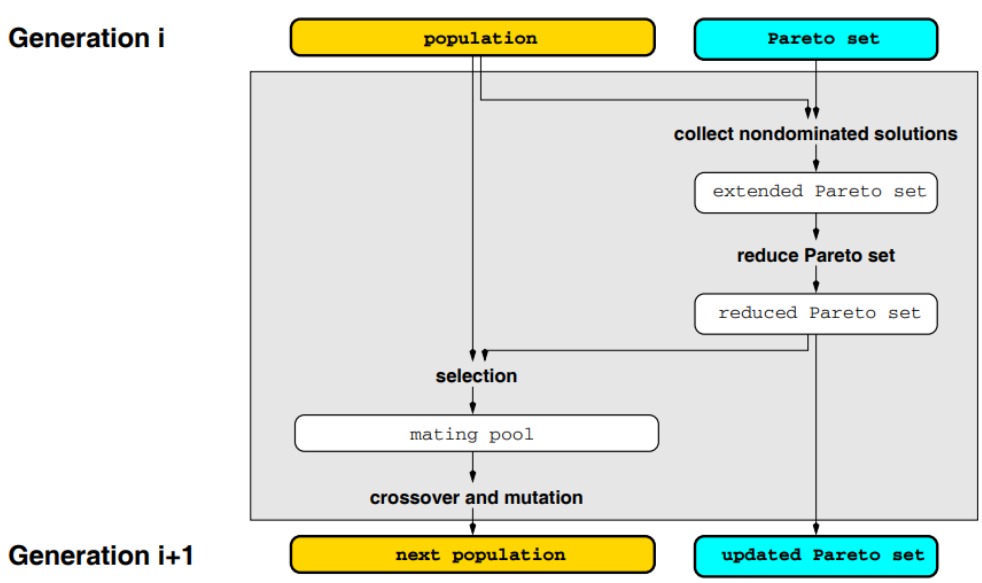

Figure 3.26. The Outline of SPEA [29]

After implementing SPEA, the length of the path and the fuel consumption of the path are chosen as objectives. The Pareto-optimal solution set is shown in Figure 3.27. The path with minimum length and the path with minimum fuel consumption are colored with blue and red, respectively. Climbing over an obstacle or making a turn consume more fuel than going straight, so the path that has less climbing or turnings is more costly according to fuel consumption objective. Thus, the red path

in the figure is the path with minimum fuel consumption because it has no climbing and fewer turnings than the others. Also, climbing over an obstacle shortens the length of the path significantly. For example, blue path saves a long distance by climbing over an obstacle.

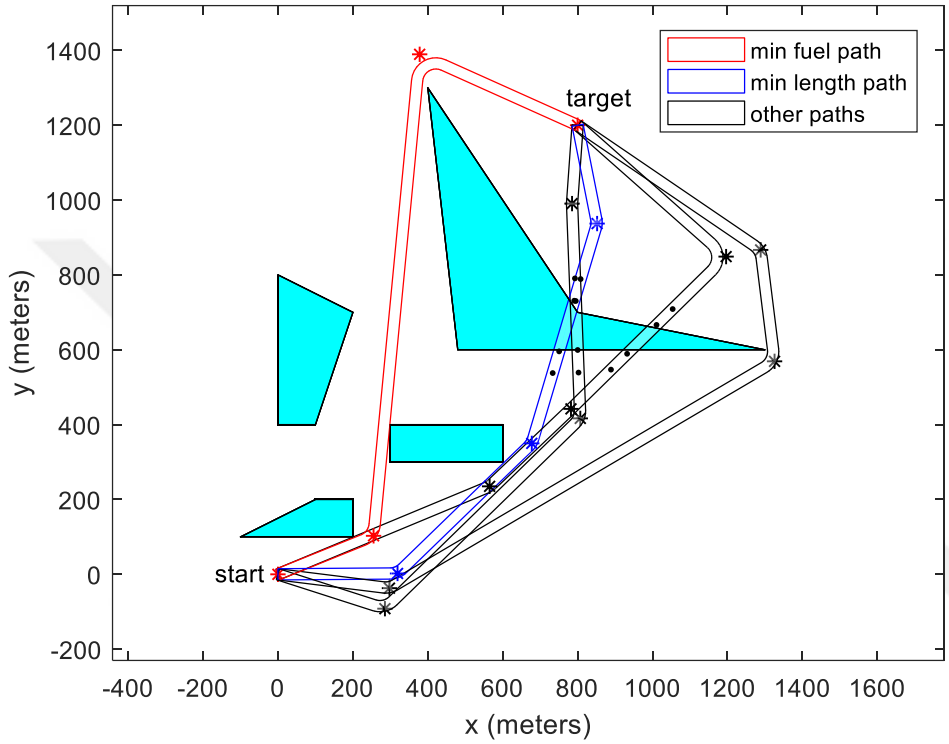

Figure 3.27. Pareto-Optimal Solutions

According to the costs of the paths, which are shown in Figure 3.28, they are all non-dominated solutions because they satisfy the non-dominance rules given above. After seeing the costs of non-dominated solutions, appropriate weights will be determined to obtain the best trade-off.

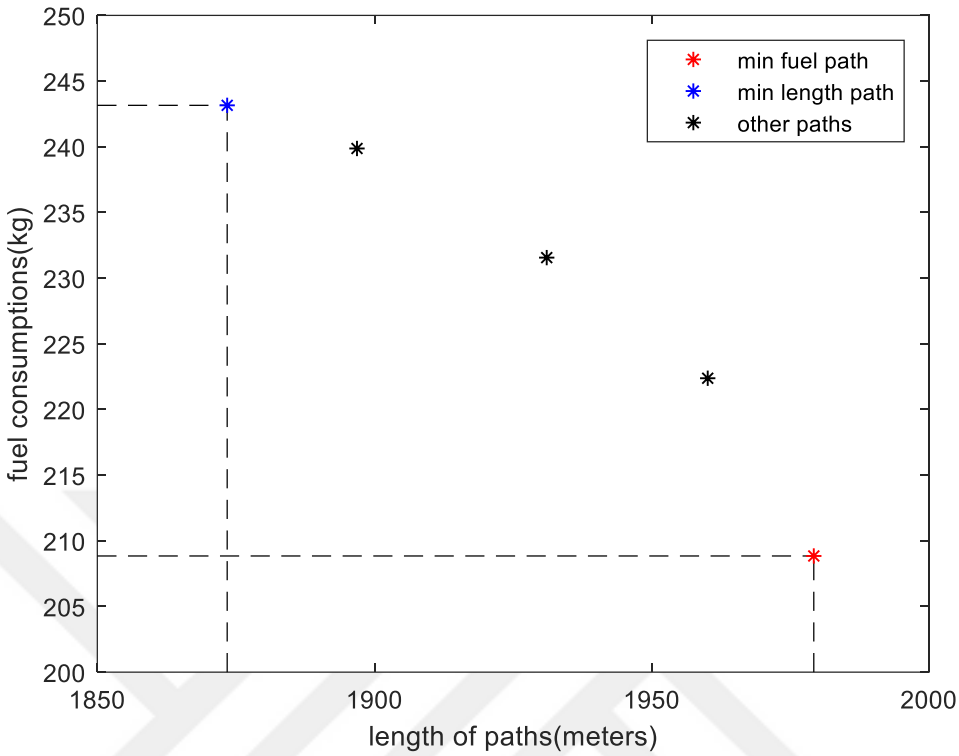

Figure 3.28. Costs of the Paths According to the Objectives

CHAPTER 4

CLOSE-IN WEAPON SYSTEMS

A gun-based CIWS is modeled and used for defending a spot against missiles in this chapter.

4.1 Modeling the Effects of CIWS Against Incoming Threats

CIWS have radars for detection of the threats, computers for calculations and rotary cannons for the assault. All these subsystems are placed on a turret, which can rotate in both azimuth and elevation axes.

Radars can detect threats that are within the detection range. As mentioned in Figure A.3, CIWS's detection range is more extensive than its assault range, so it detects the threat and tracks it without attacking until it comes within a particular range. In tracking, CIWS rotates towards the target point in both axes simultaneously so the rotation can be diagonal. As the threat enters the assault range, rotary cannon, which is an automatic firearm starts to fire at a very high cyclic rate.

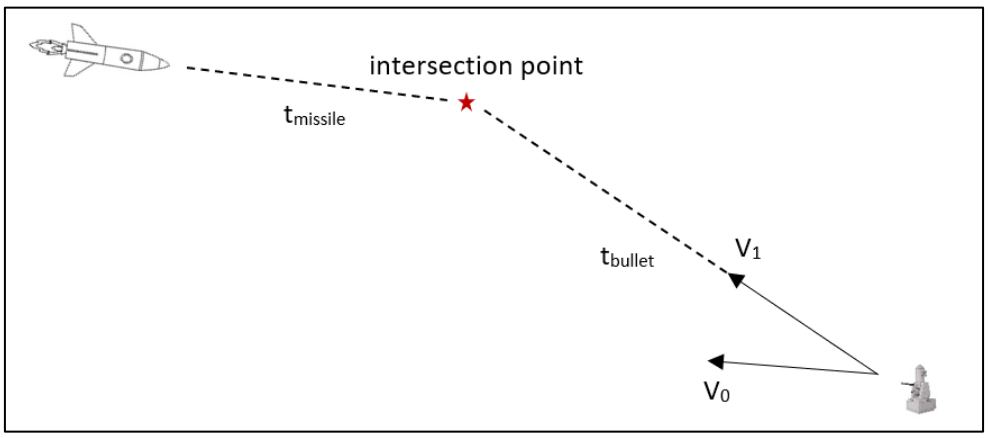

Figure 4.1. CIWS' Attacking a Threat

Time arrangement is vital in CIWS models because the threat likely moves at high speed, and its position changes quickly so that any time period matters. Figure 4.1 shows the general aspect of CIWS when attacking a threat.

The time it takes for the missile arriving the current intersection point is called $t_{missile}$. In this time duration, the CIWS needs to calculate the intersection point, rotate towards the intersection point, shoot at the intersection point, and the bullets fired need to arrive at the intersection point. Therefore, the total of the other time durations is simply $t_{missile}$, and all the calculation is based on this relation.

$$t_{missile} = t_{computation} + t_{rotation} + t_{fire} + t_{bullet} \tag{4.1}$$

The time durations given in Equation 4.1 are elaborated in the next section.

4.2 Examination of the Time Durations

4.2.1 Computation Time

It is the time taken for calculating the intersection point. The radars of the system detect the position and the velocity of the missile. Thus, the intersection point can be simply calculated if the $t_{missile}$ is known with the Equation 4.2.

$$intersection\ point = x_{missile} + v_{missile} * t_{missile} \tag{4.2}$$

4.2.2 Rotation Time

It is the time taken for the CIWS to rotate towards to intersection point. As an example, Figure 4.1 illustrates that the CIWS needs to rotate from its initial orientation V_0 to V_1 before firing. The rotation time is calculated from the change in the orientation of the CIWS. The orientation is defined by Euler angles, which can

be found easily from the CIWS direction. The desired direction of the CIWS is calculated as shown below.

$$V_1 = x_{missile} + v_{missile} * t_{missile} - x_{CIWS} = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \tag{4.3}$$

The desired pitch angle(θ) and yaw angle(ψ) are calculated in Equation 4.4. Note that roll angle(φ) is not considered because the orientation of CIWS is determined by pitch and yaw angles.

$$\begin{aligned} \theta_{desired} &= \tan^{-1} \frac{z_1}{\sqrt{x_1^2 + y_1^2}} \\ \psi_{desired} &= \tan^{-1} \frac{y_1}{x_1} \end{aligned} \tag{4.4}$$

The difference between the desired and the actual orientations is calculated as below.

$$\Delta = \begin{bmatrix} \theta_{desired} - \theta_{actual} \\ \psi_{desired} - \psi_{actual} \end{bmatrix} \tag{4.5}$$

The time required for the CIWS to rotate the arc between two orientations is simply calculated in Equation 4.6. In this work, it is assumed that the CIWS rotates at a constant angular speed w_{CIWS}.

$$t_{rotation} = \frac{|\Delta|}{w_{CIWS}} \tag{4.6}$$

4.2.3 Fire Time

Rotary cannons shoot at a very high rate of fire, and they send a bunch of bullets in a certain amount of time, which can be modeled as sheets of bullets in 3D space. These sheets have their own orientation, which is exactly the same as the CIWS'

orientation at the instant they are fired. A sheet with correct dimensions and zero pitch and yaw angles is placed at the origin and rotated around y and z axes to get the correct orientation in 3D space. The rotated sheet is obtained by multiplying the points of the sheet at the origin with the transformation matrices. After all, the rotated sheet is shifted to its current position.

$$sheet_{rotated} = sheet_{origin} * T_{sheet}$$

$$T_{sheet} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} * \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.7)$$

$$sheet_{air} = sheet_{rotated} + x_{bullet}$$

4.2.4 Arrival Time of Bullet

It is time it takes for a bullet sheet to arrive at the intersection point. As the speed of bullets is assumed to be constant, the calculation is straightforward.

$$t_{bullet} = \frac{|V_1|}{v_{bullet}} \quad (4.8)$$

As a result, all variables depend only on the intersection point position, which means if the intersection point is known, all variables can be calculated. If the Equation 4.8 is combined with the Equation 4.1, Equation 4.9 is obtained.

$$|V_1| = v_{bullet} * (t_{missile} - t_{computation} - t_{rotation} - t_{fire}) \quad (4.9)$$

If $t_{computation}$, t_{fire} are taken as constants and $t_{rotation}$ is calculated from Euler angles of vector V_1, the only unknown variable is $t_{missile}$. Thus, the equation is solved for $t_{missile}$. In this work, $t_{computation}$ and t_{fire} are taken as 0.1 seconds.

4.3 Network of the CIWS

Each subsystem of the CIWS runs on different computers and communicates with the others simultaneously, as [12] proposes. Figure 4.2 shows this network and interaction between the units. According to the figure, the radar computer gets the position and velocity information of the threat and continuously gives this information to the calculation computer. The calculation computer calculates the point that the CIWS shoot at, and it also determines if the point is close enough to fire. In all cases, the turret computer rotates the system towards the threat to track the target continuously. The fire control computer waits for the turret to complete the rotation and fire if the aim is inside the assault range and the bullets can arrive at the destination on time.

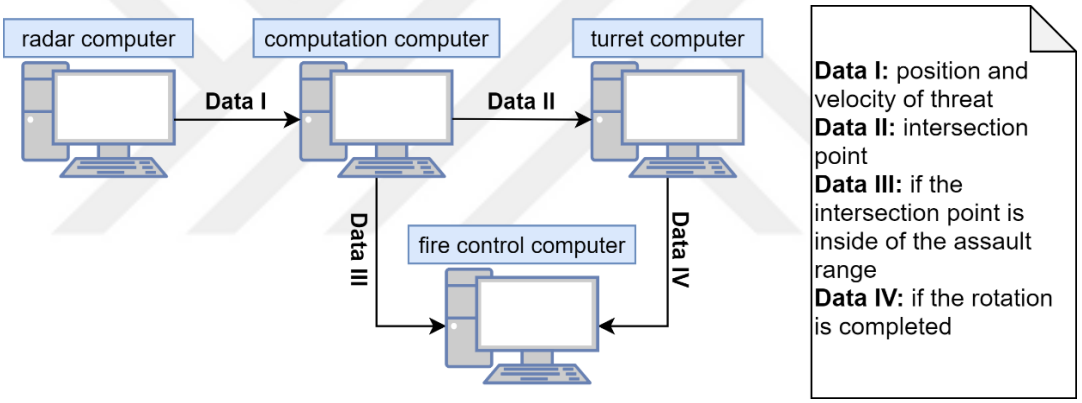

Figure 4.2. The Network of the System

4.4 Demonstration of the CIWS Model

The whole CIWS model is simulated on MATLAB Simulink, and the model is shown in Appendix A. The missile trajectory is a function of time, which is given to the system as an input. However, real radars estimate the position and the velocity of the flying objects with an error margin, and this also should be added into the model. Thus, random noise is added to the position and velocity of the threat. After

all, noisy information is sent to the system as if the threat's position and velocity are estimated by radar.

When the missile is outside of the assault range, the CIWS only tracks the threat without attacking. However, the system starts to fire just before the missile gets into the assault range because it attacks the running path of the missile. Figure 4.3 illustrates such an instant that the system starts to fire when the missile is still outside of the assault range. The sphere represents the assault region, and the line is the trajectory that the missile tracked until that instant. Many bullet sheets fired by the CIWS are on the way to come together with the missile at their own intersection points.

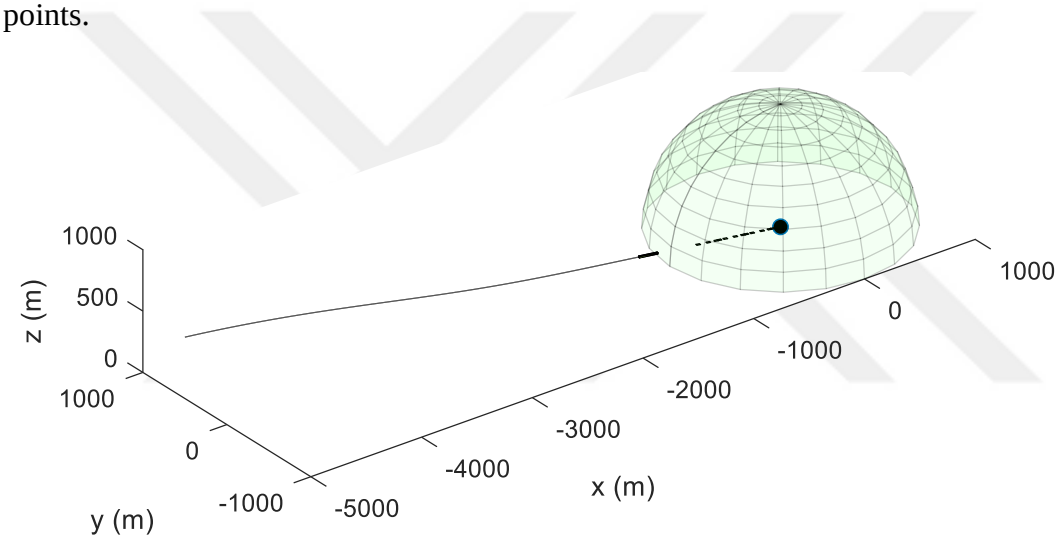

Figure 4.3. The CIWS Starts to Fire Just Before the Missile Gets into the Assault Range

The side and the top views are depicted in Figure 4.4. Note that, dimensions of the missile and the bullet sheets are exaggerated in order to visualize the situation because they are too small compared to the other distances.

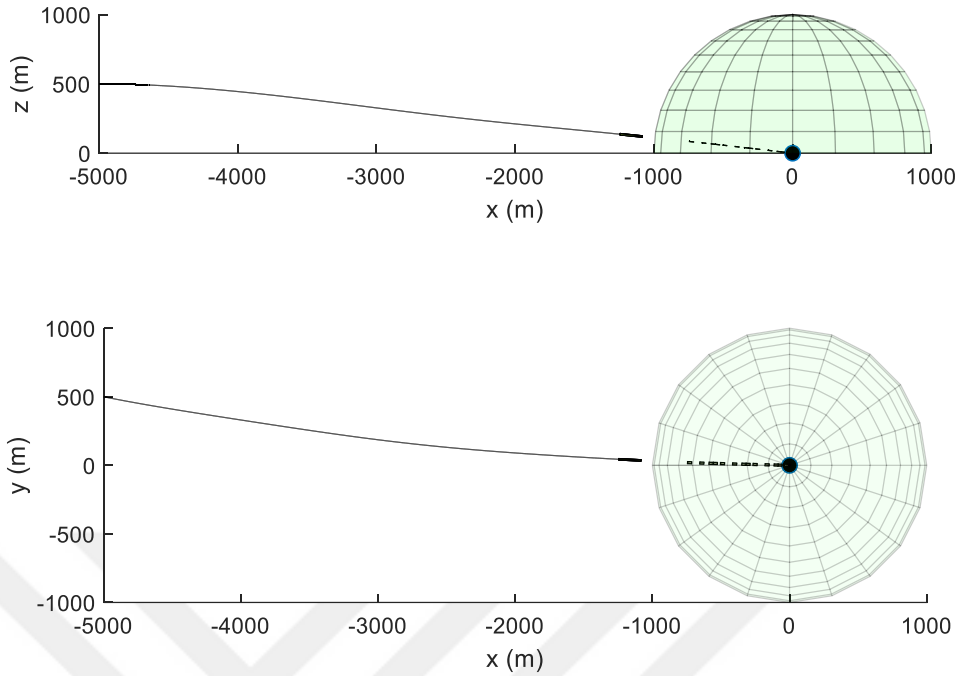

Figure 4.4. Side and Top Views of the Scenario

According to its current orientation, the missile is modeled as a rectangular prism with certain dimensions and located in 3D space. The prism is rotated with transformation matrices and shifted in the same way with the bullet sheets. The missile is accepted to be shot if a 2D bullet sheet intersects with the 3D missile prism at any time. In the simulation, it is checked if there is an intersection between missile and bullets at every time instant. The CIWS succeeds in any intersection, but the number of intersections is used for further analyses. Figure 4.5 shows the intersection detections throughout the flight of the scenario given above.

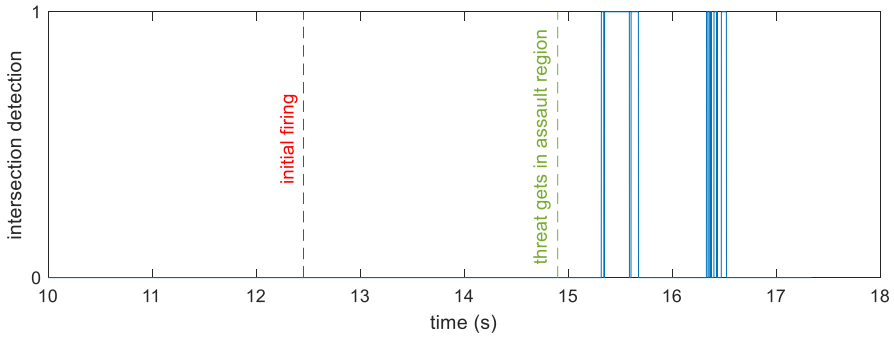

Figure 4.5. Intersection Detections Throughout the Flight

The success rate of the CIWS depends highly on the amplitude of random noise coming from the measurement of position and velocity of the threat. A Monte Carlo simulation is conducted in order to see the effect of the noise amplitude on the success rate. Note that Gaussian distribution is used with zero mean and different variance values for noise.

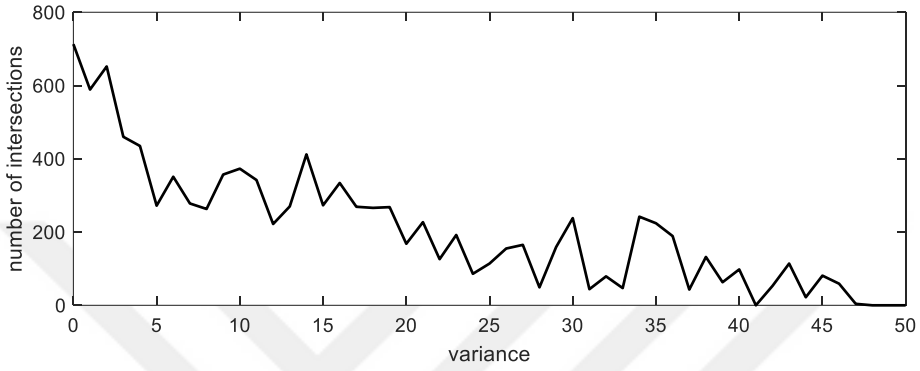

Figure 4.6. Effect of Noise Amplitude on the Success Rate

Although a single intersection between missile prism and bullet sheet is accepted as a shot, the total number of intersections is accepted as an indicator of success rate for a given missile trajectory within a particular time resolution. According to this experiment, the simulation is executed for all of the different variance values and numbers of intersections are stored, as shown in Figure 4.6. After the variance gets 50 meters, there are no intersections left. After all, as the accuracy of the information decreases, the number of intersections logically decreases.

CHAPTER 5

FINAL PHASE PATH PLANNING

In Chapter 2, the mathematical model of an air vehicle is derived, and it is built on Simulink with a guidance system and CAS. Also, some simulations are conducted in order to verify the model and the interaction between the systems. Since the dynamics of missiles are the same as the model built for general air vehicles, it will be used as a model of a missile in this chapter. Moreover, a CIWS system is designed in Chapter 4, and its performance is assessed for a given missile trajectory. In this chapter, evasive maneuvers are investigated by combining missile and CIWS models. Their Simulink models are put together, and the missile's position and velocity are fed to the CIWS model with some sensor noise. The coupled model is given in Figure A.2. In addition, the air vehicle parameters used in all the simulations in this chapter are shown in Appendix C.

In this study, different kinds of maneuvers are investigated, and it is good to have missile and CIWS models together for this purpose. The simulations are based on random movements rather than a particular movement since it becomes completely uncertain even to the attacker side by this way.

Basically, fake targets are generated randomly and given to the guidance system in order to produce random acceleration commands. The real and fake targets switch in a particular period so that a trajectory including evasive maneuvers is obtained. As a result of this, the switching period should be chosen carefully so that the CAS dynamics can handle the changes. In this work, the switching period is selected as 100 ms. In other words, the system switches its target every 0.1 seconds.

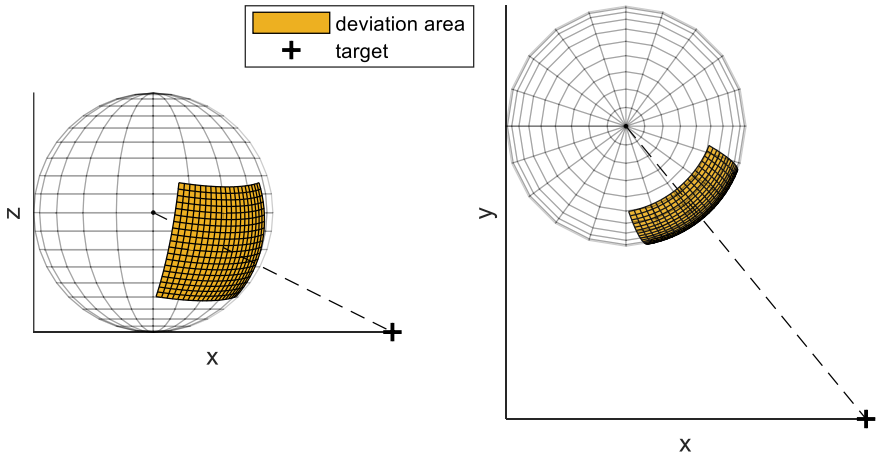

Figure 5.1. The Deviation Area for Strategy I

Fake targets are calculated by deviating the vector towards the target in the direction of pitching moment and yawing moment. The deviation angles are randomly generated between lower and upper boundaries. Thus, boundaries of pitch angle and yaw angle define the deviation area where the fake targets are randomly located, which can be seen in Figure 5.1. Logically, the deviation area must get smaller while the missile gets closer to the target because it should home to the target at the same time. Therefore, the angle boundaries are updated as the missile gets closer to the target, illustrated in Figure 5.2. In addition, the shape of the deviation area is very effective in simulations. In the figures, the areas are one piece, but they also might be multiple sectional areas.

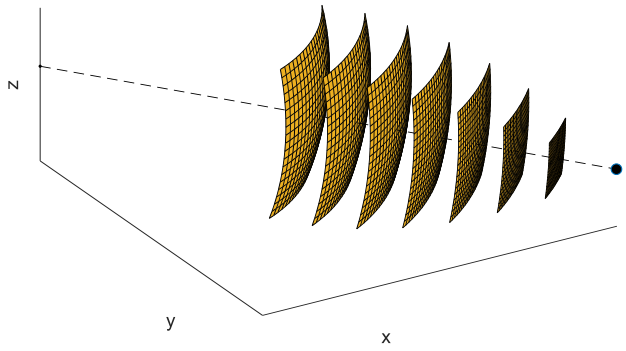

Figure 5.2. The Decreasing Deviation Areas

The number of fake targets given to the guidance system after each real target can be adjusted. For instance, the system might have three randomly chosen fake targets and the real target in a regular turn, and this process continues until hitting the target. The number of fake targets in each turn is called F for the rest of the report.

Consequently, the number of fake targets F in each turn and the shape of the deviation area determines the strategy. They both might cause some breakdowns, such as getting shot or missing the target, and they are all examined in the next section.

5.1 Examination of Different Strategies

5.1.1 Strategy I

The deviation angles are generated in boundaries $\theta \in \left(-\frac{\pi}{6}, \frac{\pi}{6}\right)$, $\psi \in \left(-\frac{\pi}{6}, \frac{\pi}{6}\right)$ and $F = 1$. The corresponding deviation area for the boundaries is shown in Figure 5.1.

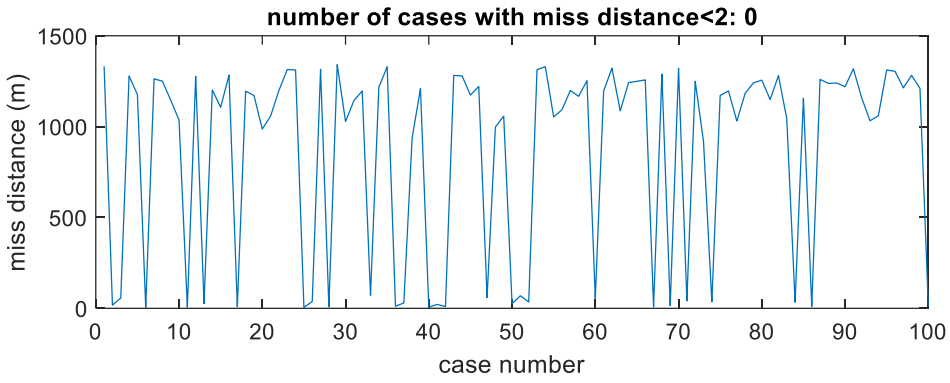

Figure 5.3. Monte Carlo Simulation Result of Strategy I

Figure 5.3 shows the Monte Carlo simulation results with the number of iterations 100 for Strategy I. The aim is to see how many cases become successful. Note that a case is successful if there is no intersection and the miss distance is smaller than 2 meters. From the simulation result, there is no case even reaching the target, as shown in Figure 5.6. Actually, the problem is that the missile goes to the negative side of the z axis, and the simulation terminates. The reason for this is the lower boundary of deviation angle of pitch angle is $-\frac{\pi}{6}$ radians. Thus, it allows the missile to go downward in the vector direction and crash the ground with the help of gravity.

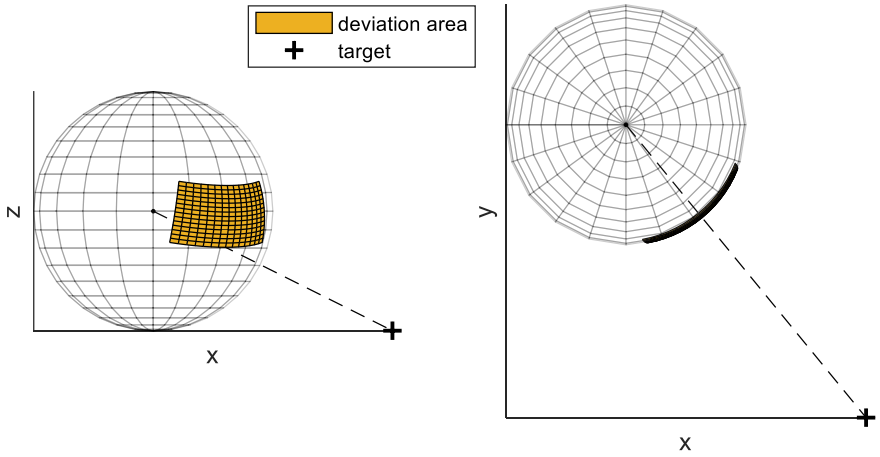

Figure 5.4. The Deviation Area for Strategy II

5.1.2 Strategy II

The deviation angles are generated in boundaries $\theta \in \left(0, \frac{\pi}{6}\right)$, $\psi \in \left(-\frac{\pi}{6}, \frac{\pi}{6}\right)$, $F = 1$. The corresponding deviation area for the boundaries is shown in Figure 5.4.

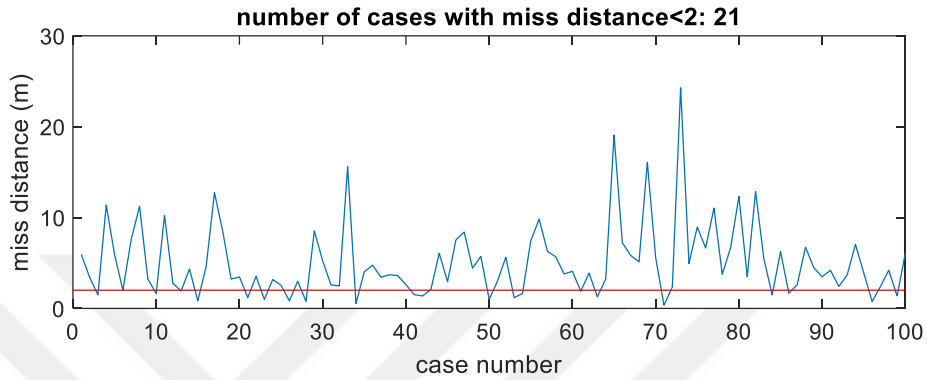

Figure 5.5.Monte Carlo Simulation Result of Strategy II

The lower boundary of deviation angle of pitch angle is changed to 0, and a Monte Carlo simulation is performed. The result is shown in Figure 5.5. According to the results, the mean of the miss distance becomes 5.04 whereas it was 917.28 in Strategy I. Therefore, the change in the boundary brings an obvious improvement. Figures 5.6 and 5.7 show xz view of the trajectories of Strategy I and II. Whereas the trajectory of Strategy I hits the ground before reaching the target, the other reaches the target.

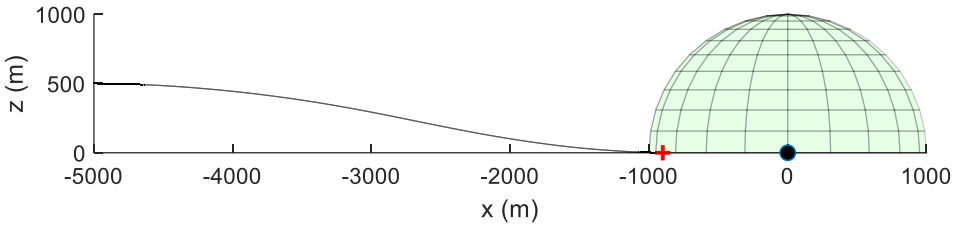

Figure 5.6. xz View of the Trajectories of Strategy I

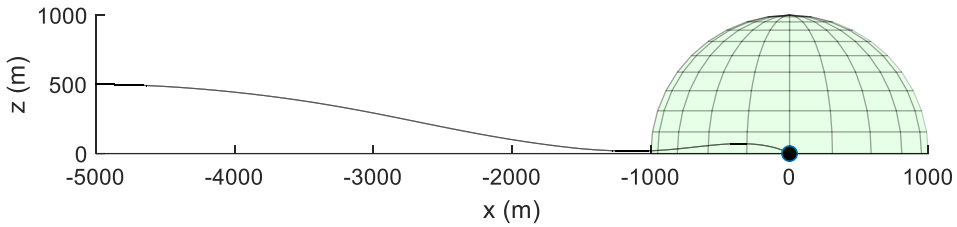

Figure 5.7. xz View of the Trajectories of Strategy II

Apart from this, the number of fake targets F in each turn needs to be adjusted as well for Strategy II. Hence, 10 Monte Carlo simulations are performed with the number of iterations 1000 for the F values 1 to 10 and the results are shown in Figure 5.8. According to the results, as the F increases, the success rate of miss distance is getting better. Also, there is no intersection in about 25% of the cases for the F values greater than 5.

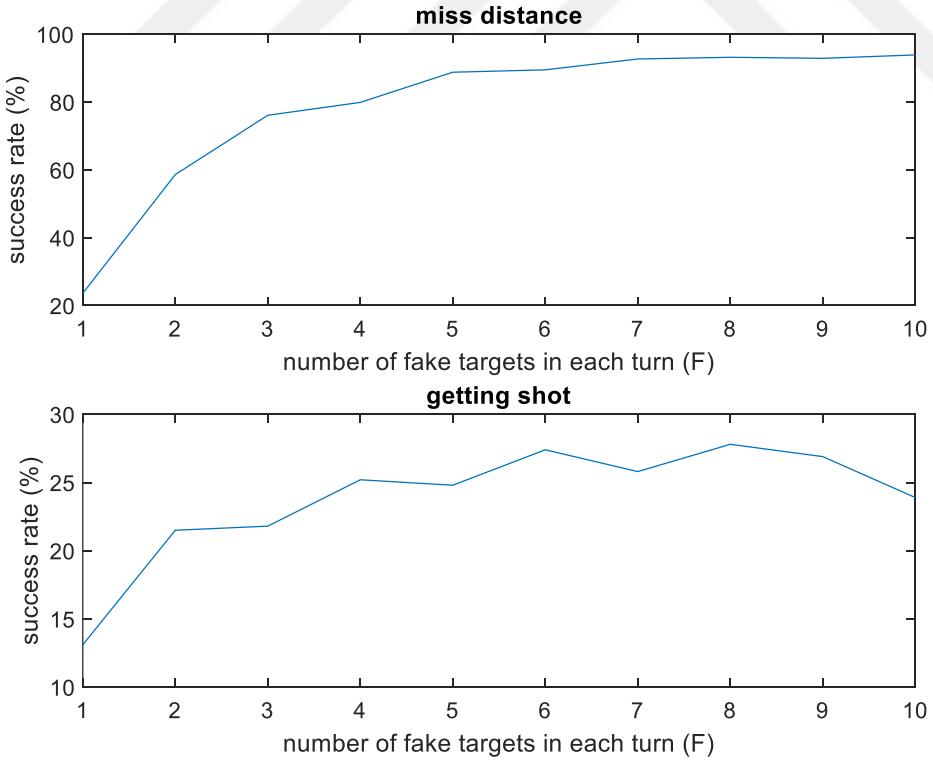

Figure 5.8. Monte Carlo Simulation Results for Different F Values

In this strategy, the system fails likely even with F's best value because the missile gets shot by the CIWS easily. The deviation areas were one piece so far. However, the successive fake targets might be generated close to each other, which causes the maneuver not to be large enough to avoid the attacks. Since tens of fake targets are generated in one flight, this case happens very likely, and it presents a threat to the missile itself. This is the reason for having a low success rate of getting shot in Strategy II. The system needs to make sure that it carries out large maneuvers in both axes in order to eliminate this threat.

5.1.3 Strategy III

The deviation angles are generated in boundaries $\theta \in \left(\frac{\pi}{18}, \frac{\pi}{6}\right)$, $\psi \in \left\{\left(-\frac{\pi}{6}, -\frac{\pi}{18}\right) \cup \left(\frac{\pi}{18}, \frac{\pi}{6}\right)\right\}$.

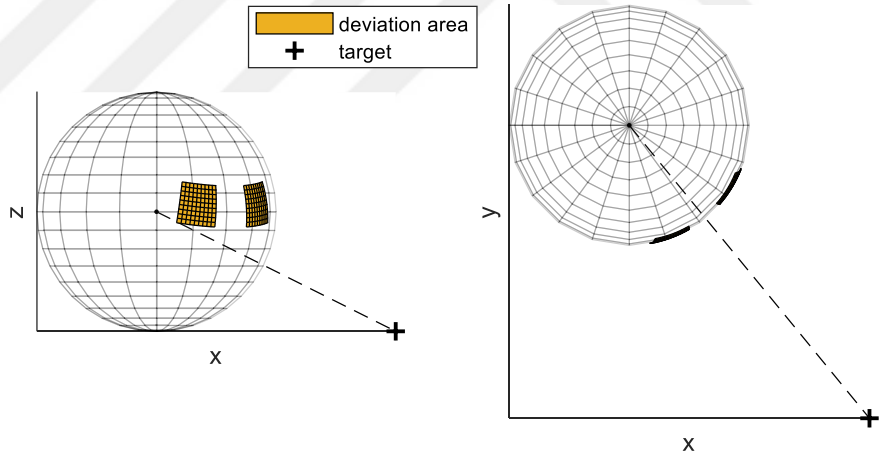

Figure 5.9. The Deviation Area for Strategy III

Figure 5.9 shows the deviation area for Strategy III. Two separate areas provide large maneuvers in the direction of yawing. Firstly, the lower bound of the pitching deviation angle is set to $\pi/18$. Hence, it allows the missile to have a larger maneuver in z-axis in the last stage because it dives to the target from a higher altitude, as

shown in Figure 5.13(b). Secondly, the piecewise angles of ψ state two symmetric deviation areas, as shown in Figure 5.9. Thus, it creates a gap between the symmetric areas, and the missile can have a larger maneuver in xy plane, as shown in Figure 5.13(a).

Figure 5.10. The Actual and Desired Accelerations in y-axis for Strategy II

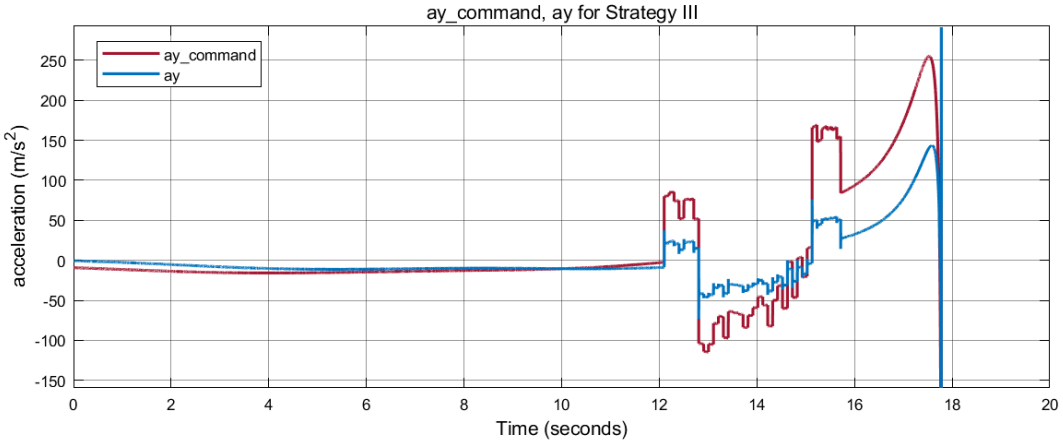

Figure 5.11. The Actual and Desired Accelerations in y-axis for Strategy III

In this strategy, the system guides towards only fake targets until a predefined distance value to the target. The value is set to 400 meters, and the missile guides towards the real target after the remaining distance becomes smaller than this value. In addition, if the fake targets are still randomly picked from the symmetric areas,

this strategy would not make a big difference because when the missile starts to go towards one side, it will probably have a fake target generated on the other side in the next step. Thus, it would not be able to have a nice maneuver towards one side. In order to overcome this problem, the fake targets are generated on one side for a number of times before it passes to the other side. The number R states how many times the system goes one side, and this is also randomized between 5 and 15 for each side. The way of selecting the fake targets in a row can be seen from a_y, $a_{y_{command}}$ signals for Strategy II and III in Figures 5.10 and 5.11. For Strategy II, fake targets are generated on both negative and positive sides, and the missile follows the command signal pretty well. However, it does not switch the sides very frequently in Strategy III. If it goes towards one side, it keeps going for a while. In this way, the missile has enough time to carry out the maneuver towards one side. Figure 5.12 shows the trajectory of Strategy III, together with Strategy II's trajectory which is given in Figure 5.7. As shown in the figure, the trajectory of Strategy III carries out a larger maneuver.

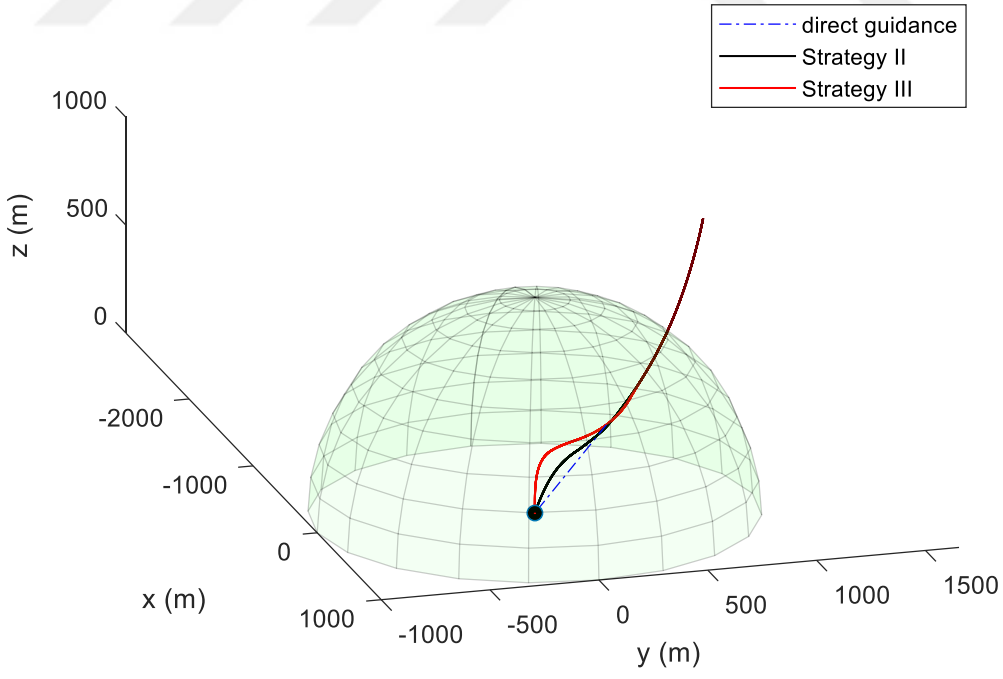

Figure 5.12. The Trajectories of Strategy II and III

Figure 5.13 indicates the xy and xz views of the same trajectories. In xy plane, the trajectory of Strategy III takes a slight left at first, goes towards to right side secondly and completes the flight by reaching the target. In xz plane, some sharper maneuvers are obtained with Strategy III.

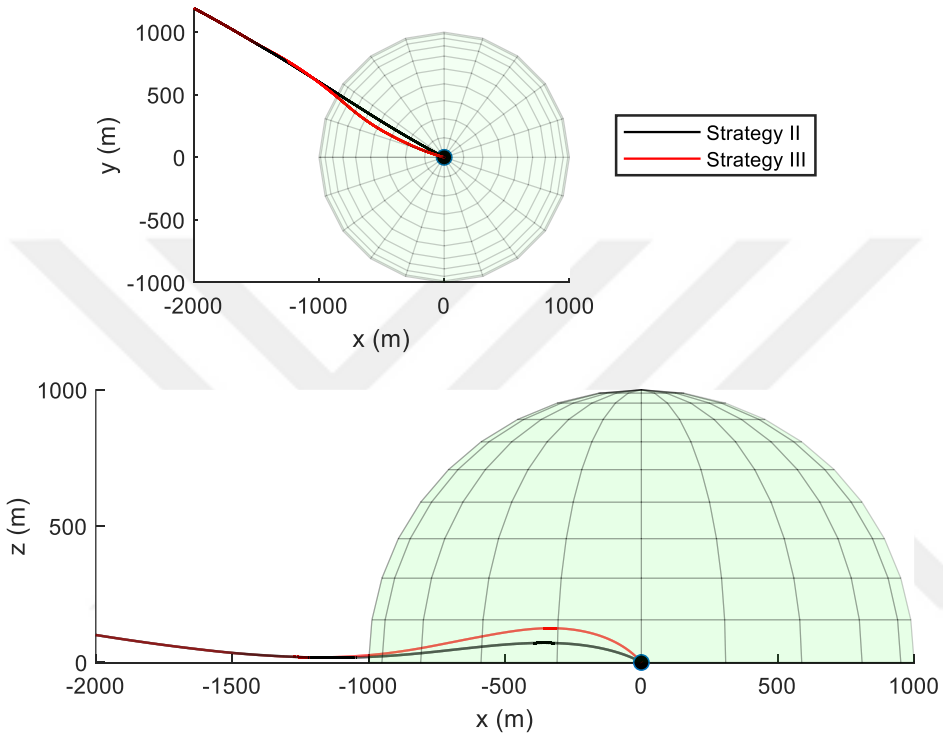

Figure 5.13. xy and xz Views of the Trajectories

A Monte Carlo simulation with 100 iterations is performed to see the performance of Strategy III, as shown in Figure 5.14. According to the results, the success rate of miss distance is 94%, which is much better than Strategy II. Also, it has 100% accuracy in reaching the target successfully. It reaches the target without getting shot for most of the cases with Strategy III. Therefore, having the largest possible maneuvers without missing the target is the key point of the evasive maneuver of missiles, and it is very effective against CIWS.

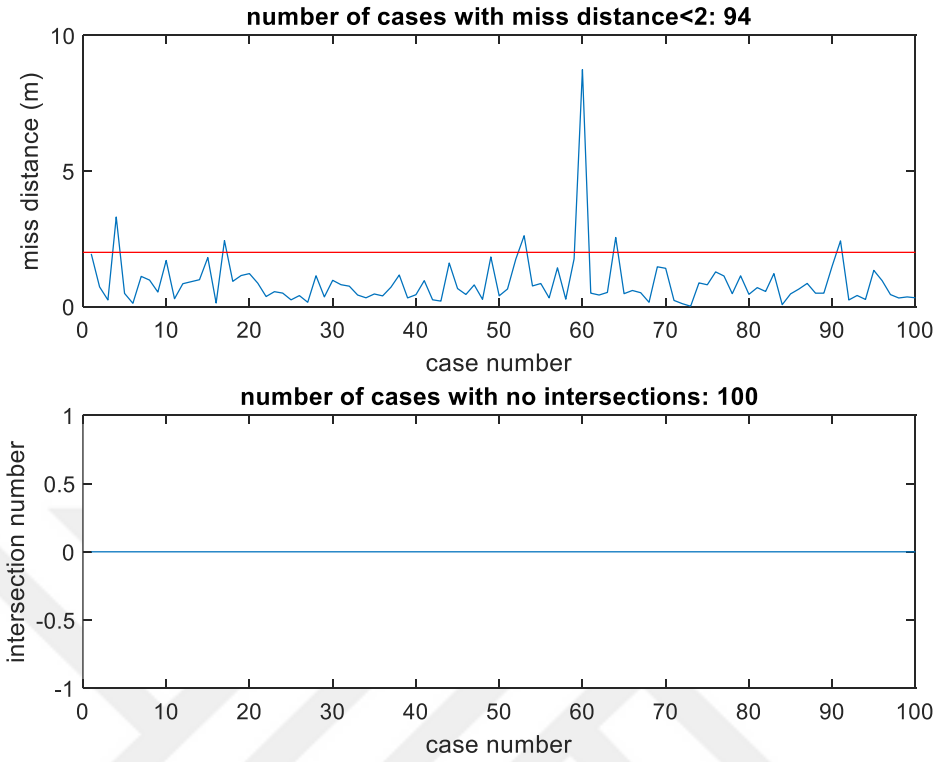

Figure 5.14. Monte Carlo Simulation Result of Strategy III

CHAPTER 6

DISCUSSION AND CONCLUSIONS

In this thesis, midcourse and final phase path planning is dealt with. The two phases state two different problems and require separate solutions. In the midcourse phase, paths with the minimum length, fuel consumption, and risk of being detected are looked for. The dynamics of the air vehicle are represented by their geometric equivalents in this phase in order to get the solution as fast as possible. In other words, the dynamics are added to the problem as the constraints of the optimization algorithm. On the other hand, mathematical models of air vehicles, guidance systems, and CAS are integrated to use against the mathematical model of the CIWS. All the subsystems are built on MATLAB Simulink and run simultaneously. Since the aim is to measure and develop the performance of the air vehicle, it is not much possible without having an air defense system. Therefore, several evasive maneuver strategies are investigated in order to avoid the assault of the CIWS. After the general scope of the thesis is given, detailed examinations are discussed in each chapter of the thesis.

In Chapter 2, some necessary mathematical tools are described and the general equations of motion of a standard air vehicle are derived step by step. After this, the aerodynamic model is integrated with the equations of motion and the vehicle becomes a controllable system with the deflection angles ($\delta_e, \delta_a, \delta_r$) and the thrust force. Besides, guidance and CAS subsystems are developed and attached to the overall system. At the end of the chapter, the demonstration of the whole model is carried out in a way that an air vehicle guides towards a target point, and the signals are examined. For a trajectory that the air vehicle reaches the target successfully, the Euler angles, commanded signals produced by the guidance system and the actual

signals controlled by the CAS were compatible with the trajectory and it proves the model operates properly in a practical manner.

The midcourse path planning is elaborated in Chapter 3. The optimization problem formulation is given with separate costs and constraints. After that, the infinite-dimensional problem is discretized and the finite-dimensional problem is obtained, whose formulation is also given. Then, geometry in turning, climbing, and diving maneuvers is explained clearly. Eventually, the Real Coded Genetic Algorithm is selected as the optimization method and its usage on path planning problem is described. Since there has to be an initial population in the Genetic Algorithm, an initial population creation method that only produces feasible solutions is set forth. Thereafter, the classical crossover and mutation operators are shown how to be utilized in this problem. Also, new problem dependent operators have been defined thanks to the flexible Genetic Algorithm and they make the algorithm capable of converging more accurately and faster. Lastly, some experiments are conducted to verify the performance of the algorithm. According to the results, the algorithm works very well even for very complicated environments. Some Monte Carlo simulations are carried out and their outcomes show the algorithm finds the global optimum solution in most cases. Besides, the effects of the separate cost functions are examined in the Multi-objective Optimization section.

For further analyses, simulations with DTED maps might be conducted to see the performance of the algorithm in real environments. The average computation time is 15 seconds in the simulations, but it might be 5 minutes with DTED maps because real environments are much more detailed. In such a case, there might be some enhancements for decreasing the number of intersection checks to improve computation time.

In Chapter 4, a gun-based CIWS is modeled mathematically and used for defending against a missile model that is the same derived for air vehicles in Chapter 2. The calculation of the time duration is very crucial to model the timing properly. The position and the velocity of the missile are estimated by the CIWS with some

measurement noise and the intersection point is calculated by accepting it moves in a linear direction. In this way, the CIWS shoots at the intersection point at the right time. In order to see the effect of amplitude of the measurement noise, a Monte Carlo simulation is carried out for variance values 0 to 50 meters. As expected, the performance of the CIWS decreases as the variance goes up. Anyway, the success rate is quite acceptable with a variance of 5 meters or less.

The models from Chapters 2 and 4 are integrated and the missile and the CIWS are made work against each other in this way in Chapter 5. When the missile directly guides towards the target without any maneuvers, it gets shot undoubtedly as shown in Section 4.4. Thus, evasive maneuvers are necessary for the missile to avoid the assault of the CIWS. In this study, evasive maneuvers are created in a way that fake targets are randomly selected from deviation areas in a certain period and these targets are given to the guidance system one by one. In this way, the missile goes towards the fake targets before the real one and carries out an unexpected maneuver. These maneuvers highly depend on the shape and the size of the deviation areas because the fake targets are generated on them. Several tactical strategies are investigated to find the best one. According to the results, the best way is to have separate areas that are far away from each other. Because it provides the missile to have large movements towards one side. Also, switching the area should not be very frequent because if it switches the areas too often, the missile would not have enough time to go towards one side.

The work on creating different strategies for both offensive and defensive sides will continue in the future. It is planned to discuss the effects of using more than one offensive vehicle and the other countermeasure techniques simultaneously.

REFERENCES

[1]. Persson, S. M., & Sharf, I. (2014). Sampling-based A* algorithm for robot path-planning. *The International Journal of Robotics Research, 33*(13), 1683-1708. doi:10.1177/0278364914547786

[2]. Yang, X., Ding, M., & Zhou, C. (2010). Fast Marine Route Planning for UAV Using Improved Sparse A* Algorithm. *2010 Fourth International Conference on Genetic and Evolutionary Computing*. doi:10.1109/icgec.2010.54

[3]. Tanil, Ç. (2012). Cooperative Path Planning for Multiple Missiles - An Evolutionary Speciation Approach. *AIAA Guidance, Navigation, and Control Conference*. doi:10.2514/6.2012-4903

[4]. Dai, X., Long, S., Zhang, Z., & Gong, D. (2019). Mobile Robot Path Planning Based on Ant Colony Algorithm With A* Heuristic Method. *Frontiers in Neurorobotics, 13*. doi:10.3389/fnbot.2019.00015

[5]. Masehian, E., & Sedighizadeh, D. (2007). Classic and Heuristic Approaches in Robot Motion Planning – A Chronological Review. *World Academy of Science, Engineering and Technology*. doi:10.5281/zenodo.1074972

[6]. Lamini, C., Benhlima, S., & Elbekri, A. (2018). Genetic algorithm based approach for autonomous mobile robot path planning. *Procedia Computer Science, 127*, 180-189. doi:10.1016/j.procs.2018.01.113

[7]. Nagib, G., & Gharieb, W. (n.d.). Path planning for a mobile robot using genetic algorithms. *International Conference on Electrical, Electronic and Computer Engineering, 2004. ICEEC '04.* doi:10.1109/iceec.2004.1374415

[8]. Xue, Y., & Sun, J. (2018). Solving the path planning problem in mobile robotics with THE Multi-Objective evolutionary algorithm. *Applied Sciences, 8*(9), 1425. doi:10.3390/app8091425

[9]. Berglund, E. Guidance and Control Technology. In: RTO SCI Lecture Series on Technologies for Future Precision Strike Missile Systems, Atlanta, GA, 2000, pp.1-10.

[10]. The Naval Institute Guide to World Naval Weapon Systems. (2017, November). Retrieved January, 2021.

[11]. Ender, T. (2006). *A Top-Down, Hierarchical, System-of-Systems Approach to the Design of an Air Defense Weapon*. Atlanta, GA: Georgia Institute of Technology.

[12]. Ma, W., Zhang, Y., Duan, J., Yang, B., & Huo, J. (2010). Design of General and Digital Simulation Software for Close-In Weapon System. *2010 International Conference on Computational Intelligence and Software Engineering*. doi:10.1109/cise.2010.5677170

[13]. Wang, Z., Zhang, Y., & Yang, B. (2010). Study on a Closed-Loop Fire Correction Algorithm in Vehicle Close-In Weapon System. *2010 International Conference on Computational Intelligence and Software Engineering*. doi:10.1109/cise.2010.5676756

[14]. Zarchan, P. (1995). Proportional navigation and weaving targets. *Journal of Guidance, Control, and Dynamics, 18*(5), 969-974. doi:10.2514/3.21492

[15]. Imado, F., & Uehara, S. (1998). High-g Barrel Roll Maneuvers Against Proportional Navigation from Optimal Control Viewpoint. *Journal of Guidance, Control, and Dynamics, 21*(6), 876-881. doi:10.2514/2.4351

[16]. Cook, M. (2008). *Flight dynamics principles*. Amsterdam: Elsevier.

[17]. Design of lightweight airborne mmw radar for dem generation. Simulation results. (n.d.). Retrieved January 24, 2021, from http://muhaz.org/design-of-lightweight-airborne-mmw-radar-for-dem-generation-si.html?page=3

[18]. Appendc. (n.d.). Retrieved January 24, 2021, from https://history.nasa.gov/SP-367/appendc.htm#f165

[19]. Ly, U. (1997). *Stability and Control of Flight Vehicle*. Seattle, USA: University of Washington.

[20]. Etkin, B., & Reid, L. D. (1996). *Dynamics of flight: Stability and control*. New York: Wiley.

[21]. Zarchan, P. (1997). *Tactical and strategic missile guidance*. Reston, VA: American Inst. of Aeronautics and Astronautics.

[22]. Moran, I., & Altilar, T. (2005). Three Plane Approach for 3D True Proportional Navigation. *AIAA Guidance, Navigation, and Control Conference and Exhibit*. doi:10.2514/6.2005-6457

[23]. Boiffier, J. (1998). *The dynamics of flight: The equations*. Chichester: Wiley.

[24]. Fattahi, E., Pahlavani, M., & Nekoei, M. A. (2013). *A New Proportional Guidance Law Design In The Presence of Control Loop Dynamics against Maneuvering Targets*. Tehran, Iran.

[25]. Dubins, L. E. (1957). On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents. *American Journal of Mathematics, 79*(3), 497. doi:10.2307/2372560

[26]. Dubins path. (2021, January 14). Retrieved January 24, 2021, from https://en.wikipedia.org/wiki/Dubins_path

[27]. Crossover. (2020, November 06). Retrieved January 24, 2021, from https://en.wikipedia.org/wiki/Crossover

[28]. Yoon, Y., & Kim, Y. (2012). *The Roles of Crossover and Mutation in Real-Coded Genetic Algorithms*. INTECH Open Access Publisher.

[29]. Zitzler, E., & Thiele, L. (1998). *An Evolutionary Algorithm for Multiob jective Optimization The Strength Pareto Approach*. Zurich, Switzerland: Computer Engineering and Communication Networks Lab.

APPENDICES

A. Simulink Models

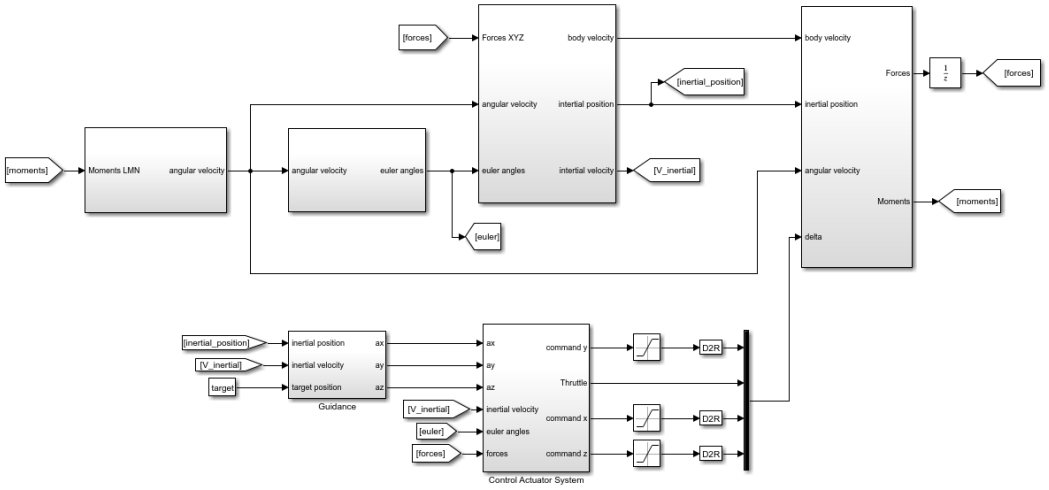

Figure A.1. The Overall Simulink Model of an Air Vehicle

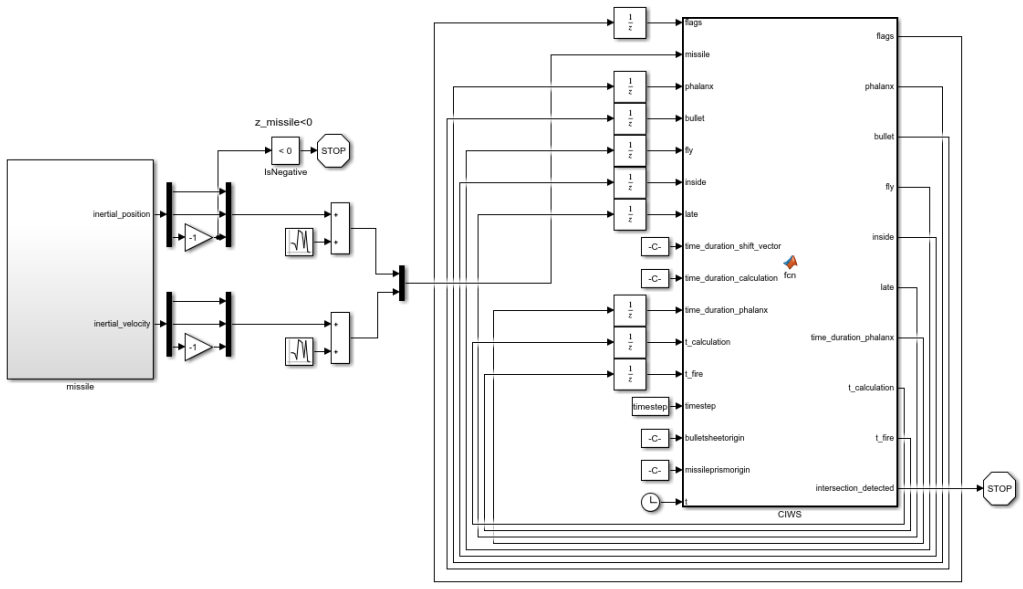

Figure A.2. The Coupled Simulink Model for Analyzing Evasive Maneuver

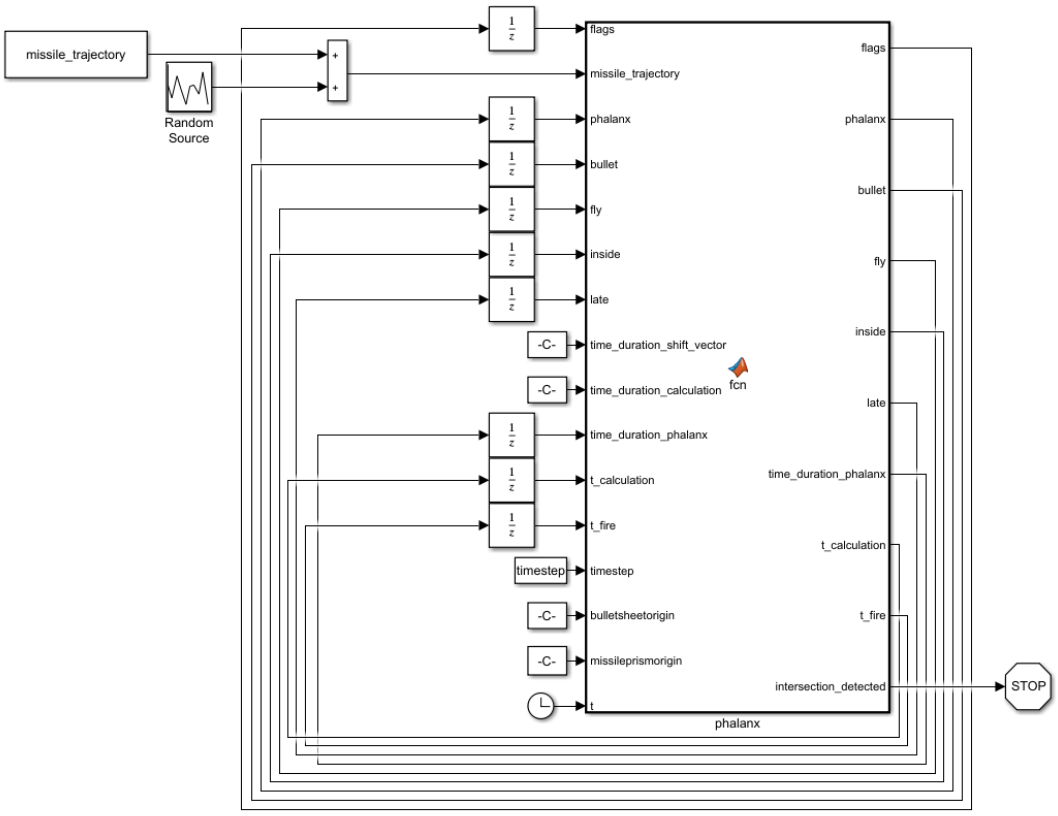

Figure A.3. Simulink Model of the CIWS

B. Configuration Parameters of the Genetic Algorithm

generation number	50
population size	80
shifting length	500 meters
elite rate	0.1
mutation rate	0.5
mutation distance	50 meters
mutation angle	2π radians
climbing angle	0.5 radians
height margin	0.10
gps error	20 meters
extend distance	100 meters
placedistafterdive	200 meters
mindistafterstart	100 meters
mindistbetweenturnings	100 meters
mindistbeforefinal	100 meters
mindistbeforeclimb	100 meters
mindistafterdive	100 meters
mindistoverobstacle	100 meters
radius	50 meters
step angle	0.2 radians
maxdeviation angle	$3\pi/4$ radians
cost weights	1/3

C. Missile and CIWS Parameters Used in the Simulations

m	514.34 kg
d	0.125 m
S	0.01227 m^2
Ix	19.18 kg.m^2
Iy	837.13 kg.m^2
Iz	837.13 kg.m^2
Cd	0.53
$C_{y_{beta}}$	-6 rad^{-1}
$C_{y_{rudder}}$	6 rad^{-1}
$C_{L_{alpha}}$	6.3 rad^{-1}
$C_{L_{elevator}}$	6 rad^{-1}
C_{L_q}	5.4 rad^{-1}
C_{l_p}	-6.6 rad^{-1}
$C_{l_{aileron}}$	6.3 rad^{-1}
$C_{m_{alpha}}$	-5.6 rad^{-1}
C_{m_q}	-250 rad^{-1}
$C_{m_{elevator}}$	6 rad^{-1}
$C_{n_{beta}}$	5.6 rad^{-1}
C_{n_r}	-250 rad^{-1}
$C_{n_{rudder}}$	6 rad^{-1}
length	5 meters
width	0.5 meters
height	0.5 meters

w_{CIWS}	0.69 rad/s
V_{bullet}	500 m/s
$length_{sheet}$	8 meters
$width_{sheet}$	2 meters