AUTONOMOUS AND MANUAL DRIVING OF A MULTIPLE TURRET SYSTEM IN EXTREME ENVIRONMENT

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

BY

ÜMİT YERLİKAYA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
MECHANICAL ENGINEERING

JUNE 2021

Approval of the thesis:

AUTONOMOUS AND MANUAL DRIVING OF A MULTIPLE-TURRET SYSTEM IN EXTREME ENVIRONMENT

submitted by **ÜMİT YERLİKAYA** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy** i**n Mechanical Engineering, Middle East Technical University** by,

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

Prof. Dr. M. A. Sahir Arıkan
Head of the Department, **Mechanical Engineering** _____

Prof. Dr. R. Tuna Balkan
Supervisor, **Mechanical Engineering Dept., METU** _____

Examining Committee Members:

Assoc. Prof. Dr. Kıvanç Azgın
Mechanical Engineering Dept., METU _____

Prof. Dr. R. Tuna Balkan
Mechanical Engineering Dept., METU _____

Assist. Prof. Dr. Hakan Çalışkan
Mechanical Engineering Dept., METU _____

Assoc. Prof. Dr. S. Çağlar Başlamışlı
Mechanical Engineering Dept., Hacettepe University _____

Assist. Prof. Dr. Özgür Ünver
Mechanical Engineering Dept., Hacettepe University _____

Date: 14.06.2021

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 : Ümit Yerlikaya

Signature :

ABSTRACT

AUTONOMOUS AND MANUAL DRIVING OF A MULTIPLE-TURRET SYSTEM IN EXTREME ENVIRONMENT

Yerlikaya, Ümit
Doctor of Philosophy, Mechanical Engineering
Supervisor: Prof. Dr. R. Tuna Balkan

June 2021, 170 pages

In this thesis, firstly two methods are developed to obtain multi-dimensional configuration space for path planning problems. In typical cases, the path planning problems are solved directly in the 3-D workspace. However, this method is inefficient in handling the robots with various geometrical and mechanical restrictions. To overcome these difficulties, path planning may be formalized and solved in a new space which is called configuration space. In the first method, the point clouds of all the bodies of the system and interaction of them are used. The second method is performed via using the clearance function of simulation software where the minimum distances between surfaces of bodies are simultaneously measured. A sample 4-D configuration space of a double-turret system is obtained in these two methods. As a result of this, the difference between these two ways is about 1% which depends on the point cloud density.

Then, Instead of using the tedious process of manual positioning, an off-line path planning algorithm has been developed for military turrets to improve their accuracy and efficiency. In the scope of this research, an algorithm is proposed to search a path in three different types of configuration spaces which are rectangular, circular

and torus shaped by providing three converging options named as fast, medium and optimum depending on the application. With the help of the proposed algorithm, 4-dimensional (D) path planning problem was realized as 2-D + 2-D by using 6 sequences and their options. The results obtained were simulated and no collision was observed between any bodies in these three options.

Finally, with the help of new collision avoidance algorithm, all types of turrets can be driven more efficiently and safely according to the specified speed, acceleration and jerk limits. Since all possible worst scenarios are examined one by one, it is guaranteed that the algorithm provides collision free motion in both simulations and real-time tests. A configuration space where worst scenarios can occur is created for the performance measurement of the algorithm, and the same space is used in all tests. As a result of these tests, it is shown that there is no collision. Finally, by adding cascade position control loop, the departure from the starting point to the desired target point is achieved without any collision. The most important feature that distinguishes this algorithm from others is both speed and position can be controlled and during transition phase, the target point can be changed instantly. In addition, no target position is required for the system to move collision-free, only axis speed commands are sufficient. Since the algorithm does not intervene in the speed and torque loops in contrast to potential field-based methods, it can be added to ready-to-use systems by manipulating only the speed references.

Keywords: End Damping Algorithm, Collision Avoidance, Configuration Space, Path Planning, Stabilized Gun Turrets

ÖZ

ÇOKLU BİR KULE SİSTEMİNİN AŞIRI ORTAMDA OTONOM VE MANUEL SÜRÜŞÜ

Yerlikaya, Ümit
Doktora, Makina Mühendisliği
Tez Yöneticisi: Prof. Dr. R. Tuna Balkan

Haziran 2021, 170 sayfa

Bu tezde öncelikle yol planlama problemleri kapsamında çok boyutlu konfigürasyon uzayı elde etmek için iki yöntem geliştirilmiştir. Yol planlama problemleri genellikle doğrudan 3 boyutlu çalışma alanında çözülür. Ancak bu yöntem, çeşitli geometrik ve mekanik kısıtlamalara sahip robotların işlenmesinde yetersiz kalmaktadır. Bu zorlukların üstesinden gelmek için, yol planlaması biçimselleştirilebilir ve konfigürasyon uzayı adı verilen yeni bir uzayda çözülebilir. Birinci yöntemde, sistemin tüm gövdelerinin nokta bulutları ve bunların etkileşimi kullanılır. İkinci yöntem, vücut yüzeyleri arasındaki minimum mesafelerin aynı anda ölçüldüğü simülasyon yazılımının boşluk fonksiyonu kullanılarak gerçekleştirilir. Bu iki yöntemde bir çift taret sisteminin örnek bir 4-D konfigürasyon alanı elde edilir. Bunun bir sonucu olarak, bu iki yol arasındaki fark, nokta bulutu yoğunluğuna bağlı olarak yaklaşık %1'dir.

Ardından, sıkıcı manuel konumlandırma sürecini kullanmak yerine, askeri taretlerin doğruluklarını ve verimliliklerini artırmak için çevrimdışı bir yol planlama algoritması geliştirilir. Bu araştırma kapsamında, uygulamaya bağlı olarak hızlı, orta ve optimum olmak üzere üç yakınsayan seçenek sağlanarak dikdörtgen, dairesel ve torus şekilli olmak üzere üç farklı konfigürasyon uzayında yol aramak için bir

algoritma önerilmiştir. Önerilen algoritma yardımıyla 6 dizi ve seçenekleri kullanılarak 4 boyutlu (D) yol planlama problemi 2-D + 2-D olarak gerçekleştirilmiştir. Elde edilen sonuçlar simüle edildi ve bu üç seçenekte hiçbir cisim arasında çarpışma gözlemlenmedi.

Son olarak, yeni çarpışmadan kaçınma algoritması sayesinde her türlü taret belirtilen hız, ivme ve sarsıntı limitlerine göre daha verimli ve güvenli bir şekilde sürülebilmektedir. Olası tüm en kötü senaryolar tek tek incelendiği için hem simülasyonlarda hem de gerçek zamanlı testlerde algoritmanın çarpışmasız hareket sağlaması garanti edilmektedir. Algoritmanın performans ölçümü için en kötü senaryoların oluşabileceği bir konfigürasyon alanı oluşturulur ve tüm testlerde aynı alan kullanılır. Bu testler sonucunda çarpışma olmadığı gösterilmiştir. Son olarak, kademeli konum kontrol döngüsü eklenerek, herhangi bir çarpışma olmadan başlangıç noktasından istenen hedef noktasına hareket sağlanır. Bu algoritmayı diğerlerinden ayıran en önemli özelliği hem hız hem de konumun kontrol edilebilmesi ve geçiş aşamasında hedef noktasının anlık olarak değiştirilebilmesidir. Ayrıca sistemin çarpışmasız hareket etmesi için herhangi bir hedef pozisyon gerekli değildir, sadece eksen hız komutları yeterlidir. Algoritma potansiyel saha tabanlı yöntemlerin aksine hız ve tork döngülerine müdahale etmediği için sadece hız referansları manipüle edilerek kullanıma hazır sistemlere eklenebilir.

Anahtar Kelimeler: Son Kontrol Sönümleme Algoritması, Çarpışmadan Kaçınma, Konfigürasyon Uzayı, Yol Planlama, Stabilize Silah Taretleri

To my beloved son Ayaz

ACKNOWLEDGMENTS

First of all, I would like to express my sincere appreciation to my thesis supervisor Prof. Dr. R. Tuna BALKAN for his guidance, advice, criticism, encouragements and insight throughout the research

I wish to express my gratitude to the members of the thesis supervision committee, Assoc. Prof. Dr. Kıvanç AZGIN and Assist. Prof. Dr. Özgür ÜNVER for their guidance, suggestions and comments throughout my thesis study. I would also like to thank Assoc. Prof. Dr. S. Çağlar BAŞLAMIŞLI for helping me look at events from a different perspective in my work.

I also owe thanks to my colleagues in my unit in FNSS Defense Systems Inc, who helped me in every subject throughout the years of my study.

Then, I would like to thank Scientific and Technological Research Council of Turkey (TUBİTAK) for scholarship and financial supports which made this project possible. I hope their endless support to science and scientists continue to enrich our scientific heritage.

No words are sufficient to express my gratitude to my father Ahmet, my mother Nigar, who made me the man who I am today.

I am especially grateful to my lovely wife Büşra for her love, care, patience, and enormous support. Without her, this thesis would have never been completed. Last but not least, I thank my precious two sons Onur and Ayaz for their loves and wonderful smiles that have been my fuel at times I felt exhausted. Lastly, I dedicate this thesis work to my beloved newborn son Ayaz.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS

AABB	Axis Aligned Bounding Box
BA	Barrel
BV	Bounding Volume
C-SPACE	Configuration Space
CW	Clockwise
CCW	Counter-clockwise
D	Dimension
DOF	Degree of Freedom
DWA	Dynamic Window Approach
FPC	Fixed Point cloud
G	Gun
H	Hull
IB	Intersection Box
LL	Lower-left
LR	Lower-right
MPC	Moving Point Cloud
O	Obstacle
PC	Point Cloud
PRM	Probabilistic Roadmap
PSO	Particle Swarm Optimization
RCT	Rectangular-Circular-Torus Shaped
RRT	Rapidly Exploring Random Tree
R-P	Revolute-Prismatic
T	Turret
UL	Upper-left
UR	Upper-right
VFH	Vector Field Histogram
VO	Velocity Obstacle

LIST OF SYMBOLS

SYMBOLS

$(C_{k,k+1})^{fix}$	Fix 4-D C-space of "k" and "k+1" turret pair
$(C_{k,k+1})^{alive}$	Live 4-D C-space of "k" and "k+1" turret pair which changes according to instant positions of "k-1" and "k+2" turrets
X_{in}, Y_{in}, Z_{in}	Column vectors consist of x, y and z coordinates of i-th point cloud with length of n [mm]
X_{km}, Y_{km}, Z_{km}	Column vectors consist of x, y and z coordinates of k-th point cloud with length of m [mm]
D	Distance matrix between two point clouds [mm]
d_e^k	Number of grids on elevation axis of C-space of k-th turret in the aligned multi-turret system
d_t^k	Number of grids on traverse axis of C-space of k-th turret in the aligned multi-turret system
d_e	Distance between equivalent look ahead position and top of obstacle [deg]
d_m	Minimum distance between two mesh points in a point cloud
d_t	Distance between equivalent look ahead position and closest side of obstacle [deg]
g_s	Increase in safe distance (s_d) [grid]
i_{end}	Number of repeated C-spaces on the horizontal axis
j_{end}	Number of repeated C-spaces on the vertical axis
n_c	Number of equivalent look ahed positions of turret
n_o	Number of obstalce in the C-space
O_k	k-th obstacle in the the aligned multi-turret system
PC_H	A point cloud in Cartesian coordinates ($n \times 4$)
R_k	Rotation matrix about axis-k (x, y, z)
s_d	Safe distance given as grid for obstacle grouping [grid]
s_k	Size of k-th variable [grid]
S^1	Circle description

T^2	Two-dimensional torus surface
T_k	k-th turret in the aligned multi-turret system
t_e	Torque input of traverse axis [Nm]
t_s	Time step [s]
t_t	Torque input of traverse axis [Nm]
T_V	Translational matrix along V vector
V_k	Cartesian based rotation vector at axis-k (x, y, z)
w_t^o	Width of obstacle on the traverse axis [deg]
$\delta\theta$	Step angles of n variables [deg]
θ^1	Traverse angle of first turret of double-turret system [deg]
θ^2	Elevation angle of first turret of double-turret system [deg]
θ^3	Traverse angle of second turret of double-turret system [deg]
θ^4	Elevation angle of second turret of double-turret system [deg]
θ_s	Start positions of axes for path planning [deg]
θ_t	Target positions of axes for path planning [deg]
θ_g	Path planning solution as grid base [grid]
θ'	Path planning solution as angle base [deg]
θ_{in}	Initial values of n variables [deg]
θ_{end}	Final values of n variables [deg]
θ_{sg}	The starting grids of the variables [grid]
θ_{tg}	The target grids of the variables [grid]
θ_{og}	The obstacle grids in the C-space [grid]
θ_e^k	Elevation angle of k-th turret in the multi-turret sys [deg]
θ_t^k	Traverse angle of k-th turret in the multi-turret sys [deg]
Δ_t^{la}	Look ahead distance of traverse axis [deg]
Δ_e^{la}	Look ahead distance of elevation axis [deg]
Δ_c^{la}	Equivalent look ahead distance of turret [deg]
Δ_a	Vertical distance between instant elevation angle and top of obstacle only during avoiding phase [deg]
p_t^{ref}	Position reference of traverse axis [deg]
p_e^{ref}	Position reference of elevation axis [deg]
p_t^i	Instant position feedback of traverse axis [deg]

p_e^i	Instant position feedback of elevation axis [deg]
p^i	Instant position feedback of turret [deg]
p_t^{la}	Look ahead position of traverse axis [deg]
p_e^{la}	Look ahead position of elevation axis [deg]
p_c^{la}	Equivalent look ahead position of turret [deg]
ω_t	Unshaped angular speed reference of traverse axis [deg/s]
ω_e	Unshaped angular speed reference of elevation axis [deg/s]
ω_c	Equivalent unshaped speed reference of turret [deg/s]
ω_t'	Shaped speed reference of traverse axis [deg/s]
ω_e'	Shaped speed reference of elevation axis [deg/s]
ω_c'	Equivalent shaped speed reference of turret [deg/s]
$\omega_{c_{pre}}'$	Equivalent shaped speed reference of previous iteration [deg/s]
ω_a	Custom defined avoiding speed command for elevation [deg/s]
ω_t^{max}	Maximum allowable angular speed of traverse axis [deg/s]
ω_e^{max}	Maximum allowable angular speed of elevation axis [deg/s]
a_t	Acceleration reference for traverse axis [deg/s^2]
a_e	Acceleration reference for elevation axis [deg/s^2]
a_c	Equivalent acceleration reference of turret [deg/s^2]
a_t'	Applied acceleration reference for traverse axis [deg/s^2]
a_e'	Applied acceleration reference for elevation axis [deg/s^2]
a_c'	Applied equivalent acc. reference of turret [deg/s^2]
$a_{c_{pre}}'$	Applied equivalent acc. reference of previous iteration [deg/s^2]
a_t^{max}, d_t^{max}	Maximum angular acc./deceleration limit for traverse [deg/s^2]
a_e^{max}, d_e^{max}	Maximum angular acc./deceleration limit for elevation [deg/s^2]
j_t^{max}	Maximum allowable angular jerk of traverse axis [deg/s^3]

j_e^{max}	Maximum allowable angular jerk of elevation axis [deg/s^3]
δ	Safe distance for detection of collision between point clouds [mm]
δ_t	Threshold angle of virtual dynamic barrier for traverse [deg]
δ_e	Threshold angle of virtual dynamic barrier for elevation [deg]
ε	Small angle value for local minima (10 % of δ_e) [deg]
a	Rotational angle about X-axis [deg]
β	Rotational angle about Y-axis [deg]
γ	Rotational angle about Z-axis [deg]

CHAPTER 1

INTRODUCTION

1.1 Motivation

The robotic manipulation is an established technology that is widely used in the industry [1]. Besides, the topic of Industry 4.0 has become popular among many companies, research centers, and universities. The transition to industry 4.0, the rise of unmanned and smart factories, smart production, machine-to-machine and advanced manufacturing have greatly increased the need for co-working robots and robots with a certain degree of independence has become more attractive [1-6]. Also, robotic systems have become indispensable for the military as well. The need for remote controlled weapon systems and multiple missile / rocket launcher systems is increasing day by day. They use these systems to increase military capability and security of forces and allow personnel to concentrate on specific tasks which can only be fulfilled by manpower. With the development of robotic systems, most military systems have now become remotely controllable, and this has significantly reduced the need for military personnel [7]. As the number of remote-controlled systems on military vehicles increase, autonomous driving of the systems and the use of collision avoidance algorithms are inevitable.

Although there are no works directly focusing on the path planning and collision avoidance of turrets in the literature, algorithms for the braking and overtaking needs of similar robotic systems, Unmanned Aerial Vehicles and autonomous vehicles can be guiding. Within the scope of this thesis, our motivation is to overcome this deficiency. Motivated by the above problems, we focus on the new collision

avoidance algorithm named as End Damping Algorithm which provides systems to be driven both in speed and position control modes more efficiently and safely.

Figure 1.1. Jobaria multiple cradle launcher [8]

A multiple cradle launcher in Figure 1.1 can be an example for this study. 3 dependent turrets that have 2 DOFs in each one means that the whole system has 6 DOFs in total [8] with some constraints. It is impossible to rotate turrets independently because of the space restrictions. Examples of similar problems can be reproduced as in Figure 1.2 and Figure 1.3 is an infantry fighting vehicle with two turrets which have 2 DOFs each which means that the whole system has 4 DOFs in total [9].

Figure 1.2. Infantry fighting vehicle with multiple turrets [9]

Also, the configuration in Figure 1.3 can be another example where two robotic manipulators are working together in the same environment which creates collision problems [10]. Each manipulator in the Figure 1.3 has 6-DOFs and satisfies the Pieper criterion, that is, the last three consecutive axes wrist of the robot intersect at one point, and therefore it is possible to model the last 3 DOFs of the manipulator as a sphere that covers all the motion of last 3 links. So, the motions of two manipulators can be expressed as a 6-dimensional configuration space which helps to conduct the path planning without colliding.

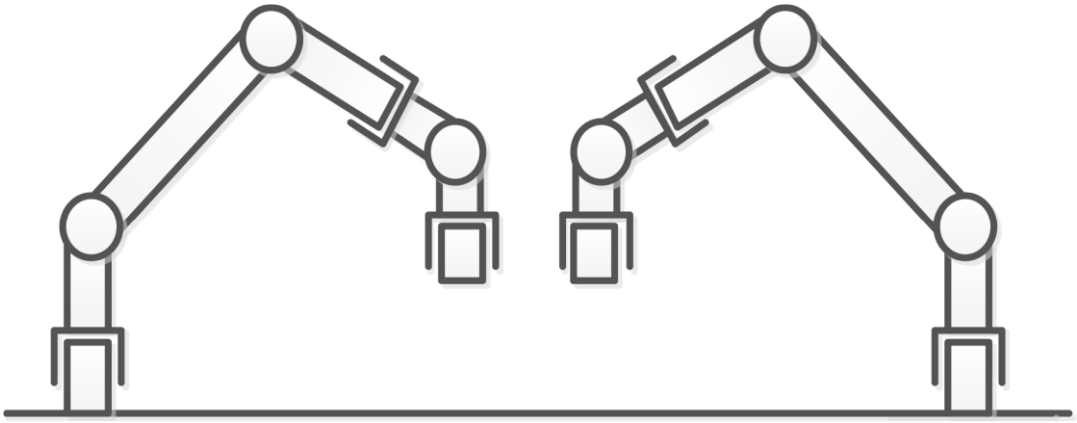

Figure 1.3. Model of two manipulators [10]

Path planning can be done both on-line and off-line to overcome these challenges and obtain multi-dimensional configuration space. These approaches can provide both benefits and drawbacks to the structures they are applied to. Since the online methods identify objects in the workspace at the same time, the configuration space is continually updated. This method benefits from continuous updating of the configuration space, but the electronic devices used for updating will make the system more costly. Off-line strategies are less expensive as long as the workspaces of the cooperating robots do not shift.

1.2 Literature Survey

Potential field algorithms can be used to explicitly formalize and solve motion planning problems in the 3-D workspace [13]. These workspace solutions, on the other hand, cannot easily accommodate robots with various geometrical and mechanical constraints. Path planning can be formalized and solved in a new space called configuration space [14-18] to solve these problems. In a 2-D workspace, a complex geometrically formed robot is mapped to a point robot; as a result, the robot's motion corresponds to a continuous curve in the high-dimensional C-Space, as shown in Figure 1.4.

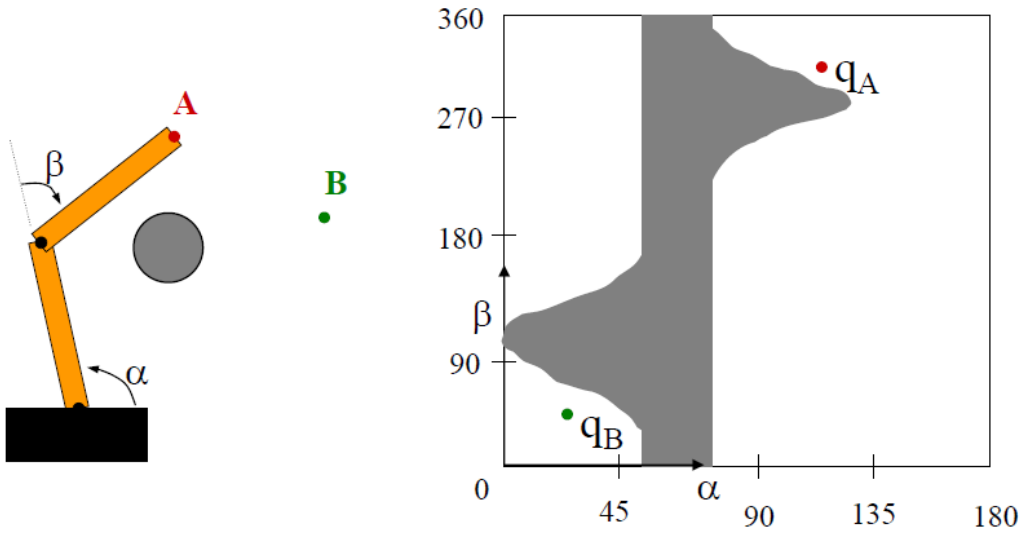

Figure 1.4. Workspace and C-Space for 2-D [11]

An example of 3-D workspace and C-space is given in Figure 1.5. In two stages, we can solve the issue of motion planning [11]. To move robots or manipulators to target positions without colliding, first acquire the configuration space and then optimize the C-space that has been discovered. The configuration space is generated by integrating all of the cooperative systems' possible motions. A defined safety factor may also be used to extend the mapping of obstacles in configuration space. The safety factor is calculated based on the minimum safe distance between the

cooperators. For the 2-D case, the "Minkowski" sum form [11, 12] can be used as shown in Figure 1.6.

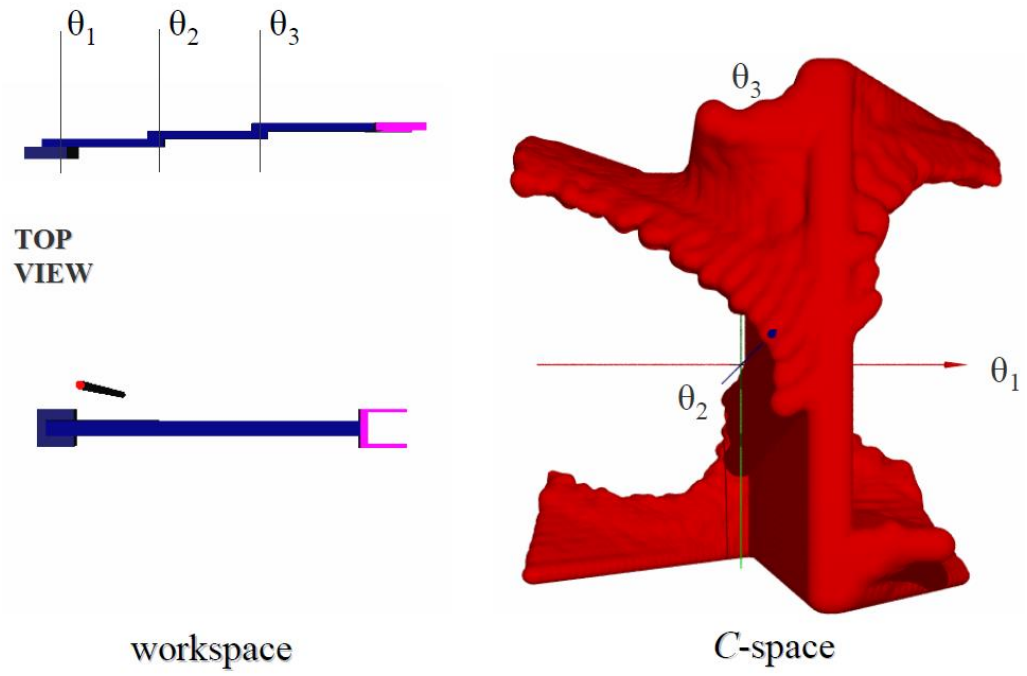

Figure 1.5. An example of 3-D workspace and C-space [11]

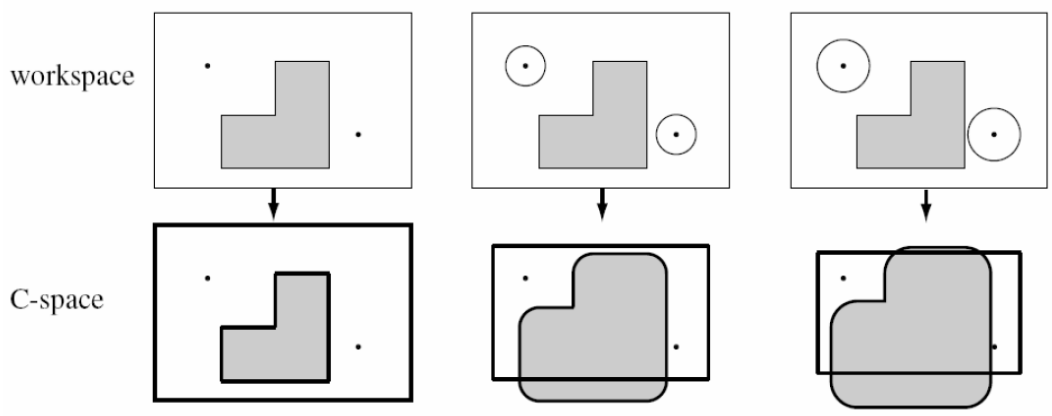

Figure 1.6. Minkowski sum method [11]

We would be interested in the configuration space of systems that do not alter regularly, in other words, off-line systems, in this study. The resulting configuration space should be modified if the workspace is altered.

Many studies on collision-free motion in multi-robot systems have been published in the literature. However, there are a few papers on collision-free motion preparation of dual-arm robots, which is a subject that is close to multiple-turret systems. There are some obvious similarities between them in terms of collision-free motion preparation. The collision-free motion planning for two R–P robots is analyzed using Freund and Hoyer's algorithm [19], which considers a hypothetical robot whose effector is permanently collided with that of the master robot. By introducing a safety factor for minimum robot clearance, a trajectory can be achieved without collision. In another research, a 2-D horizontally articulated dual-arm SCARA robot as shown in Figure 1.7 is studied [20]. The configuration space (C-Space) of the related robot (4-R) is derived using the reachable manifold and touch manifold principle, which is insufficient to handle complex geometric shapes; thus, robot arms are depicted using simplistic shapes such as rectangles. Furthermore, all revolute joints can only rotate in a certain range; none of them can rotate completely which is limiting the double-turret system's capability.

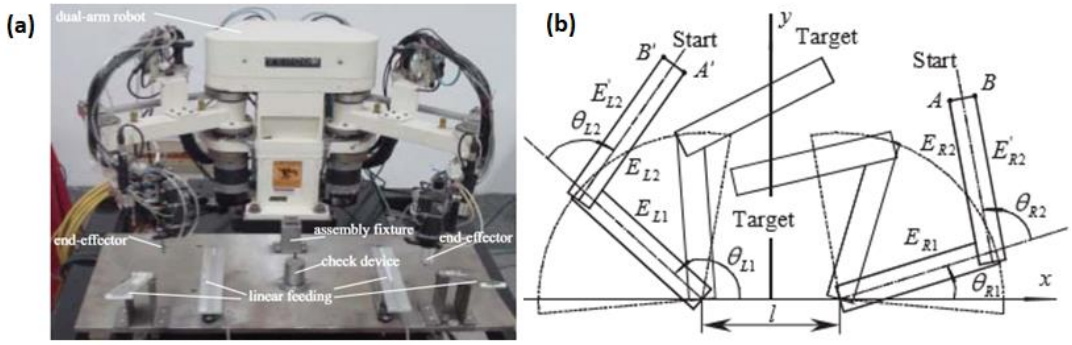

Figure 1.7. (a) dual-arm SCARA robot (b) a sketch of this robot [20]

The study of navigation can be divided into two categories: global route planning and local collision avoidance. Local collision avoidance takes a given waypoint assignment as a local aim to avoid obstacles, while global route planning algorithms construct a group of waypoints from a start location to a target position, navigating through obstacles in a working space and configuration area [21,22]. Since the 1970s, path planning has attracted a lot of attention [23]. For the last four decades,

several algorithms have been known for. Potential field method [24], heuristic search method [25], and graph strategy [26] are all used in these algorithms.

Global route planning algorithms are used for missile system launchers because it is more important to travel from one position to another than to follow a trajectory, while local collision avoidance algorithms are used for armored ground vehicle turrets. As a result, real-time collision avoidance algorithms are getting a lot of attention. Local approaches include the Velocity Obstacle, Set-Based Guidance, Vector Field Histogram, and Dynamic Window Approach. Borenstein and Koren proposed the Vector Field Histogram (VFH) [27] technique for collision avoidance for flexible robots in 1991. It was introduced by Khatib in 1985 [28] as a solution to the fundamental issues of the Potential Field strategies. The Dynamic Window Approach (DWA) was introduced by Fox, Burgard, and Thrun in 1997 as another technique for collision avoidance for flexible robots [29]. The Dynamic Window is a diminished speed space containing only the speeds reachable inside a short measure of time and that guarantees a way where the robot is capable to stop securely. An objective function is controlled by consolidating the speed of the robot, the advancement towards the objective and the separation between the robot and the hindrance. It brings about a harmony between keeping a rapid motion towards the objective and performing sly moves to dodge deterrents. The translational and rotational speeds are picked by maximizing the objective function. Since most robots have some sort of limitations on speed and acceleration, DWA is particularly decent since it handles this automatically [30]. Velocity Obstacle (VO) was first published by Fiorini and Shiller in 1998 [31]. VO is a movement arranging strategy for robots in extreme situations. It aims to find all speeds for which the robot will crash into a deterrent sooner or later. These speeds will make confined regions in the speed space molded like cones, one for each obstacle.

Potential filed method, reactive methods, deformable virtual zones, velocity potential field and acceleration velocity obstacle can be cited as examples of local collision avoidance algorithms where intervention to speed and even torque loops is inevitable for implementation [32-35]. These proposed collision avoidance

algorithms for autonomous vehicles [36-42] are not easy to apply directly to turrets with two different independent axes, as these algorithms perform their maneuvers by braking or steering [43-46].

1.3 Contribution of the Thesis

The main contributions in this thesis are summarized as follows.

(1) High-dimensional C-Space of any system can be obtained by using intersections of point clouds in space, independent of the shapes and sizes of the system components and without the need to surround these shapes with familiar shapes.

(2) An algorithm is proposed for the satisfaction of path planning in a 4-D C-Space.

(3) With the proposed algorithm, it is now possible to work on the different C-Spaces like circular and torus shaped C-Spaces in addition to rectangular shaped C-Space.

(4) Since the configuration spaces of two or more aligned multiple-turret systems can be obtained using the 4-D C-Spaces of adjacent turrets, more than two aligned turrets on the same platform can be driven as well.

(5) The user can drive the turret on any axis and at the desired speed without paying attention to the obstacles and specifying a target position.

(6) The obstacles are defined as Cartesian based rectangles in configuration space so that they can be expressed with fewer parameters.

(7) The system acts in compliance with all speed, acceleration and jerk limits.

(8) Even in situations such as avoiding obstacles, deceleration/acceleration, if there appear new commands from user which does not cause a collision, the algorithm starts to apply the new commands which is so critical for military systems to reduce response times.

(9) By only adding cascade position control loop, the departure from the starting point to the desired target point can be achieved by using same algorithm and during transition phase, the target point can be changed instantly.

(10) Since the algorithm does not intervene in the speed and torque loops in contrast to potential field-based methods, it can be added to ready-to-use systems by manipulating only the speed references.

1.4 Outline of the Thesis

The rest of the thesis is organized as follows.

In the second chapter, the subject of how to obtain the high dimensional configuration space is discussed and it has been shown that it can be obtained in two different methods. In order to compare these two methods, a double-turret system and its 4-D C-space are focused on. Then, the accuracy of the configuration space is proved using the obtained paths on the simulation model of the double-turret system.

In the third chapter, after obtaining of high-dimensional C-Space with point clouds, the approach of path planning on 4-D C-Space is proposed. How to handle C-Spaces of the systems with more than two turrets is described. The performance of the proposed algorithm is simulated on the triple-turret system and finally the results are summarized and discussed.

In the fourth chapter, for the satisfaction of collision avoidance requirement, an algorithm named as End Damping is proposed. The proposed algorithm itself and possible worst cases are examined. The algorithm is tested on both simulation and real-time test benchmarks in all possible scenarios including all worst cases. The simulation and experimental results are illustrated and the results are summarized and discussed.

CHAPTER 2

OBTAINING OF CONFIGURATION SPACE

The shift to Industry 4.0, as well as the rise of autonomous and smart factories, smart development, machine-to-machine and advanced manufacturing, has significantly increased the demand for co-working robots, making robots with a degree of autonomy more appealing [1-4]. However, as more complex and intelligent operations are developed, complex tasks in certain processes can no longer be completed effectively with just a single manipulator. At the same time, it is well understood that coordinating two manipulators increases the task's complexity while also improving its performance. When the two manipulators operate in the same workspace, however, it is possible for them to collide with each other and with the obstacles in the environment, so the coordinated activity of the two manipulators has become a hot topic for researchers [5].

The number of dimensions in the configuration space is determined by the number of degrees of freedom (DOFs) in the structure under consideration. There are numerous studies on route planning in configuration space in the literature. To check their route planning strategy, they often used a sample 2 or 3-dimensional configuration space [5-8]. Unlike those studies, however, this one aims to develop methods for obtaining high-dimensional configuration spaces for real-world structures such as robotic manipulators, turrets, and so on [9-10].

Jobaria, a multiple cradle launcher shown in Figure 1.1, can be used as an example in this analysis. With three dependent turrets, each with two degrees of freedom (DOFs), the system has a total of six degrees of freedom (DOFs) [9] with some constraints. Due to space constraints, it is difficult to rotate launchers independently.

An infantry combat vehicle with two turrets, each with two degrees of freedom, can also be used as an example for this study [10].

Also, the configurations in Figure 2.1 can be similar examples where two robotic manipulators or dual-arm robots are working together in the same environment which creates collision problems [5]. The motion of two dual-arm robots, each of which has 2 degrees of freedom in Figure 2.1 (a), can be defined as 4-dimensional C-Space. Each manipulator in the Figure 2.1 (b) has 6-DOFs and satisfies the Pieper criterion, that is, the three consecutive axes of the robot intersect at one point, and therefore it is possible to model the last 3 DOFs of the manipulator as a sphere that covers all the motion of last 3 links. So, the motions of two manipulators can be expressed as a 6-dimensional configuration space which helps to conduct the path planning without colliding. To handle these difficulties and obtain high-dimensional configuration space, there are on-line and off-line methods to conduct path planning. These methods can provide advantages and disadvantages to the systems they are used. Because the online methods recognize objects in the workspace simultaneously, it constantly updates the configuration space. The continuous updating of the configuration space provides advantages for this method, while the electronic devices used for updating can make the system more expensive. As long as the workspaces of the robots which are working together do not change, off-line methods are cheaper.

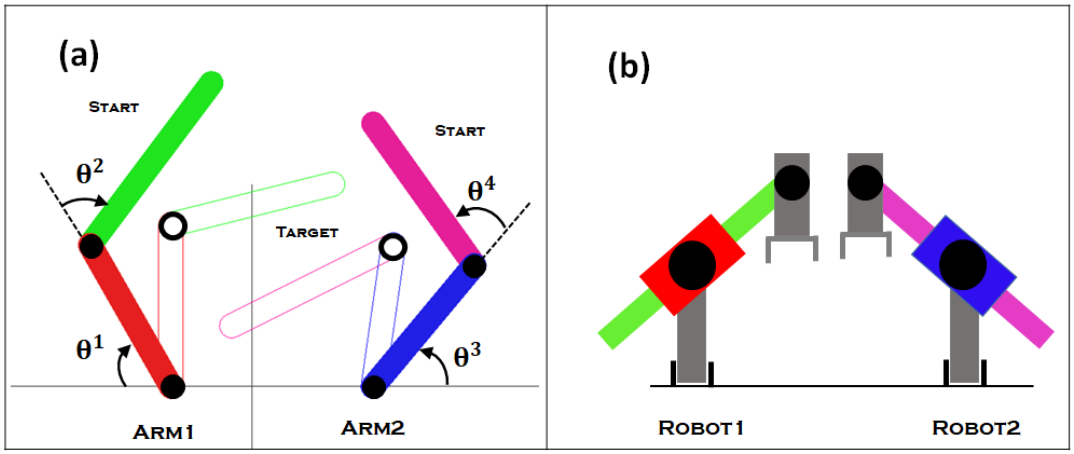

Figure 2.1. (a) collaborative dual-arm robots, (b) model of two manipulators

2.1 Configuration Space

The motion planning problems can be directly formalized and solved in the 3-D workspace, generally by the potential field algorithms [47]. However, it cannot be easily handled the robots with various geometrical and mechanical restrictions by these workspace solutions. To solve these difficulties, path planning may be formalized and solved in a new space which is called configuration space [12, 14-16]. The complex geometric shaped robot in a 3-D workspace is mapped to a point robot, therefore the motion of the robot corresponds to a continuous curve in the high-dimensional configuration space as given in Figure 2.2.

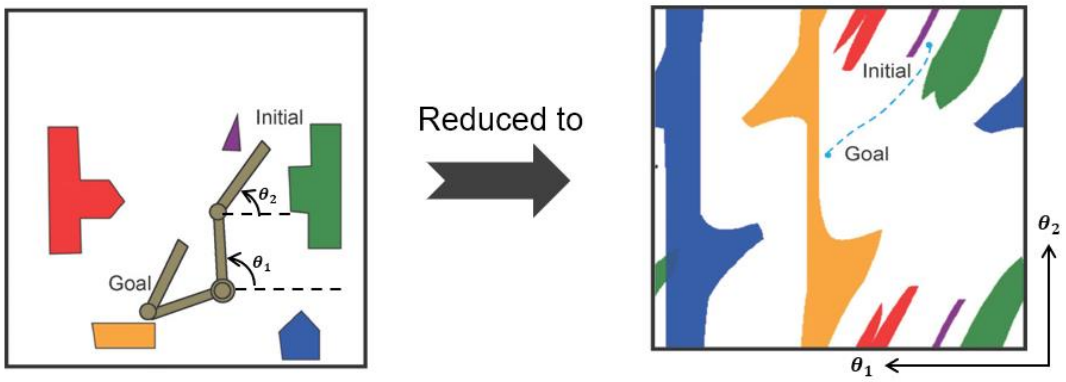

Figure 2.2. Workspace and C-Space representations [12]

We can solve the motion planning problem in two steps [12]. For robots or manipulators to be driven to desired locations without colliding, firstly it is required to obtain the configuration space and perform an optimization on the found C-Space. The configuration space is created by combining all the possible motions that cooperative systems can do. Besides, the mapping of the obstacles in configuration space may be expanded by a specified safety factor. The safety factor is determined according to the safe distance that must be between the co-operators. It can be done by "Minkowski" sum method for the 2-D cases [12].

In this thesis, we will be interested in the configuration space of the systems which does not change frequently, in other words, focused on the off-line systems. If the

workspace is changed, then the corresponding configuration space should be updated. During this study, we do not focus on path planning itself. Besides, there will be some path planning solutions to verify the obtained n-dimensional configuration space. There are two methods to obtain n-dimensional configuration spaces which will be input for n-dimensional path planning. One of them is to obtain n-dimensional configuration space for a system by using point clouds and the detection of the intersection of these clouds. The other is to handle 3-D models and degree of freedoms by using simulation softwares. During this study, the configuration space will be obtained by using these two methods.

2.1.1 First Method: Obtain by Point Clouds

To obtain configuration space by using point clouds, firstly all the 3-D shapes are converted into point clouds. For this purpose, any finite element meshing software can be used. After creating a mesh of 3-D shapes, the point cloud representation of the double-turret system can be seen as in Figure 2.3.

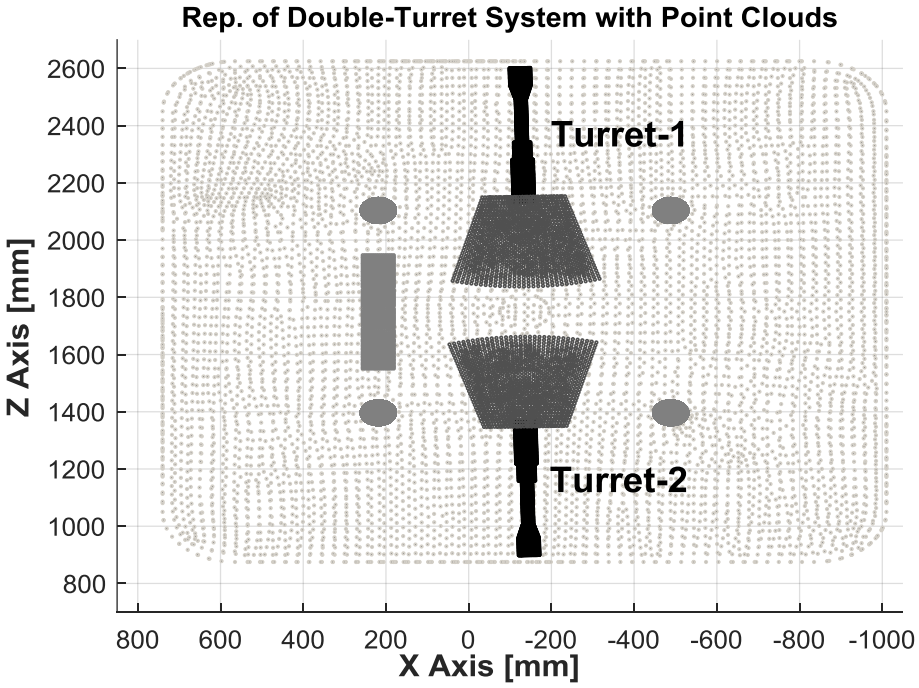

Figure 2.3. Point cloud representation of double-turret system

The point clouds can be classified and represented as moving point clouds (MPC) and fixed-point clouds (FPC) as seen in Eqs. (2.1) and (2.2).

$$MPC = \{MPC_1 \;\; MPC_2 \;\; MPC_3 \;\; \ldots..\} \tag{2.1}$$

$$FPC = \{FPC_1 \;\; FPC_2 \;\; FPC_3 \;\; \ldots..\} \tag{2.2}$$

Each element of MPC and FPC can be represented as in Eq. (2.3). The x, y and z values of point clouds are given with respect to earth fix frame in order to check collision easily.

$$MPC_i = \left[X_{in}\; Y_{in}\; Z_{in}\; \underline{1}\right] \qquad FPC_k = \left[X_{km}\; Y_{km}\; Z_{km}\; \underline{1}\right] \tag{2.3}$$

where

$$X_{in} = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \cdot \\ \cdot \\ x_{in} \end{bmatrix} \; Y_{in} = \begin{bmatrix} y_{i1} \\ y_{i2} \\ \cdot \\ \cdot \\ y_{in} \end{bmatrix} \; Z_{in} = \begin{bmatrix} z_{i1} \\ z_{i2} \\ \cdot \\ \cdot \\ z_{in} \end{bmatrix}$$
$$X_{km} = \begin{bmatrix} x_{k1} \\ x_{k2} \\ \cdot \\ \cdot \\ x_{km} \end{bmatrix} \; Y_{km} = \begin{bmatrix} y_{k1} \\ y_{k2} \\ \cdot \\ \cdot \\ y_{km} \end{bmatrix} \; Z_{km} = \begin{bmatrix} z_{k1} \\ z_{k2} \\ \cdot \\ \cdot \\ z_{km} \end{bmatrix} \tag{2.4}$$

For a given point cloud PC_H (n×4) in Cartesian coordinates, the translational and rotational operations can be represented as below.

$$T_{V^T} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

$$(PC_H)^T_{translate} = T_{V^T} \cdot PC_H^T \tag{2.6}$$

where $V = [t_x \quad t_y \quad t_z]$. First, the Cartesian based rotation vectors are defined in three-axes as in Eq. (2.7).

$$V_x = [0 \quad r_y \quad r_z], \qquad V_y = [r_x \quad 0 \quad r_z], \qquad V_z = [r_x \quad r_y \quad 0] \tag{2.7}$$

Rotations are defined by 4×4 transformation matrices. Rotation about X-axis by an angle of α, rotation about Y axis by an angle of β, and rotation about Z by an angle of γ are defined, respectively, as

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\alpha & sin\alpha & 0 \\ 0 & -sin\alpha & cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_y(\beta) = \begin{bmatrix} cos\beta & 0 & -sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ sin\beta & 0 & cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

$$R_z(\gamma) = \begin{bmatrix} cos\gamma & -sin\gamma & 0 & 0 \\ sin\gamma & cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

All point clouds are created on Cartesian based as mentioned and therefore, rotated point clouds about Cartesian axes can be obtained as below.

$$(PC_H)_x^T = T_{V_x^T} \cdot R_x(\alpha) \cdot T_{-V_x^T} \cdot PC_H^T \tag{2.9}$$

$$(PC_H)_y^T = T_{V_y^T} \cdot R_y(\beta) \cdot T_{-V_y^T} \cdot PC_H^T \tag{2.10}$$

$$(PC_H)_z^T = T_{V_z^T} \cdot R_z(\gamma) \cdot T_{-V_z^T} \cdot PC_H^T \tag{2.11}$$

The hull of the turret in Figure 2.3 can only rotate around y axis. Therefore, the point cloud of hull can be rotated as in Eq. (2.10). However, the barrel of turret is mounted directly to hull and therefore the point cloud of barrel has to be rotated first around x axis and then around y axis with hull. Hence, the rotated point cloud of barrel can be obtained as in Eq.(2.12).

$$PC_B = T_{V_y^T} \cdot R_y(\beta) \cdot T_{-V_y^T} \cdot T_{V_x^T} \cdot R_x(\alpha) \cdot T_{-V_x^T} \cdot PC_1^T \tag{2.12}$$

To find the distances between any point in MPC_i and any point in FPC_j, the matrix operations given below are used.

$$X = X_{in} - X_{km}^T \quad Y = Y_{in} - Y_{km}^T \quad Z = Z_{in} - Z_{km}^T \tag{2.13}$$

X, Y and Z matrices are as follows.

$$X = \begin{bmatrix} x_{i1} - x_{k1} & x_{i1} - x_{k2} & \cdot & \cdot & x_{i1} - x_{km} \\ x_{i2} - x_{k1} & x_{i2} - x_{k2} & \cdot & \cdot & x_{i2} - x_{km} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ x_{in} - x_{k1} & x_{in} - x_{k2} & \cdot & \cdot & x_{in} - x_{km} \end{bmatrix}$$

$$Y = \begin{bmatrix} y_{i1} - y_{k1} & y_{i1} - y_{k2} & \cdot & \cdot & y_{i1} - y_{km} \\ y_{i2} - y_{k1} & y_{i2} - y_{k2} & \cdot & \cdot & y_{i2} - y_{km} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ y_{in} - y_{k1} & y_{in} - y_{k2} & \cdot & \cdot & y_{in} - y_{km} \end{bmatrix} \tag{2.14}$$

$$Z = \begin{bmatrix} z_{i1} - z_{k1} & z_{i1} - z_{k2} & \cdot & \cdot & z_{i1} - z_{km} \\ z_{i2} - z_{k1} & z_{i2} - z_{k2} & \cdot & \cdot & z_{i2} - z_{km} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ z_{in} - z_{k1} & z_{in} - z_{k2} & \cdot & \cdot & z_{in} - z_{km} \end{bmatrix}$$

Finally, distance matrix between two point clouds are found as in Eq. (2.15).

$$D = \sqrt{X^2 + Y^2 + Z^2} \tag{2.15}$$

The condition to check collision between point clouds is given in Eq. (2.16) where δ is the safe distance defined in the definition of the collision. The value of δ may not be less than maximum mesh distance.

$$Collision = \begin{cases} 1, if & \min(D) \leq \delta \\ 0, if & \min(D) > \delta \end{cases} \tag{2.16}$$

As mentioned before, the 3-D point clouds are created from 3-D shapes. It is assumed that the length of any mesh in 3-D shape is approximately equal and named as d_m. The worst case of placing two point clouds and x-y and x-z views of the worst-case are given in Figure 2.4 which will help to find the minimum required safe distance, δ.

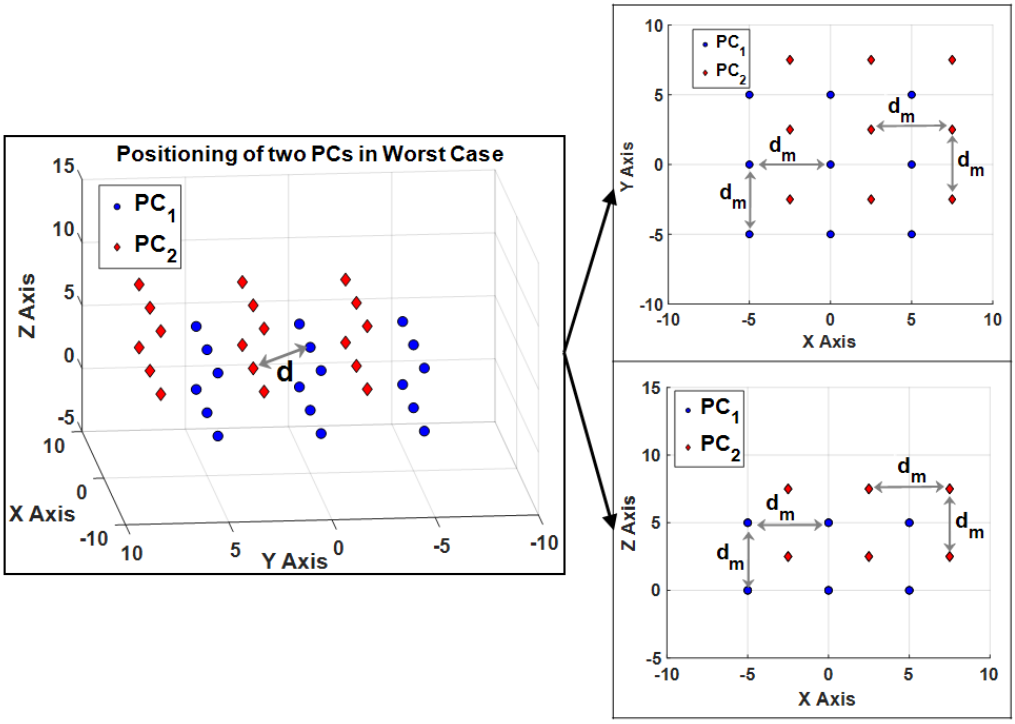

Figure 2.4. Positioning of two PCs in worst case

According to these figures, the safe distance should satisfy the criterion in Eq. (2.17) which is necessary and sufficient.

$$\delta \geq d = \frac{\sqrt{3}}{2} \; d_m \tag{2.17}$$

The overall algorithm for obtaining the high-dimensional configuration space can be summarized as in Algorithm-2.1. There is given the information about a 3-D environment such as moving point clouds (MPC), fixed-point clouds (FPC), safe distance (δ), the number of variables (n), initial values of n variables (θ_{in}), final values of n variables (θ_{end}) and step angles of n variables ($\delta\theta$). There is n number of nested loops in the algorithm where each of them belongs to a variable, in order to be an example, during this study the number of variables (n) is kept as 4. As a result of the algorithm, n-dimensional configuration space which is named ConfMap is obtained. Lines between 1-4 and 13-16 vary according to number of variables. The location of moving point clouds (MPC) is updated according to instantaneous variable values in Line 5. In Line 6-7, it is checked whether there is collision between MPC and FPC, and in Line 8-9 between MPC and MPC by avoiding matching of same point clouds according to the safe distance (δ).

Algorithm-2.1: ObtainConfSpace

In: $MPC, FPC, \theta_{in}, \delta\theta, \theta_{end}, \delta$

Out: $CSpace$

1 **for** $\theta^1 = \theta^1_{\text{in}} : \delta\theta^1 : \theta^1_{\text{end}}$ $(s_1 + +)$

2 **for** $\theta^2 = \theta^2_{\text{in}} : \delta\theta^2 : \theta^2_{\text{end}}$ $(s_2 + +)$

3 **for** $\theta^3 = \theta^3_{\text{in}} : \delta\theta^3 : \theta^3_{\text{end}}$ $(s_3 + +)$

4 **for** $\theta^4 = \theta^4_{\text{in}} : \delta\theta^4 : \theta^4_{\text{end}}$ $(s_4 + +)$

5 $update$ ***MPC*** $using\ current\ \theta^1, \theta^2, \theta^3\ and\ \theta^4$

6 **if** ***CheckCollision*** $(MPC, FPC, \delta) = 1\ (true)$

7 $CSpace\ (s_1, s_2, s_3, s_4) = 1$

8 **elseif** ***CheckCollision*** $(MPC, MPC, \delta) = 1\ (true)$
$(matching\ same\ p.clouds\ is\ avoided)$

9 $CSpace\ (s_1, s_2, s_3, s_4) = 1$

10 **else**

11 $CSpace\ (s_1, s_2, s_3, s_4) = 0$

12 **end if**

13 **end for**

14 **end for**

15 **end for**

16 **end for**

17 **return** $CSpace$

Also, the Algorithm-2.1 calls Algorithm-2.2 named CheckCollision which is used to detect any collision between point cloud sets to fill configuration space map for corresponding variables. The Algorithm-2.2 expects the MPC, FPC and δ as inputs and also needs three different functions which are named CheckLap, RemovePoints and FindDistance. In Line 3, it is checked whether there is an intersection between bounding volumes (BV) of two point clouds.

Algorithm 2.2: CheckCollision

In: MPC, FPC, δ

Out: $Collision$

1 **for** I ← number of P. clouds in MPC (I + +)

2 **for** k ← number of P. clouds in FPC (k + +)

3 $[Lap, IB] = \boldsymbol{CheckLap}\ (MPC_i, FPC_k)$

4 **if** $Lap = 1\ (true) \rightarrow P.clouds\ overlapped$

5 $MPC_i' = \boldsymbol{RemovePoints}\ (MPC_i, IB)$

6 $FPC_k' = \boldsymbol{RemovePoints}\ (FPC_k, IB)$

7 $d = \boldsymbol{FindDistance}\ (MPC_i', FPC_k')$

8 **if** $\min(d) \leq \delta$

9 $Collision(i, k) = 1$

10 **return** $\max(Collision)$

11 **else**

12 $Collision(i, k) = 0$

13 **end if**

14 **else**

15 $Collision(i, k) = 0$

16 **end if**

17 **end for**

18 **end for**

19 **return** max($Collision$)

If there is no intersection, then it can be said that there is no collision between the two point clouds. In the case of an intersection, the point clouds are updated by getting rid of the points outside the intersection box and a distance matrix is obtained between them to see if the minimum distance is smaller than the safe distance (δ).

To effectively check collision between point clouds, it is advisable to approximate objects with bounding volumes (BV) [48, 49]. Various bounding volume types which are widely used in the literature are presented in Figure 2.5.

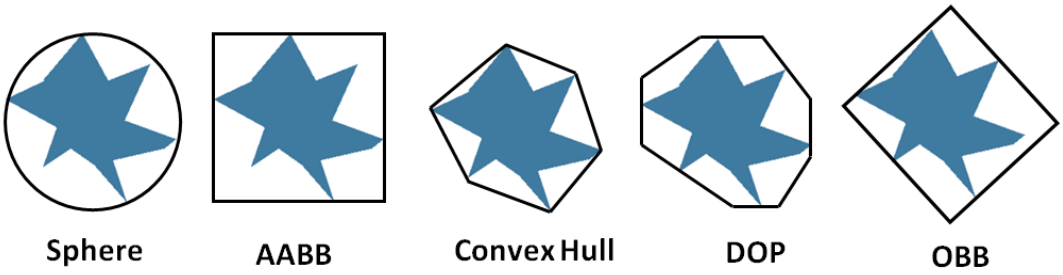

Figure 2.5. The types of bounding volumes

The selection of a bounding volume type depends on the usage. To easily select correct bounding volume, the type of objects should be known beforehand. If no information is available for the object size or shape, the more general shape is always better. However, the more general bounding volume adds extra complexity and hence heavier computationally [48-53]. To overcome this difficulty and uncertainty, the sphere or axis-aligned bounding boxes (AABB) are in options. In this study, AABB is used for that reason. The following function checks whether the AABBs of two point clouds (PC1 and PC2) are intersected. If there exists an intersection between AABBs, the function gives information about the intersection box (IB),

otherwise the overall algorithm returns to no-collision-detected state. Two point clouds and their AABBs are given in Figure 2.6. As seen in this figure, there is an intersection between these two AABBs. Also, the intersection box (IB) is given in Figure 2.7.

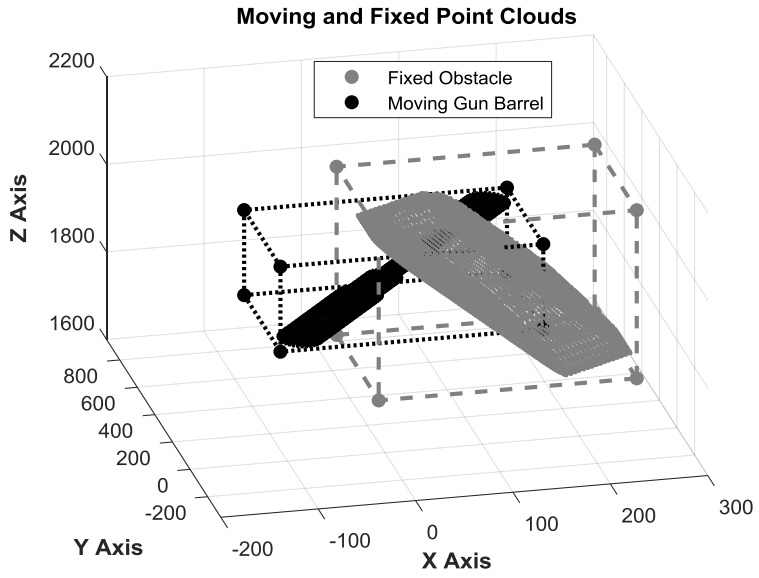

Figure 2.6. Two point clouds and their AABBs in 3-D Space

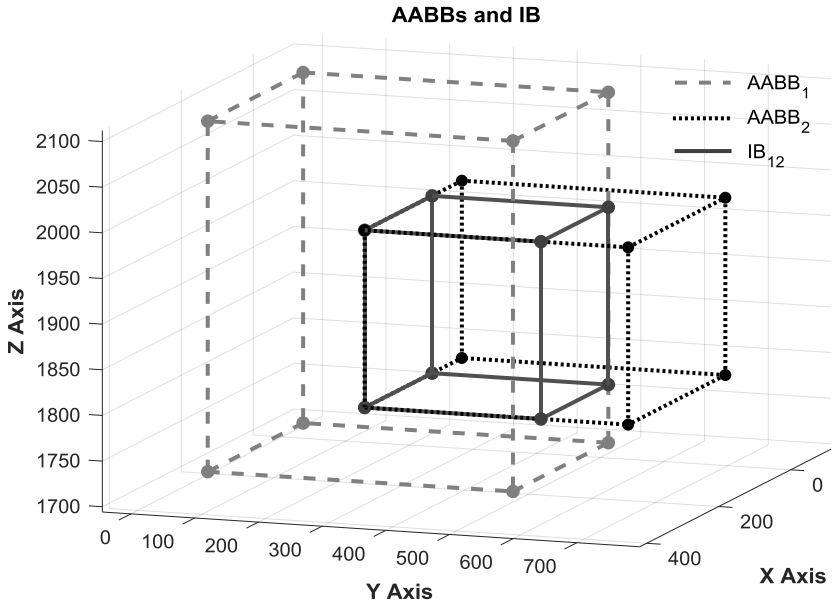

Figure 2.7. AABBs of two point clouds and their intersection box (IB)

The following Function-2.1 named CheckLap allows to see if the AABBs of the two point cloud intersect, and also to obtain intersection box (IB) information in case of intersection.

Function-2.1: CheckLap

In: PC_1, PC_2

Out: $[Lap, IB]$

for $i = [1, 2, 3]$(x: 1, y: 2, z: 3)
  if $\min\{PC_1(i^{th}\ column)\} \geq \min\{PC_2(i^{th}\ column)\}$
    $min^i = \min\{PC_1(i^{th}\ column)\}$
  else
    $min^i = \min\{PC_2(i^{th}\ column)\}$
  end if
  if $\max\{PC_1(i^{th}\ column)\} \leq \max\{PC_2(i^{th}\ column)\}$
    $max^i = \max\{PC_1(i^{th}\ column)\}$
  else
    $max^i = \max\{PC_2(i^{th}\ column)\}$
  end if
  $IB(i,1) = min^i \ , \quad IB(i,2) = max^i$
  if $min^i \leq max^i$
    $Lap^i = 1$
  else
    $Lap^i = 0$
  end if
end for
$Lap = Lap^1 \cdot Lap^2 \cdot Lap^3$
return $[Lap, IB]$

After determining the intersection box (IB) and providing point clouds and information about IB to the Function-2.2 named RemovePoints, now it is time to get

rid of excess points that do not belong to IB. By using this function, the point clouds are modified and the points outside the IB are removed as seen in Figure 2.8.

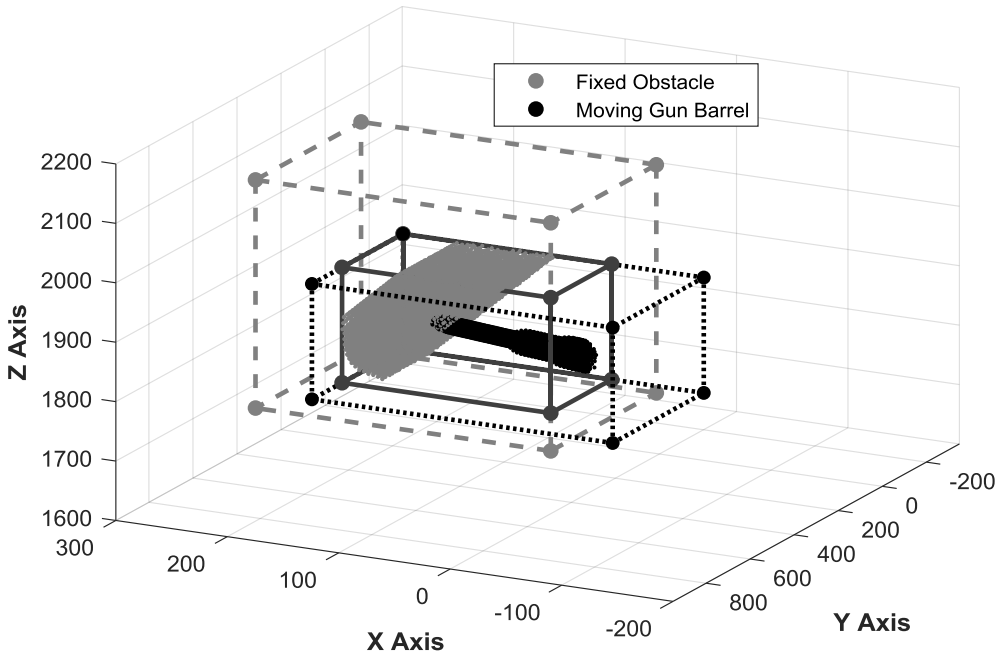

Figure 2.8. Modified point clouds after removing points outside intersection box (IB)

Function 2.2: RemovePoints

In: PC, IB

Out: PC'

if $\{x\ of\ \forall\ points\ \in\ PC > IB(1,2)\ \ \boldsymbol{OR} < IB(1,1)\}$

$\boldsymbol{OR}\ \{y\ of\ \forall\ points\ \in\ PC > IB(2,2)\ \ \boldsymbol{OR} < IB(2,1)\}$

$\boldsymbol{OR}\ \{z\ of\ \forall\ points\ \in\ PC > IB(3,2)\ \ \boldsymbol{OR} < IB(3,1)\}$

$\rightarrow remove\ \forall\ points\ of\ PC\ \notin\ IB$

$PC' = \forall\ points\ of\ PC\ \in\ IB$

else

$PC' = PC$

end if

return PC'

Firstly, the number of points is decreased by deleting the points outside the intersection box (IB) and then the modified MPC and FPC are created which only belong to the IB. Now, in order to decide whether there is a collision between point clouds, the distances between any combination of the points in MPC and FPC are measured. The following Function-2.3 named FindDistance measures g*h distances and gives g×h sized matrix where the PCs have g and h number of points, respectively.

Function -2.3: FindDistance

In: PC_1, PC_2

Out: d

1 $x_1 = PC_1(1^{th}\ column) \rightarrow (m \times 1)$

2 $y_1 = PC_1(2^{nd}\ column) \rightarrow (m \times 1)$

3 $z_1 = PC_1(3^{rd}\ column) \rightarrow (m \times 1)$

4 $x_2 = PC_2(1^{th}\ column) \rightarrow (n \times 1)$

5 $y_2 = PC_2(2^{nd}\ column) \rightarrow (n \times 1)$

6 $z_2 = PC_2(3^{rd}\ column) \rightarrow (n \times 1)$

7 $X = x_1 - x_2^T \rightarrow (m \times n)$

8 $Y = y_1 - y_2^T \rightarrow (m \times n)$

9 $Z = z_1 - z_2^T \rightarrow (m \times n)$

10 $d = \sqrt{X^2 + Y^2 + Z^2} \rightarrow (m \times n)$

11 **return** d

After obtaining the distance matrix, now it is easy to decide if there is a collision at that moment. If we sort and draw the elements of the distance matrix, the distances between points can be obtained as in Figure 2.9. If there is any distance lower than safe distance (δ), then we can decide that there is a collision for corresponding variables.

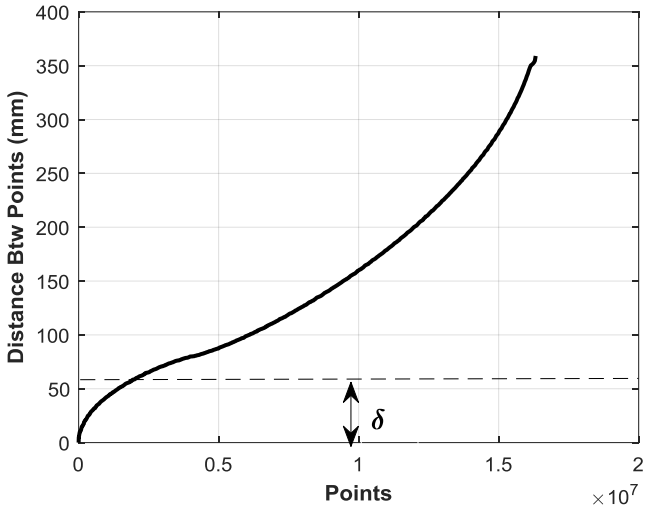

Figure 2.9. Sorting and drawing distance matrix to check collision

2.1.2 Second Method: Obtain by a Simulation Software

In the first method, obtaining the configuration space by detection of intersection of point clouds is explained in detail. In this method, the only collision detection algorithm is changed with the clearance function of the simulation software. Clearance function measures the minimum distance between the surfaces of bodies and only focuses the surfaces of bodies so that the bodies are assumed to be hollow. During any contact between the bodies, the clearance returns to zero which shows that there is a collision. We can summarize the steps of this method as shown in Figure 2.10.

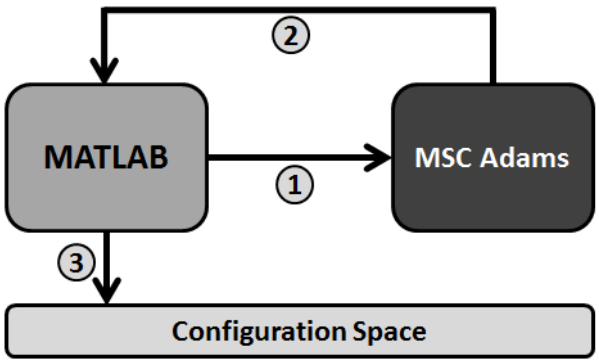

Figure 2.10. The sequence of the second method

1. Create position profiles for variables by converting the nested for-loops to time-domain. As an example, the position profiles for four variables in time-domain are illustrated as in Figure 2.11.

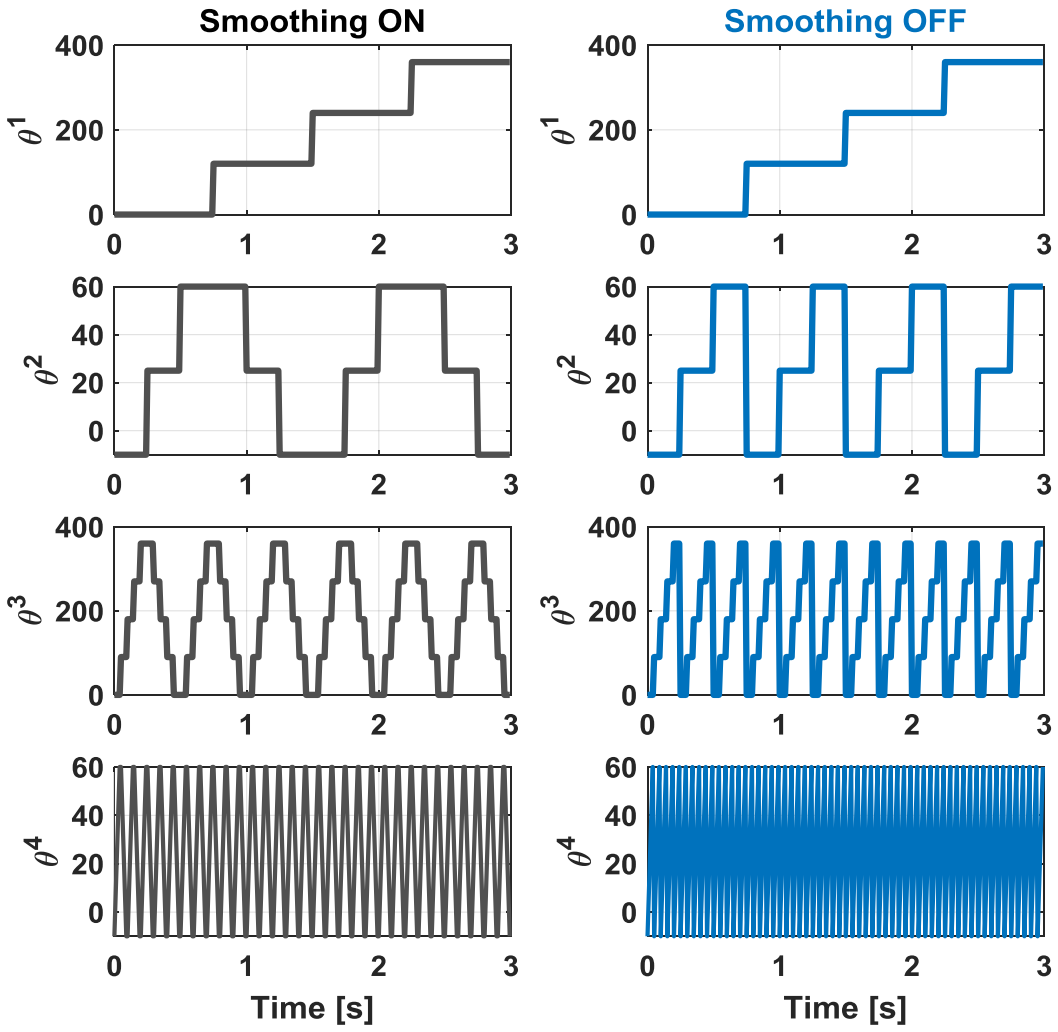

Figure 2.11. Converting nested for-loops into time-domain position profiles for variables

On the right side of the Figure 2.11, the position profiles are directly mapped from 4 nested for-loops with sharp edges. The hard transition from the maximum to minimum value causes some problems during solving kinematic equations in simulation software. Therefore, by doing a smoothing operation the position profiles

are updated as shown in the left of the Figure 2.11 and so that the possible errors in simulation are prevented.

2. Create the exact simulation model of the system in MSC Adams®. For instance, the simulation model of the 4-DOF double-turret system is given in Figure 2.12.

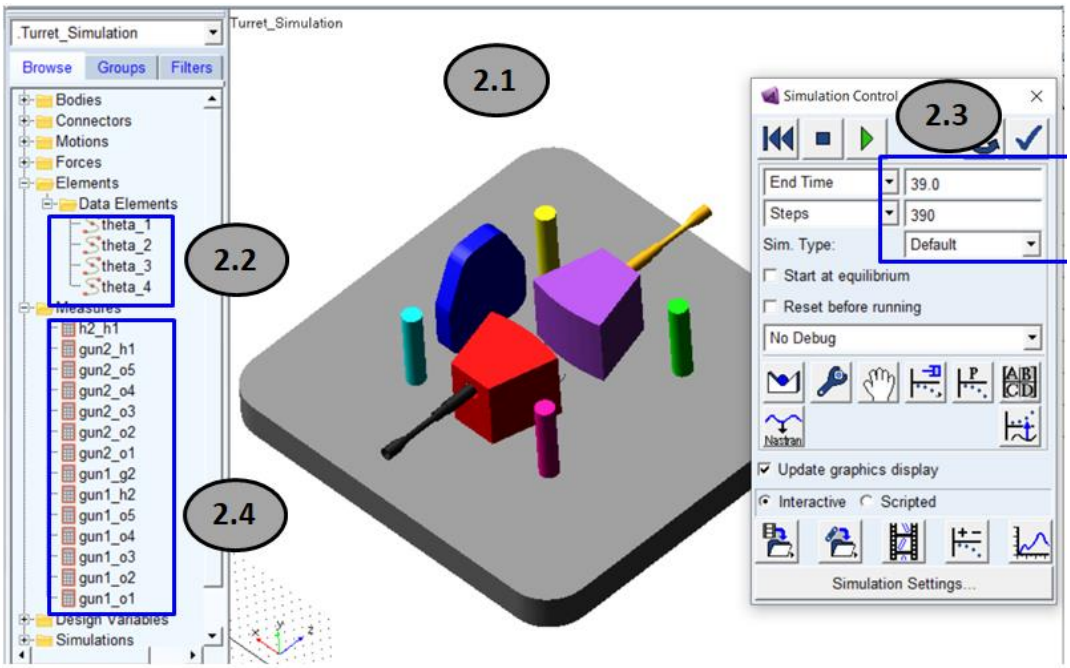

Figure 2.12. The Simulation model of a sample system

All joints, motions and clearances between stationary (obst-1 to 5) and moving bodies (hull and gun barrel) have to be defined. The position profiles in time-domain which are explained in the first step have to be imported and assigned to motions on the joints (θ_1, θ_2). According to steps and step time of position profiles, simulation is conducted.

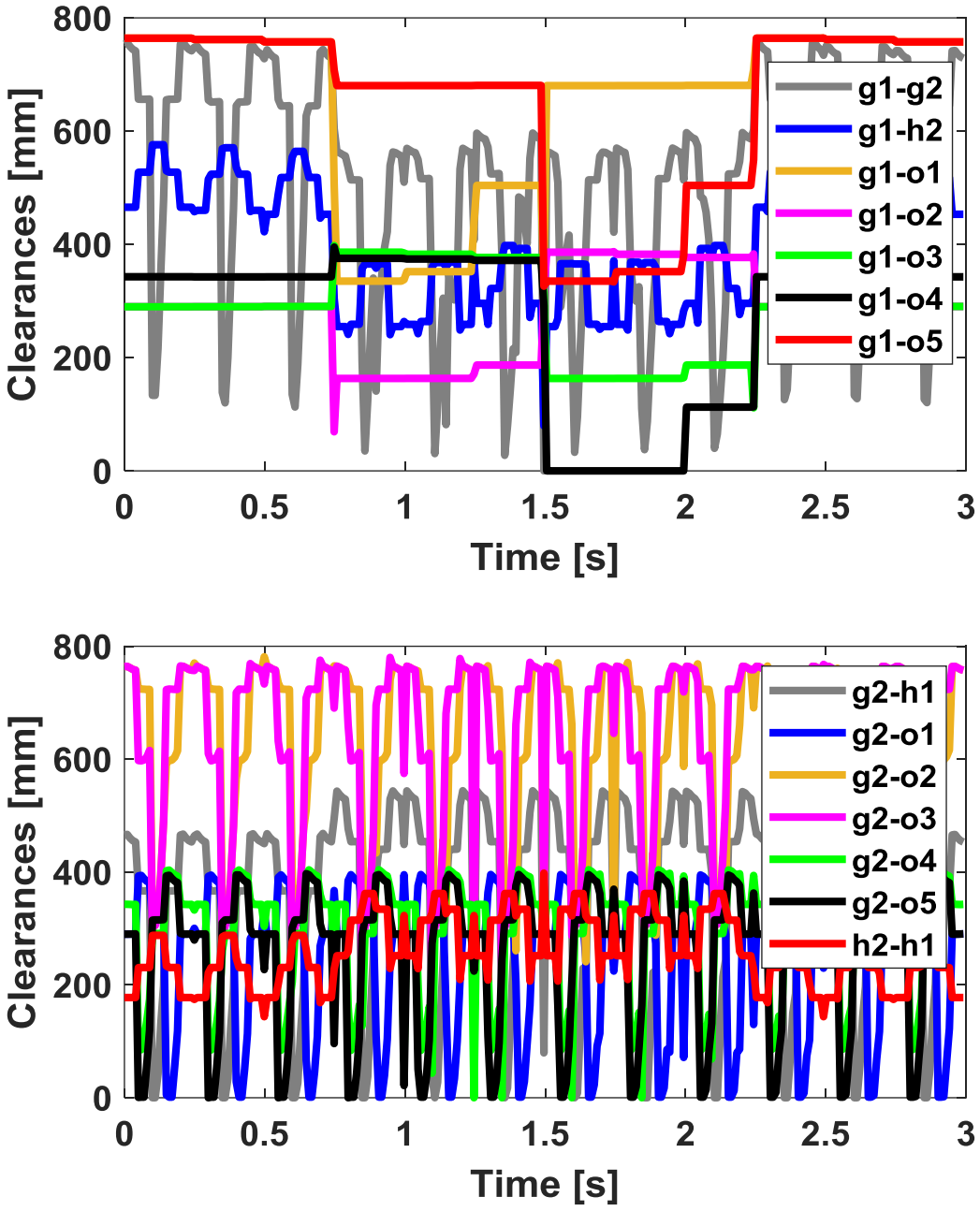

Figure 2.13. The change of minimum clearances as a result of sample run

3. After completing the analysis, the change of clearances again in time-domain are obtained as in Table 2.1 and sent to other simulation software (MATLAB®) in order to obtain configuration space by processing with position profiles inputs.

Table 2.1 Results of Simulation

$\mathbf{t\ [s]}$	$\boldsymbol{\theta^1}[\ ^\circ]$	$\boldsymbol{\theta^2}[\ ^\circ]$	$\boldsymbol{\theta^3}[\ ^\circ]$	$\boldsymbol{\theta^4}[\ ^\circ]$	$\mathbf{Collision(\delta)}$
0	0	-10	0	0	0
t_s	0	-10	0	20	0
$2t_s$	0	-10	0	40	0
….	….	….	….	….	….
t_{end}	358	0	358	0	0

2.2 Case Study

To compare 2 methods of obtaining configuration space, a double-turret system is considered as an example. The simulation and point cloud models of the double-turret system are given in Figure 2.14 and Figure 2.3, respectively.

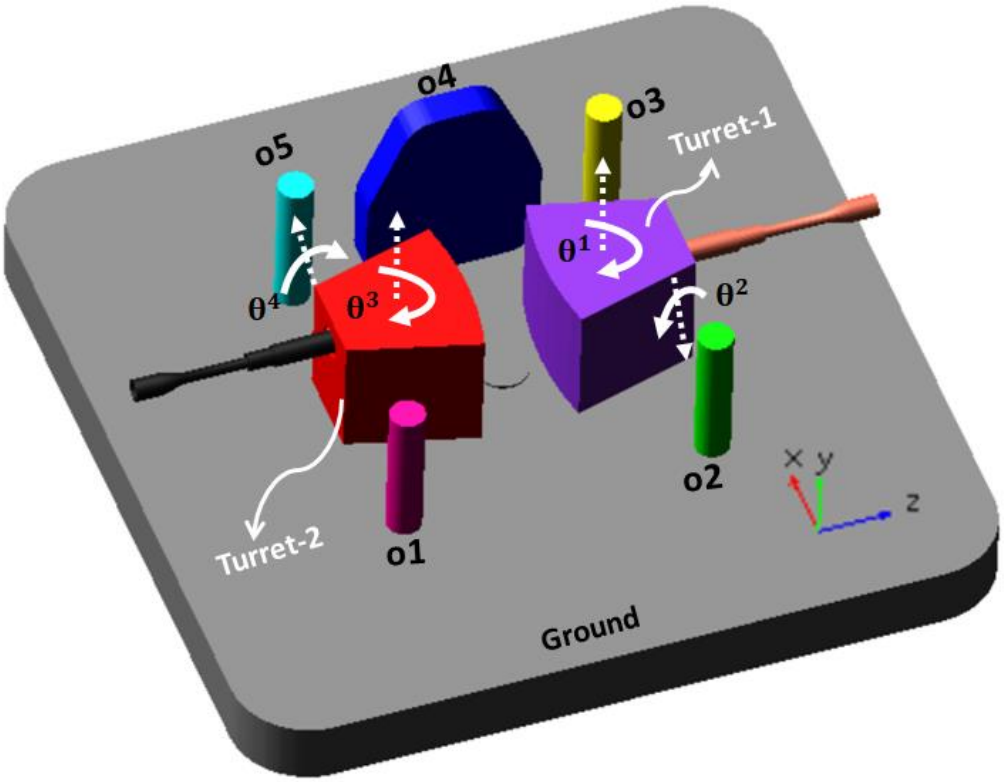

Figure 2.14. The Simulation model of double-turret system with stationary obstacles

The turret has two DOFs, one for traverse and the other for elevation. In this example, we have 4 DOFs because of two turrets in the same environment. Also, there are 5 stationary obstacles around these two turrets. Finally, the configuration space up to 4 dimensions can be obtained. Point numbers in the point clouds of all the elements used in the double-turret system are as in Table 2.2. Within scope of this study, the capital or lowercase letters g, o and h represent guns, obstacles and hulls respectively. Also, the ground was not included in the calculations because it is known that no part interacts with the ground at maximum and minimum operating limits for faster results.

Table 2.2 Number of points

Parts	**Number of Points (for each)**
O1, O2, O3, O5	4065
O4	18831
H1 and H2	5614
G1 and G2	3624

The 4-D configuration space of the example is obtained. In this 4-D C-Space, the elevation axes (θ^2, θ^4) start from - 10°, end at 60° with a step size of 2°. The traverse axes (θ^1, θ^3) start from 0°, end at 358° with a step size of 2°. In the method of path planning on the C-Space, the grid number of C-Space has great effect on the algorithm itself. If the grid is too great, the precision of planning will decrease. If the grid is small, the calculation payload will increase. A reasonable grid decomposition should be based on some optimum criterion [54]. The 4-D C-Space can be represented by a certain number of 3-D configuration spaces. For this example, it can be obtained with 36 different 3-D C-Spaces for each value of elevation axis of turret-2 (θ^4) in the example of double-turret system. The 3-D representations of this 4-D C-Space are given for eight different θ^4 angles in Figure 2.15 and Figure 2.16.

As can be seen from the figures, as the angle of θ^4 increases, the volume of the disabled area in the C-Space decreases.

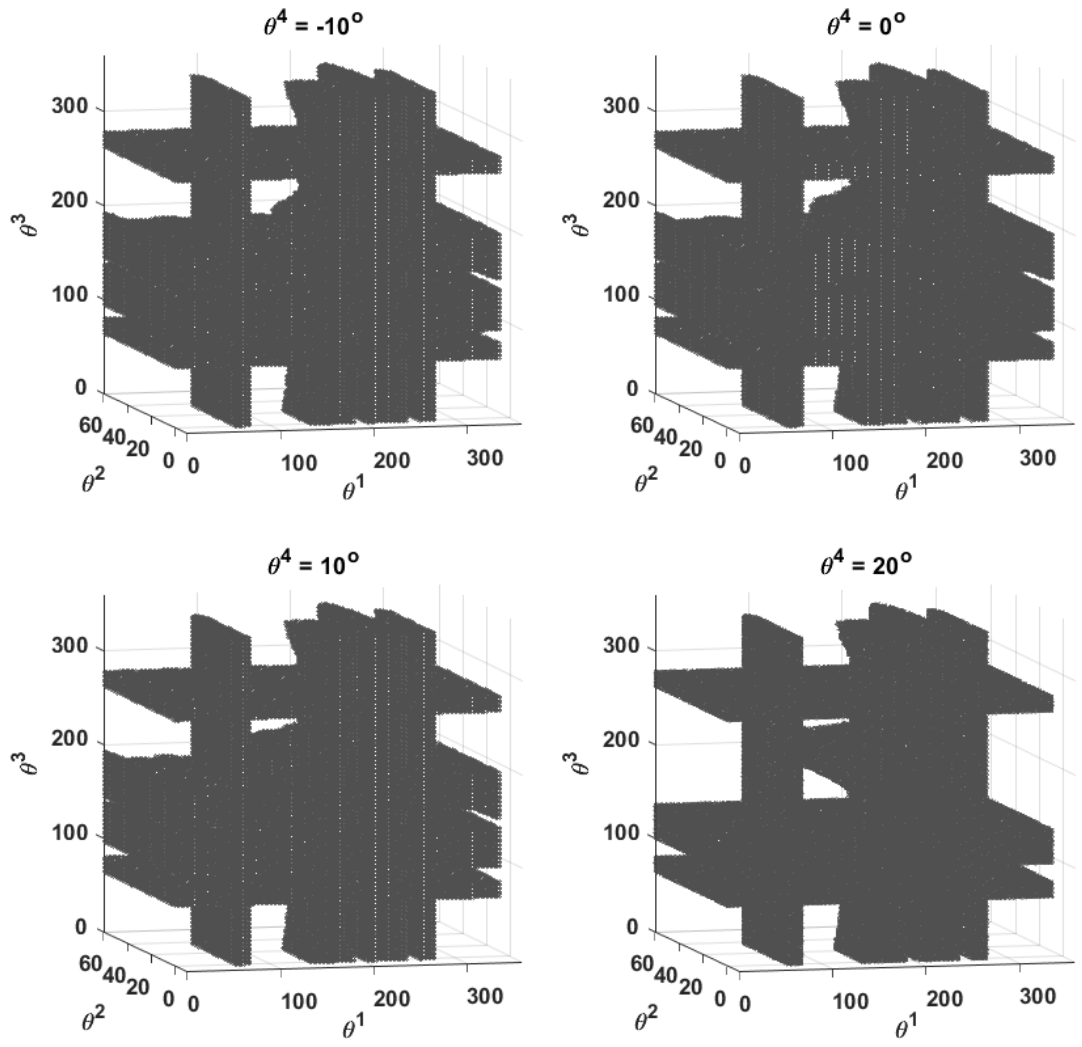

Figure 2.15. 3-D C-Space of double-turret system (for θ^4 equals -10°, 0°, 10° and 20°)

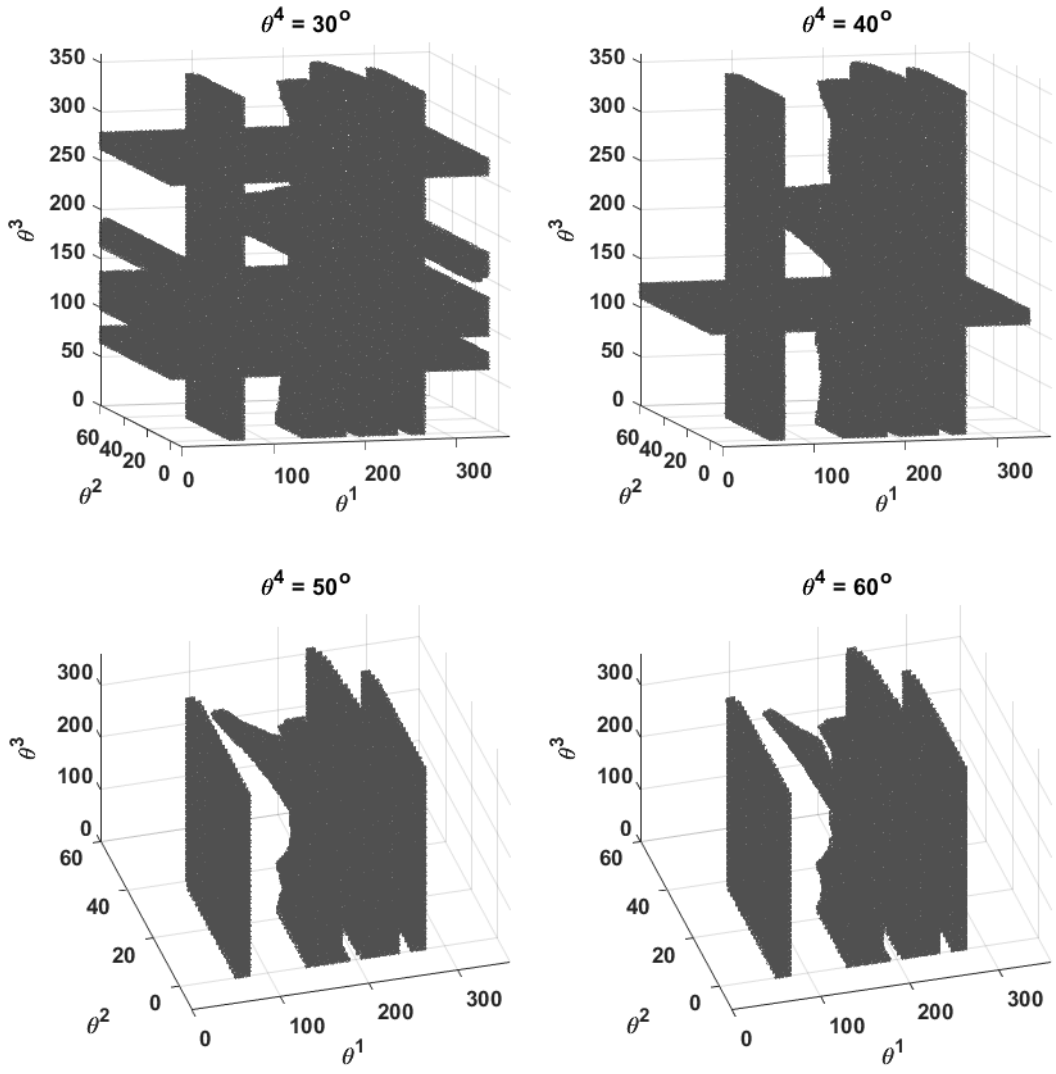

Figure 2.16. 3-D C-Space of double-turret system (for θ^4 equals 30°, 40°, 50° and 60°)

The obtained 3-D configuration spaces of two methods (by point cloud, by MSC Adams® software) are almost the same. The difference between them is about 1% which comes from the method difference. In the first method, the clearances between bodies are measured from point to point. However, in the second method, they are measured from the surface to surface. In this example, the distance between points in the point clouds is about 5 mm which creates a 1% difference than the mentioned

and also the safe distance is 5 mm. The difference between these two methods depends on the distance between points. Difference between methods decreases when the distance between points decreases. Also, the first method is only created to obtain configuration space, however in the second one, we only use the power of clearance function of the MSC Adams® simulation software in order to create an alternative for obtaining configuration space.

The computation times for obtaining one of the 3-D configuration space shown in Figure 2.15 and Figure 2.16 with first and second method are 298.5 and 2414.5 minutes, respectively. One of the 3-D C-Space has 180×180×36 = 1,166,400 configurations. Since 4-D C-Space consists of thirty-six 3-D C-Spaces, it contains approximately 42 million different configurations. It took approximately 179 hours to calculate 4-D C-Space using the first method. Since this simulation of 42 million will take about a month with the second method, the performance comparison of the two methods was carried out during obtaining 3-D C-Spaces. For that reason, it can be said that the first method is 8.1 times more efficient. Since C-Spaces do not change frequently in off-line systems, calculation times are quite reasonable.

2.3 Verification of the Results

After obtaining the configuration space, the rest is related to path planning algorithms which can be A* [55, 56], many variants of rapidly exploring random tree (RRT) [57], probabilistic roadmap (PRM) [58] and so forth. In order to verify obtained 4-D C-Space, 4-D path planning problem was realized as 2-D + 2-D by choosing two axis sets from four existing axes. In this case, whichever two axes will be driven first, motion planning is made by taking the 2-D section of the 4-D C-Space corresponding to the starting positions of the other two axes. In order to drive the second axes pair, motion planning is made by taking the 2-D section of the 4-D C-Space corresponding to the target position of the first two axes. For this example, first 1^{st} and 2^{nd}, then 3^{rd} and 4^{th} axes are driven simultaneously.

In order to plan a sample path, the starting and target positions are selected as in Eqs. (2.18) and (2.19), respectively.

$$\theta_s = [\,\theta_s^1, \theta_s^2, \theta_s^3, \theta_s^4] = [40°, -10°, 160°, 16°] \tag{2.18}$$

$$\theta_t = [\,\theta_t^1, \theta_t^2, \theta_t^3, \theta_t^4] = [264°, 0°, 10°, -10°] \tag{2.19}$$

The configuration space of first driven axes pair and second driven axes pair are given in Figure 2.17 and Figure 2.18. The configuration space is shown in a circular instead of rectangular since the second and fourth axes can be rotated full. Each of the circular C-Spaces has 180×36 grids (axes are divided into 2° steps). A sample path is planned with the help of A* path planning algorithm using two circular C-Spaces as shown in following figures.

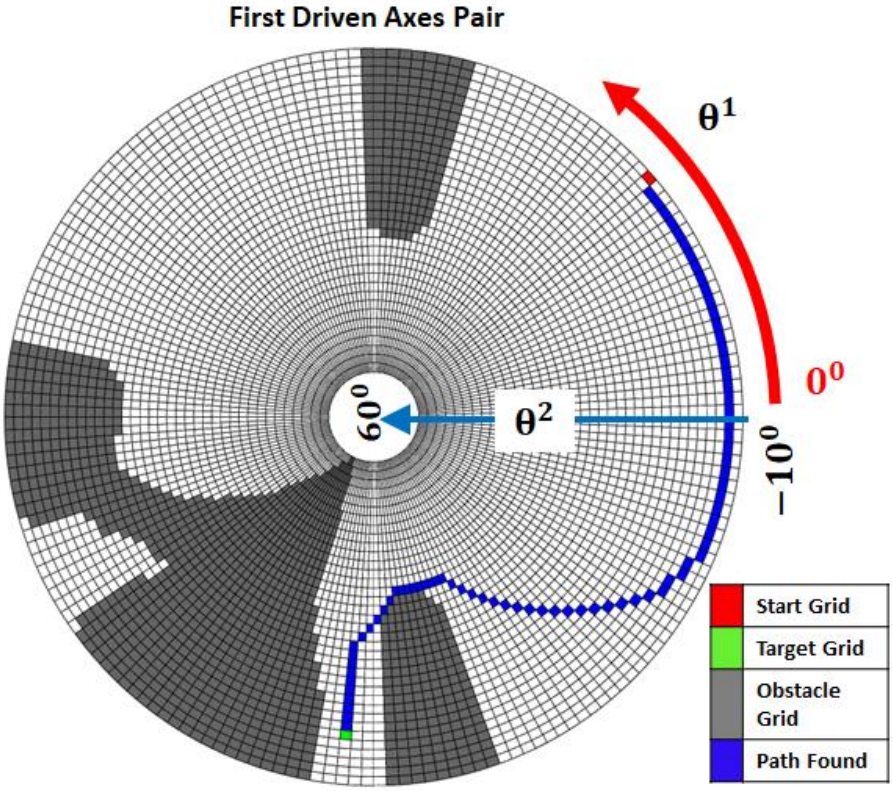

Figure 2.17. 2-D C-Space of double-turret system for first driven axis pair and path planning result

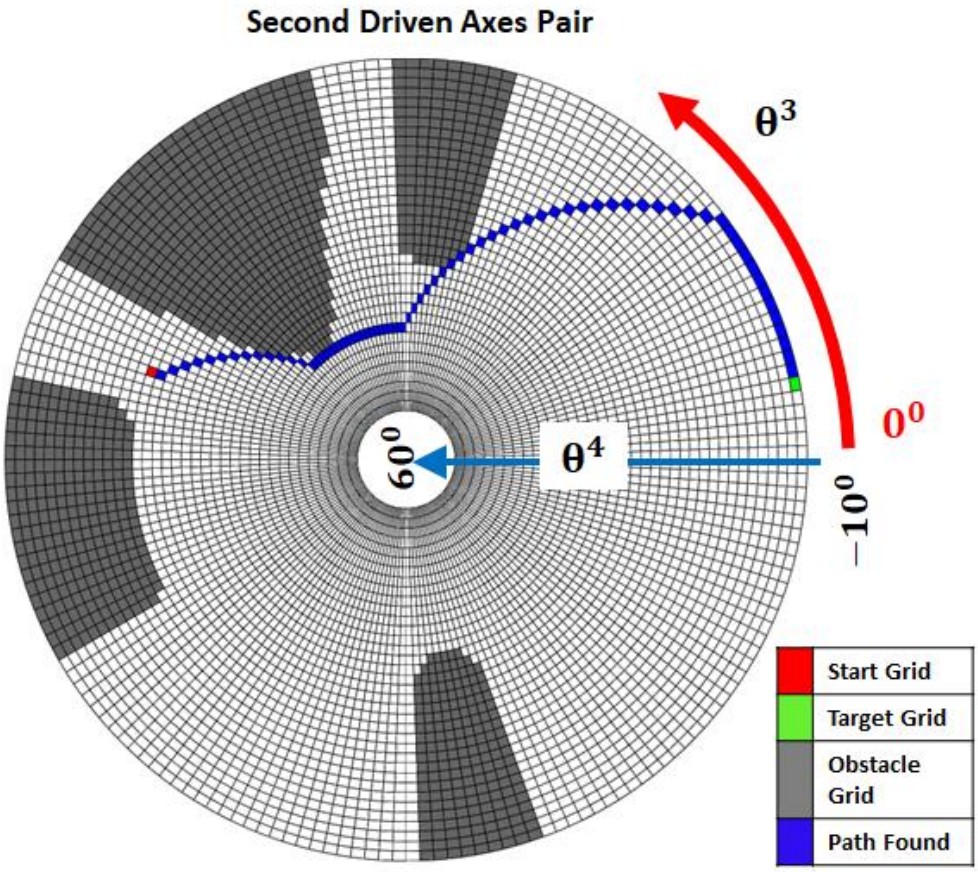

Figure 2.18. 2-D C-Space of double-turret system for second driven axis pair and path planning result

The shortest path is calculated so that the turrets can rotate to the goal position. The turrets will reach the goal position without any collision if the system performs simultaneous traverse and elevation rotations as shown in Figure 2.17 and Figure 2.18. The traverse and elevation motion profiles should be created according to angular speed, acceleration or time requirements. If the angular velocities of the axes are set to 2 °/s, the time required to pass one grid is 1 second. In this case, the system can reach the target position from the specified start position in 156 seconds. Thus, the traverse and elevation motion profiles are created as shown in Figure 2.19. In order to drive double-turret system, the absolute changes of positions are converted into incremental position changes by resetting starting positions as zero.

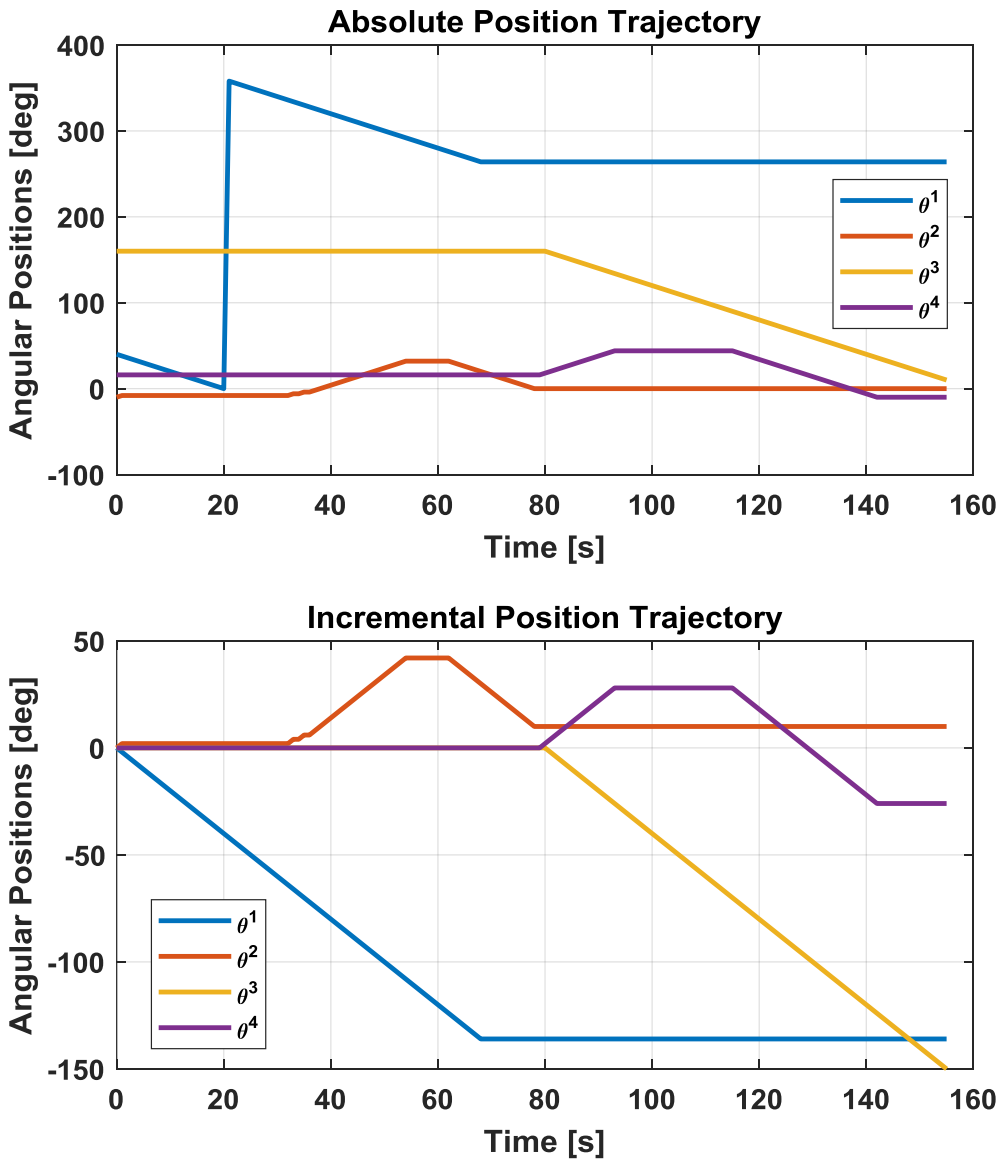

Figure 2.19. The change of position profiles concerning path planning result space

Once the motion profiles are obtained, the rotations of the turret system were simulated on simulation model. As a result of this simulation, the changes of minimum clearances between bodies are obtained as shown in Figure 2.20.

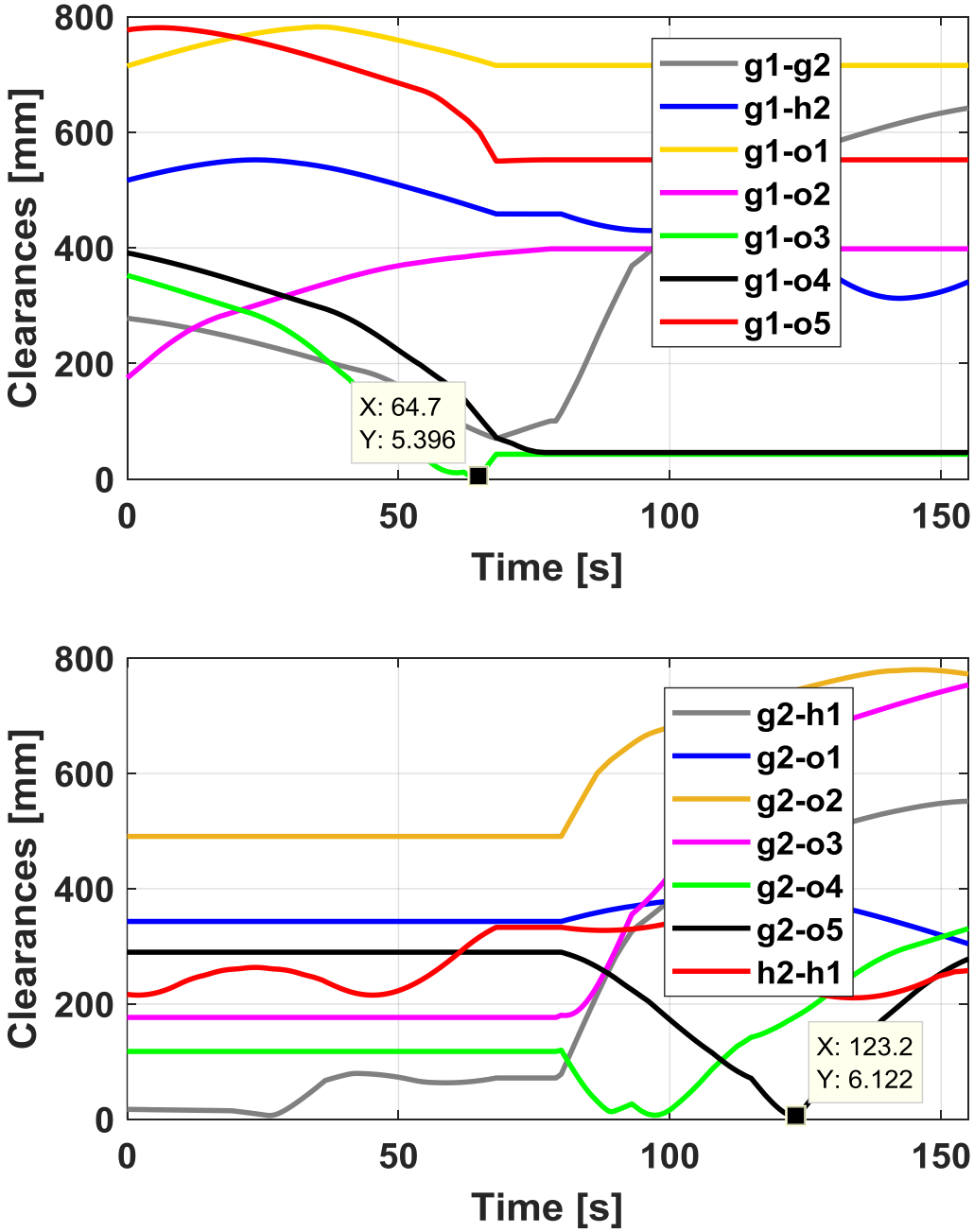

Figure 2.20. The change of minimum clearances between bodies

Accordingly, as the turrets are rotated to their goal positions, the minimum clearance is between g1-o3 which is 5.4 mm and no collision occurs as expected.

2.4 Conclusion

In this chapter, a method has been developed with two different methods to obtain a high-dimensional configuration space. One of the methods is obtaining by using point clouds and the second one is using only simulation software. In first method, the bodies in the system are meshed and converted into points and then the configuration space is obtained by using the method of intersection of point clouds. In the second method, this work is carried out using clearance function which measured minimum distance between the surfaces of defined bodies. A double-turret system is held in the scope of this study. 4-D configuration space of double-turret system is obtained by these two methods. As a result of this, the difference between these two methods is about 1% which depends on the point cloud density. As the point cloud density increases, the difference between the two methods decreases gradually. The first method is 8.1 times more efficient than the second method which is an advantage of the first method. The need to create a point cloud for each part is seen as a disadvantage of the first method. However, for the second method, 3-D cad models of parts and appropriate input profiles are sufficient to carry out the analysis. At the end of the study, a sample path planning with A* algorithm was made by using the previously obtained 4-D configuration space. Then, the accuracy of the configuration space was proved via using the obtained paths on the simulation model of the double-turret system. To fully verify the results of two different methods on the 3 or more dimensions, different path planning algorithms such as RRT and RRT* can be used or different approach like potential field methods can be proposed. Therefore, the focus should be on full verification of the methods for future work by conducting path planning studies on more cases.

CHAPTER 3

MOTION PLANNING OF MULTIPLE-TURRET SYSTEM

The robotic systems have become indispensable for the military as well. They use these systems to increase military capability and security of forces and allow personnel to concentrate on specific tasks which can only be fulfilled by manpower. With the development of robotic systems, most military systems have now become remotely controllable, and this has significantly reduced the need for military personnel [7].

In literature, there are many researches about the collision-free motion in multi-robot systems. However, few papers are concerning collision-free motion planning of dual-arm robots which is a similar topic with double-turret system. Obviously, there are some similarities between them on the aspect of collision-free motion planning. In Freund and Hoyer's algorithm [19], the collision-free motion planning for two robots of R–P type is studied by considering a fictitious robot whose effector permanently collided with that of the master robot. By introducing a safety factor for minimum robot clearance, a trajectory can be achieved without collision. In another research, a 2-D horizontally articulated dual-arm SCARA robot as shown in Figure 3.1 is studied [20]. The configuration space (C-Space) of the related robot (4-R) is obtained based on the reachable manifold and contact manifold theory which is really inadequate to handle complex geometric shapes; therefore, robot arms are illustrated with simple shapes such as rectangles. Also, all revolute joints operate only in a given range, none of them are rotated fully (0-2π) which limits the double-turret system to operate at full capacity.

Figure 3.1. A dual-arm SCARA robot [20]

Being able to place more turrets on the platform depends on their ability to be driven autonomously. Thus, military operational competence can be increased. A multiple cradle launcher in Figure 3.2 can be an example for this study. The four dependent turrets have 8-DOF in total with some constraints [8]. It is impossible to rotate turrets independently because of the space restrictions. These types of launchers are primarily designed for use on naval platforms, armored military vehicles, 4x4 armored vehicles, and armored patrol vehicles as well as for stationary use in order to serve for the defense of strategic assets according to tactical requirements [59].

Figure 3.2. Jobaria multiple cradle launcher [8]

If the former studies are summarized, it is seen that C-Spaces with more than 2 dimensions are not studied, complex geometries are avoided and the full rotation capacity of the R joint is ignored during path planning. In this research, the motion planning of the double-turret system operating in a common workspace is focused on. Unlike other studies, the four-dimensional C-Space of the double-turret system (4-R) is obtained by using intersection of point clouds and also the system operates in the 3-D workspace and two of the four axes can turn fully (0-2π) which creates some difficulties for path planning because of torus and circular shaped C-Spaces. Normally the turrets in the mentioned systems are brought to the desired position by manually and visually controlled which increases the operation time and the risk of collision. These problems can be easily overcome with the proposed method, so that the system can perform its task faster and without the risk of collision. In addition, the movement capability of the system due to its full rotational axes will be preserved.

Within the scope of the study, detailed information will be given about obtaining high-dimensional C-Space with the method of detection of intersection of point clouds. Moreover, with the help of six sequences created by selecting two-axis from four axes, the solution of 4-dimensional path planning problem with 2-D sections of 4-D C-Space as 2-D + 2-D will be explained. The results obtained are verified on the simulation model.

3.1 Motion Planning on 4-D Configuration Space

Motion planning of the systems with four degrees of freedom (DOF) or two systems with 2-DOF should be made which are working together in the same environment without any collisions. As known, the difficulty of motion planning increases exponentially as the number of DOF increases [60]. With the proposed method in this study, it is aimed to reduce the motion planning difficulty of the system with 4-DOF, from $O\,(n^4)$ to $O\,(n^2+n^2)$. The $O\,(n^4)$ problem can be solved as $O\,(n^2+n^2)$ as well as $O\,(n^3+n)$. However, the main reason why we prefer

$O\ (n^2 + n^2)$ instead of $O\ (n^3 + n)$ is to reduce the configuration space size to be used, and therefore the number of configurations to be used, to make path planning algorithms do less search. Because while the number of $n^2 + n^2$ configurations is handled in $O\ (n^2 + n^2)$, this number is $n^3 + n$ in $O\ (n^3 + n)$. In addition, since the system we are working on is 2-D + 2-D, using a 4-dimensional space as 2-D + 2-D is more efficient as a computational cost in this case. Unlike other studies, since high-dimensional C-Spaces are obtained with point clouds, there are no restrictions on the shape of the system parts; therefore, any 3-D complex shaped parts can be used on the system easily. This shows us that this proposed algorithm can be applied to all kinds of systems, not only horizontally articulated systems which are generally used in former studies [19, 20]. Also, all the 4-joints mentioned can have full rotation capacity. Since the joints can take a full turn, the mobility of the system is preserved with the help of the proposed algorithm.

First of all, the 4-D C-Space of the system with the described point cloud method is obtained. Then we move on to motion planning, where the starting and target points are clear. Here, the 2-D sections of 4-D C-Space according to the starting/target points are taken and the motion planning is carried out on them. Information about axes of the double-turret system, maximum-minimum operating ranges and full turn capabilities are given in Table 3.1. As can be seen, there exist four axes in total and the 1st and 3rd axes can rotate fully.

Table 3.1 The axes of double-turret system

Axis Number	Axis Symbol	Definition	Min.	Max.	Full Rot.
1	θ^1	Tra. Axis of 1. Turret	0^0	360^0	Yes (1)
2	θ^2	Ele. Axis of 1. Turret	-10^0	60^0	No (0)
3	θ^3	Tra. Axis of 2. Turret	0^0	360^0	Yes (1)
4	θ^4	Ele. Axis of 2. Turret	-10^0	60^0	No (0)

3.1.1 Sequence

It is aimed to make motion planning by choosing two axis sets from four existing axes. Therefore, there are six options in total which is result of combination of C(4,2). It can be examined these six options under the title of the sequence as given in Table 3.2. Information about the first and the second driven axes is clear there. For example, when 1st sequence is used, first 1st and 2nd, then 3rd and 4th axes are driven simultaneously. Looking at the first driving axes pair of the 1st sequence, it can be seen that the C-Space is cylindrical / circular shaped due to the full rotational capability of the 1st axis, which shows that the motion planning can be done by driving clockwise or counterclockwise. The same is true for the second driven axes pair of the 1st sequence.

Table 3.2 The six sequences

# of Seq.	Order of Axes	First Driven Axes Pair			Second Driven Axes Pair		
		Axes Pair	Type of C-Space	Full Rot.	Axes Pair	Type of C-Space	Full Rot
1	1-2-3-4	1-2	Circular	1-0	3-4	Circular	1-0
2	3-4-1-2	3-4	Circular	1-0	1-2	Circular	1-0
3	1-3-2-4	1-3	Torus	1-1	2-4	Rectangular	0-0
4	2-4-1-3	2-4	Rectangular	0-0	1-3	Torus	1-1
5	1-4-2-3	1-4	Circular	1-0	2-3	Circular	0-1
6	2-3-1-4	2-3	Circular	0-1	1-4	Circular	1-0

There are three different types of C-Spaces which are rectangular, cylindrical/circular and torus shaped in all given sequences. Motion planning should be done by considering all of them.

3.1.2 RCT Path Planning Approach

In the scope of study, rectangular, circular and torus shaped C-Spaces (RCT) are focused on. With the mentioned method within the scope of the study, double-turret system or two dual-arm robots will be able to work in the common environment

without colliding each other. First, 4-D C-Space is obtained and then fast /medium/ optimum path planning can be made using the sequences that mentioned in Section 3.1.2. After obtaining the C-Space, the rest is related to path planning algorithms which can be A* [55, 61], many variants of rapidly exploring random tree (RRT) [62], probabilistic roadmap (PRM) [58], particle swarm optimization (PSO) [63] and so forth. In this study, A* path planning algorithm will be used. The resulting C-Space in Eq. (3.1) is 4-D and consists of $s_1 \times s_2 \times s_3 \times s_4$ grids which are 1 or 0 according to disabled/free regions. The term of s_x indicate the number of grids on the axis of x.

$$CSpace4D: (s_1 \times s_2 \times s_3 \times s_4) \tag{3.1}$$

The initial values and step sizes of axes in the C-Space are defined as in Eqs. (3.2) and (3.3). Superscripts 1-4 refer to axis numbers and subscripts represent definitions.

$$\theta_{in} = [\,\theta_{in}^1, \theta_{in}^2, \theta_{in}^3, \theta_{in}^4] \tag{3.2}$$

$$\delta\theta = [\,\delta\theta^1, \delta\theta^2, \delta\theta^3, \delta\theta^4] \tag{3.3}$$

Within the scope of path planning, the starting and target positions of axes can be expressed as in Eq. (3.4) and (3.5) in degrees.

$$Start\ Position = \theta_s = [\,\theta_s^1, \theta_s^2, \theta_s^3, \theta_s^4] \tag{3.4}$$

$$Target\ Position = \theta_t = [\,\theta_t^1, \theta_t^2, \theta_t^3, \theta_t^4] \tag{3.5}$$

The positions defined above are transformed into start and target grids by rounding as in Eq. (3.6) and (3.7).

$$\theta_{sg} = rnd\left[\frac{\theta_s^1 - \theta_{in}^1}{\delta\theta^1}, \frac{\theta_s^2 - \theta_{in}^2}{\delta\theta^2}, \frac{\theta_s^3 - \theta_{in}^3}{\delta\theta^3}, \frac{\theta_s^4 - \theta_{in}^4}{\delta\theta^4}\right] = \left[\theta_{sg}^1, \theta_{sg}^2, \theta_{sg}^3, \theta_{sg}^4\,\right] \tag{3.6}$$

$$\theta_{tg} = rnd\left[\frac{\theta_t^1 - \theta_{in}^1}{\delta\theta^1}, \frac{\theta_t^2 - \theta_{in}^2}{\delta\theta^2}, \frac{\theta_t^3 - \theta_{in}^3}{\delta\theta^3}, \frac{\theta_t^4 - \theta_{in}^4}{\delta\theta^4}\right] = \left[\theta_{tg}^1, \theta_{tg}^2, \theta_{tg}^3, \theta_{tg}^4\right] \quad (3.7)$$

As an example, if the 1st sequence is examined, since the first driven axes pair in the 1st sequence is 1-2, two-dimensional C-Space in the size of $s_1 \times s_2$ is obtained from the 4-D "CSpace4D" by taking the starting grids of 3rd and 4th axes $[\theta_{sg}^3, \theta_{sg}^4]$ as the 3rd and 4th axis values. In this 2-D C-Space, the starting grid is selected as $[\theta_{sg}^1, \theta_{sg}^2]$, the target grid as $[\theta_{tg}^1, \theta_{tg}^2]$ and path planning is carried out. The path obtained is named as $Path^1$ and this path may cover the change of the position of the 1st axis with respect to the position of 2nd axis. Then, the other 2-D C-Space is derived from "$CSpace4D$" for the second driven axes pair which is 3rd and 4th. This 2-D C-Space is in the size of $s_3 \times s_4$ and obtained by taking the target grids of 1st and 2nd axes $[\theta_{tg}^1, \theta_{tg}^2]$ as the 1st and 2nd axis values. In this 2-D C-Space, the starting grid is selected as $[\theta_{sg}^3, \theta_{sg}^4]$, the target grid as $[\theta_{tg}^3, \theta_{tg}^4]$ and path planning is carried out. The path obtained is named as $Path^2$ and this path may cover the change of the position of the 3rd axis with respect to the position of 4th axis. Using both the $Path^1$ $(2 \times n)$ and $Path^2$ $(2 \times m)$ arrays obtained after the two successful path planning, the position changes of the axes can be obtained by adding the arrays based on the grid as below. In this case, the dimensions of the position change arrays will be (n+m). The unit array is represented as I (1×n) which is in size of n.

$$\theta_g^1 = \left[\, Path^1(1^{st}\, row) \quad \theta_{tg}^1\, x\, I(1 \times m)\right] \quad (3.8)$$

$$\theta_g^2 = \left[\, Path^1(2^{nd}\, row) \quad \theta_{tg}^2\, x\, I(1 \times m)\right] \quad (3.9)$$

$$\theta_g^3 = \left[\, \theta_{sg}^3\, x\, I(1 \times n) \quad Path^2(1^{st}\, row)\,\right] \quad (3.10)$$

$$\theta_g^4 = \left[\, \theta_{sg}^4\, x\, I(1 \times n) \quad Path^2(2^{nd}\, row)\,\right] \quad (3.11)$$

By transforming the position changes from the grid back to the angle, the position change matrix of the axes can be obtained in size of 4× (n + m) as shown in Eq. (3.12). The position change rows in the θ array are relative to the sequence. The order of axes for the first sequence is $[\theta^1, \theta^2, \theta^3, \theta^4]$, whereas for sequence 4, this order will be $[\theta^2, \theta^4, \theta^1, \theta^3]$.

$$\theta' = \begin{bmatrix} \theta^1 \\ \theta^2 \\ \theta^3 \\ \theta^4 \end{bmatrix} = \theta'_{in} + \delta\theta' \cdot \begin{bmatrix} \theta_g^1 \\ \theta_g^2 \\ \theta_g^3 \\ \theta_g^4 \end{bmatrix} = \begin{bmatrix} \theta_{in}^1 + (\theta_g^1 - 1) \cdot \delta\theta^1 \\ \theta_{in}^2 + (\theta_g^2 - 1) \cdot \delta\theta^2 \\ \theta_{in}^3 + (\theta_g^3 - 1) \cdot \delta\theta^3 \\ \theta_{in}^4 + (\theta_g^4 - 1) \cdot \delta\theta^4 \end{bmatrix} \tag{3.12}$$

Since the 1st and 3rd axes can rotate in full turns, 360 mod of the position changes obtained for these axes should be taken. For the 1st sequence, the 1st and 3rd rows of the θ array belong to the 1st and 3rd axis, while the 3rd and 4th rows for the 4th sequence belong to the 1st and 3rd axis.

In a 3-dimensional workspace, the configuration space of any system with 2 joints is two dimensional. If both of the 2 joints are not fully rotated (0-2π), then the C-space is represented as rectangular shaped. If only one of the 2 joints is fully rotated, the C-space is represented as circular. Finally, if both joints have capability of fully rotating, then C-space is represented as torus shaped.

For example, a joint in a system can work between (0-2π). Let the starting position be π/4 radians and the target position 3π/2 radians. It can go from its starting position to its target in two different ways. The first can go from π/4 to 0 radians (also 2π), then from 2π to 3π/2 radians, in this case a total of 3π/4 radians travel. Another option is to go directly from π/4 to 3π/2 radians. In this case, a total distance of 5π/4 radians has been taken. Therefore, if we do not show the C-space of fully rotated axes different from rectangular and we do not let the search algorithms know this, algorithms will use only one of the 2 options. This does not always guarantee that the optimum path will be found, and sometimes even though it is a suitable path, it

does not find solutions. Since the information of fully rotated axes is used within the scope of our study, this situation is discussed in detail.

3.1.2.1 Rectangular Shaped C-Space

Any 2-D C-Space can be represented as rectangular shaped as given in Figure 3.3. After the rectangular shaped C-Space is obtained, path planning can be performed by utilizing the starting and target grids given. If both axes mentioned do not have the capability of full turns (0-2π), there is no way to go from the starting grid to the target grid, as can be seen below.

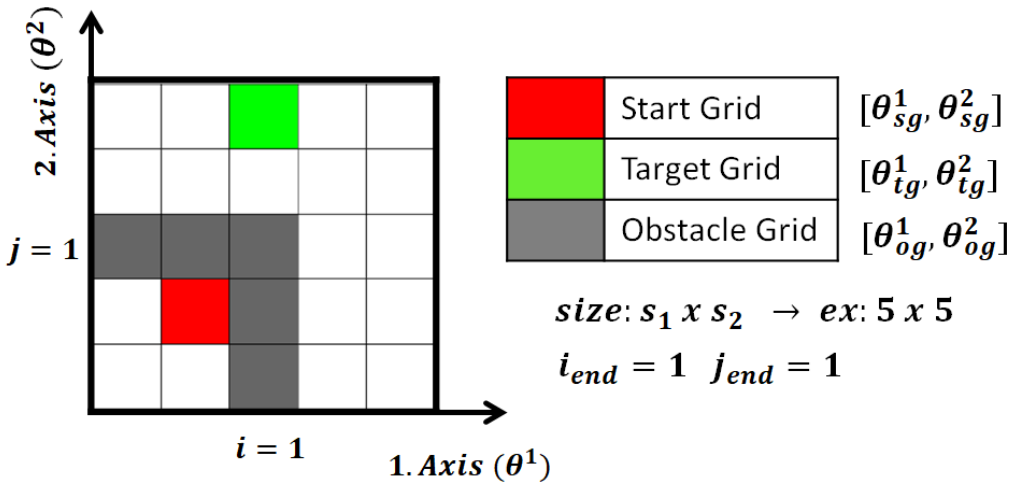

Figure 3.3. A sample rectangular shaped C-Space

3.1.2.2 Cylindrical/Circular Shaped C-Space

If one of the two axes shown in the rectangular shaped C-Space given in Figure 3.3 has the capability of full turns, the C-Space can be represented as cylindrical or circular shaped as shown in Figure 3.4. This type of C-Space is mostly used in this research, while eight out of twelve C-Spaces in sequences are cylindrical/circular shaped C-Space. According to Figure 3.4, the 1st axis can rotate fully, while 2nd axis

cannot. However, in the case only the 2nd axis could rotate fully around itself, the axes would be replaced with each other. The circular notation will be used within the scope of the study.

Path planning is different in this type of C-Space. Considering cylindrical/circular shaped C-Space only as rectangular shaped C-Space limits the system's mobility. In rectangular shaped C-Space given in Figure 3.3, while there is no way between the starting and the target grids, but if one of two axes could rotate fully (0-2π), the cylindrical C-Space may be considered as three rectangular shaped C-Spaces in a column or in a row as shown in Figure 3.5, and so it can be observed that there is actually an appropriate way.

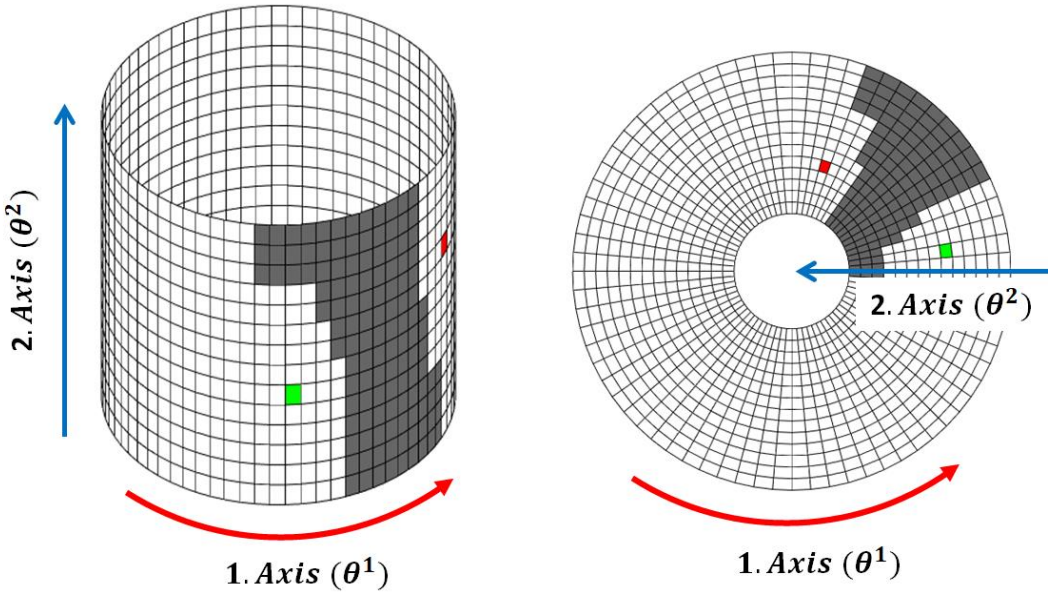

Figure 3.4. A sample cylindrical/circular shaped C-Space (only 1st axis can rotate fully)

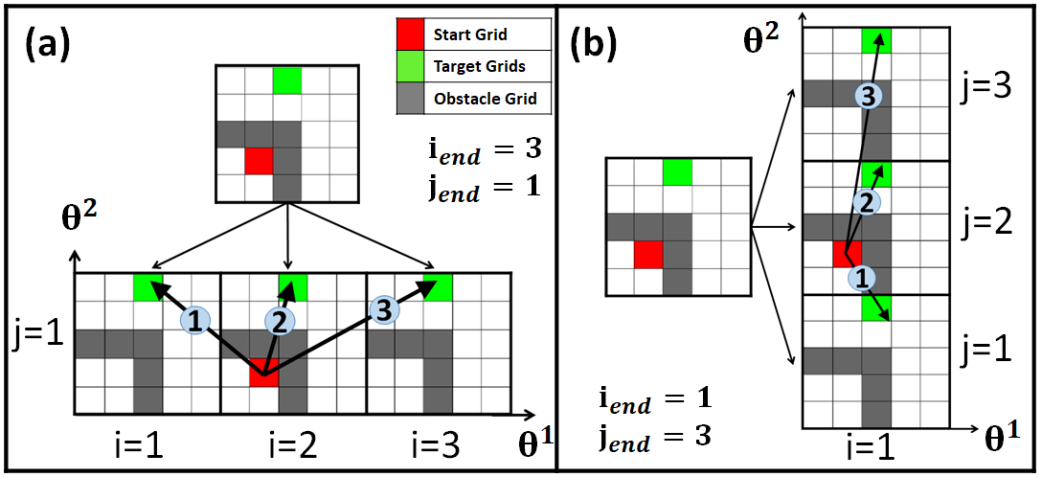

Figure 3.5. The representation of circular shaped C-Space as 3 rectangular shaped C-Space. (a) only 1st axis can rotate fully (b) only 2nd axis can rotate fully

In section (a) of Figure 3.5, the size of the C-Space becomes $3s_1 \times s_2$, while for section (b) it is $s_1 \times 3s_2$. Although the updated C-Spaces in Figure 3.5 have a single starting grid, there are three target grids in each section. This means that three different paths can be searched using three different options shown in the sections, and the shortest path in the successful paths is selected. In this case, it is necessary to update the starting and target grids according to i_{end} and j_{end} values. The starting grid can be updated as follows:

$$\left(\theta^1_{sg}\right)_{i,j} = \theta^1_{sg} + s_1 \left\lfloor \frac{i_{end} - 1}{2} \right\rfloor, \qquad \left(\theta^2_{sg}\right)_{i,j} = \theta^2_{sg} + s_2 \left\lfloor \frac{j_{end} - 1}{2} \right\rfloor \tag{3.13}$$

The target grids are updated according to the i and j values for the three options of each section as given in Eq. (3.14).

$$\left(\theta^1_{tg}\right)_{i,j} = \theta^1_{tg} + s_1\,(i-1), \qquad \left(\theta^2_{tg}\right)_{i,j} = \theta^2_{tg} + s_2(j-1) \tag{3.14}$$

3.1.2.3 Torus Shaped C-Space

Mathematical study of surfaces whose properties do not change with certain deformations such as bending or stretching is called topology of space [55, 58, 62]. The topology for robot arm with two revolute joints can be described as

$$S^1 * S^1 = T^2 \tag{3.15}$$

The above equation means that the C-Space for two revolute joints system is a torus (with no joint limits for the joints). Where S^1 is the circle description, and T^2 is the two-dimensional torus surface [64]. Each joint angle pair corresponds a unique grid on the surface of torus. If both axes shown in the rectangular shaped C-Space in Figure 3.3 have the capability of fully rotation, the C-Space can be represented as torus as in Figure 3.6.

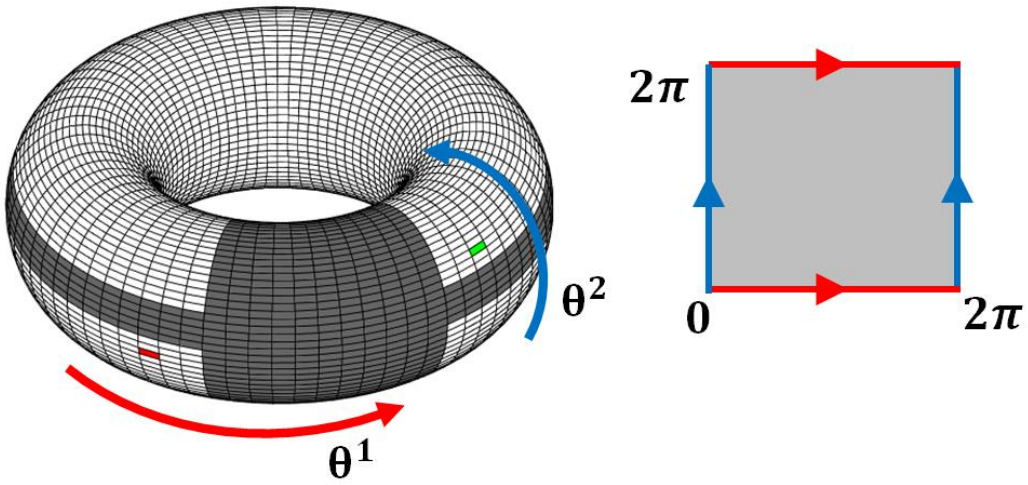

Figure 3.6. A sample torus shaped C-Space (both axes can rotate fully)

In the study of Raheem and Hussain, torus shaped C-Space is defined but path planning is made by considering torus C-Space only as a rectangular shaped C-Space which limits the system's mobility [65]. Path planning is different in this type of C-Space. There are very few studies in the literature that work on torus C-Space and find the optimum path. One of them is the study of Jaillet and Porta [66], in which torus shaped C-Space is introduced as manifold and path planning are made on the

manifold itself by using bidirectional RRT* path planning algorithm as shown in Figure 3.7.

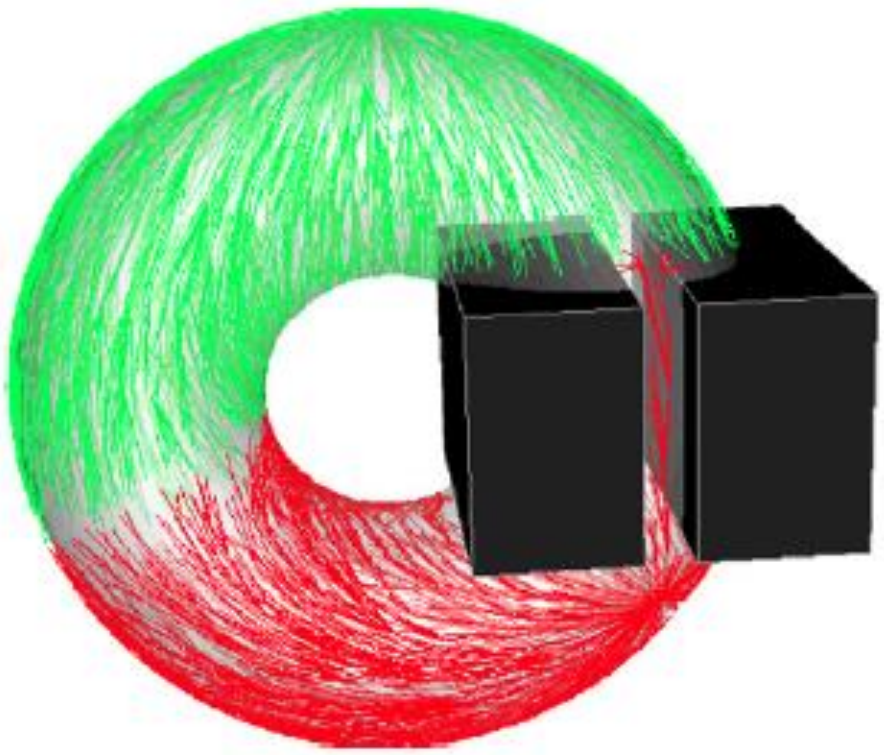

Figure 3.7. A bidirectional RRT* used on a torus-shaped manifold [66]

Using the algorithm of Jaillet and Porta is an option for path planning on the torus shaped C-Space. However, for military systems, sometimes fast but long-distance solutions may be preferred instead of optimum solutions with long processing times in some cases. Therefore, a simple method is proposed to solve path planning problems on torus shaped C-Space as given in Figure 3.8. Using this method helps us to cut the runs when finding first solution or wait until optimum solution. The torus shaped C-Space may be considered as nine rectangular shaped C-shapes which are formed as 3×3 matrix. In this case, the size of the C-Space becomes $3s_1 \times 3s_2$.

Although the updated torus shaped C-Space has a single starting grid, there are nine target grids. Hence, there are nine different paths can be searched using nine different

options shown in the Figure 3.8. The shortest path is selected among the successful paths. Depending on the need, the shortest path (time costly) or the first found path can be preferred. In this case, it is necessary to update the starting and target grids as follows. The starting grid can be updated according to i_{end} and j_{end} values via using Eq. (3.13). The target grids are updated according to the i and j values for the nine options in Figure 3.8 via using Eq. (3.14).

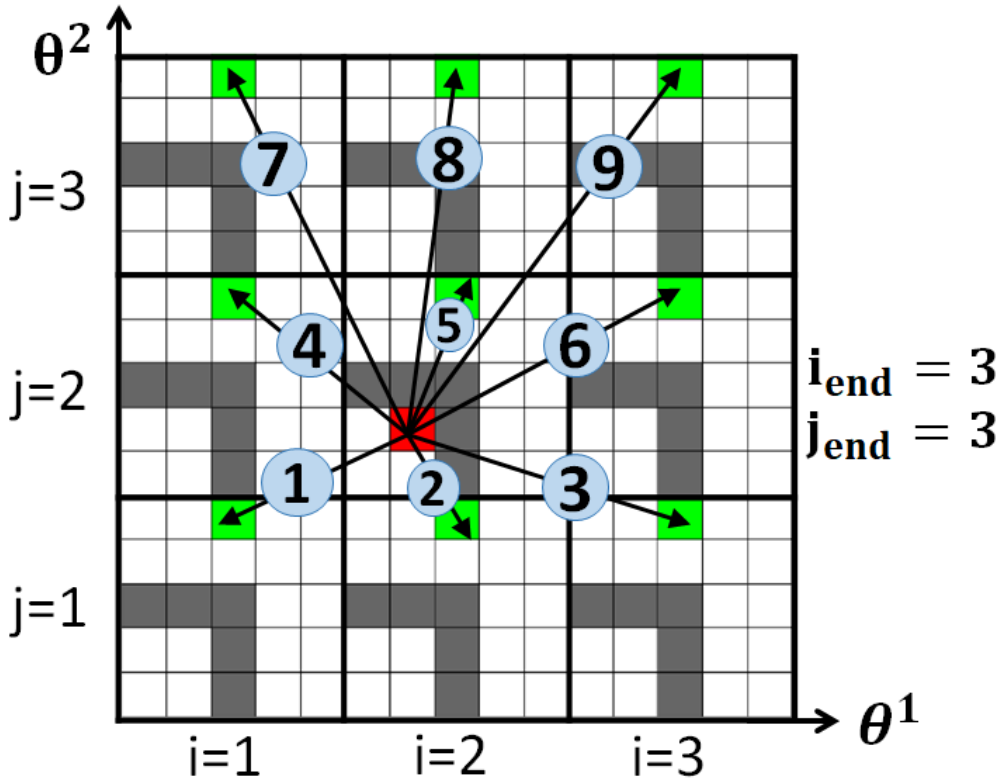

Figure 3.8. The representation of torus shaped C-Space as 9 rectangular shaped

3.1.3 Source Codes

The Algorithm-3.1 starts by initializing 4-D C-space, starting/target positions (θ_s, θ_t), initial values of axes (θ_{in}), step sizes of axes ($\delta\theta$), time step (t_s) and lastly converging option ($ConvOpt$) which is 2, 1 and 0 for fast, medium and optimum one, respectively. The details of inputs are given in Section 3.1.2. Firstly, the transformations of start and target positions to grids are done in Line 1. Then, main for-loop is initiated in Line 2 in size of six which is number of sequences. Between Lines 3-5, two 2-D configuration maps for first and second driven axes pair, their start/target nodes and i_{end}^1 ,j_{end}^1 values are generated according to instant sequence from "$CSpace4D$". Next, two paths are planned using **RCT_PP** algorithm which is explained in Algorithm-3.2 (Lines 6-7). As a result of these path planning, whether there is a path (1) or not (0) and also the data of path found are obtained. If there is a successful path in both and the converging option is fast or medium, then the loop is terminated (Line-8). Otherwise, the loop is expected to end. Among the successful sequences, the sequence with the shortest path is found in Line 11. The $Path^1$ and $Path^2$ in the corresponding sequence are expressed as the first and second driven axis paths. Using these paths, axis position changes (θ) are obtained as described in Section 3.1.2.

Algorithm-3.1: 4-D Path Planning

In: $CSpace4D\ (s_1 \times\ s_2 \times s_3 \times s_4), \theta_s, \theta_t, \theta_{in}, \delta\theta, t_s, ConvOpt$

Out: θ

1 $tranform\ angles\ of\ \ \theta_s, \theta_t\ \ to\ grids \rightarrow \theta_{sg}, \theta_{tg}$

2 **for** $SeqN = 1\ \ to\ \ 6 \rightarrow sequence\ number$

3 $\quad derive \rightarrow MAP2D^1\ \ \&\ \ MAP2D^2\ \ from\ CSpace4D$

4 $\quad define \rightarrow\ i_{end}^1\ \ j_{end}^1\ \ i_{end}^2\ \ j_{end}^2\ acc.to\ type\ of\ C.Spc$

5 $\quad derive \rightarrow StNode^1, StNode^2, TrNode^1, TrNode^2$

$\quad \theta_{sg}, \theta_{tg}\ \ according\ to\ instant\ seq.$

6 $\quad \left[Path_{SeqN}^1 \quad Rslt_{SeqN}^1\right] = \boldsymbol{RCT}_{\boldsymbol{PP}}(MAP2D^1, StNode^1, TrNode^1, ConvOpt, i_{end}^1, j_{end}^1)$

7 $\left[Path^2_{SeqN} \quad Rslt^2_{SeqN}\right] = \boldsymbol{RCT}_{\boldsymbol{PP}(MAP2D^2, StNode^2, TrNode^2, ConvOpt, i^2_{end}, j^2_{end})}$

8 **if** $(ConvOpt == 1\ or\ 2)\ \&\&\ (Rslt^1_{SeqN} \cdot Rslt^2_{SeqN} == 1) \rightarrow$ $break\ loop$ **end if**

9 **end for**

10 **if** $\max(Rslt^1 \cdot Rslt^2) == 1$

11 $SeqRes \leftarrow find\ the\ seq.which\ has\ shortest\ path\ of$ $\left(Path^1_{SeqRes} + Path^2_{SeqRes}\right)$

12 $Path^1 = Path^1_{SeqRes} \quad \& \ Path^2 = Path^2_{SeqRes}$

13 $\theta \leftarrow Path^1 \quad \& Path^2 \quad and\ \theta_{in}, \delta\theta, \theta_{sg}, \theta_{tg}, t_s$

14 **else** $\rightarrow no\ path\ is\ found$ **end if**

15 **return** θ

The Algorithm-3.2 expects a 2-D configuration map in size of $s_1 \times s_2$, starting/target grids, converging option ($ConvOpt$) and finally i_{end}, j_{end} values which are defined for each type of C-space in Section 3.1.2. Firstly, the 2-D configuration map is repeated in size of i_{end} and j_{end} which depend on type of C-Space (Line 1). In Line 2, the starting grid is updated using the sizes of map and i_{end}, j_{end} values. Then, main for-loops are initiated in Lines 3-4 in sizes of i_{end} and j_{end}. Next, a path planning is carried out using well-known **A_Star** algorithm [55, 61] which expects 2-D map and start/target grids and returns whether there is a path (1) or not (0) and also the data of path found (Line 6). If there is a successful path and the converging option is only fast, then the loops are terminated (Lines 7-9). Otherwise, the loop is expected to end. The shortest path is found in Line 13 among the successful paths. As a result of this algorithm, whether there is a path (1) or not (0) and also the data of path found are obtained.

Algorithm-3.2: RCT Path Planning Algorithm (**RCT_PP**)

In: $MAP2D\ (s_1\ \times\ s_2), \left[\theta_{sg}^1 \quad \theta_{sg}^2\right], \left[\theta_{tg}^1 \quad \theta_{tg}^2\right],$
$ConvOpt, i_{end}, j_{end}$

Out: [$Path, Rslt$]

1 $MAP2D_{new} = repmat(MAP2D, i_{end}, j_{end})$

2 $\left[\left(\theta_{sg}^1\right)_{i,j} \quad \left(\theta_{sg}^2\right)_{i,j}\right] = \left[\theta_{sg}^1 + s_1\left\lfloor\frac{i_{end}-1}{2}\right\rfloor \;\middle|\; \theta_{sg}^2 + s_2\left\lfloor\frac{j_{end}-1}{2}\right\rfloor\right)\Big]$

3 **for** $i = 1\ to\ \ i_{end}$

4 **for** $j = 1\ to\ \ j_{end}$

5 $\left[\left(\theta_{tg}^1\right)_{i,j} \quad \left(\theta_{tg}^2\right)_{i,j}\right] = \left[\theta_{tg}^1 + s_1(i-1) \;\middle|\; \theta_{tg}^2 + s_2(j-1)\right)\Big]$

6 $\left[Path_{i,j}\ \ Rslt_{i,j}\ \right] =$
$$\boldsymbol{A}_{\boldsymbol{Star}}\left(MAP2D_{new}, \left[\left(\theta_{sg}^1\right)_{i,j} \quad \left(\theta_{sg}^2\right)_{i,j}\right], \left[\left(\theta_{tg}^1\right)_{i,j} \quad \left(\theta_{tg}^2\right)_{i,j}\right]\right)$$

7 **if** $ConvOpt == 2\ \&\&\ Rslt_{i,j} == 1$

8 **break** (both for loops)

9 **end if**

10 **end for**

11 **end for**

12 **if** $max\left(Rslt_{i,j}\right) == 1 \rightarrow Rslt = 1$

13 $Path = shortest\left(Path_{i,j}\right)$

14 **else** $\rightarrow Rslt = 0, Path = [\,]$ **end if**

15 **return** [$Path, Rslt$]

3.2 Simulation and Verification of Double-turret System

In order to verify the algorithm proposed for 4-D C-space, a double-turret system is considered as an example. The simulation and point cloud models of the double-turret system are given in Figure 3.9 and Figure 3.10, respectively.

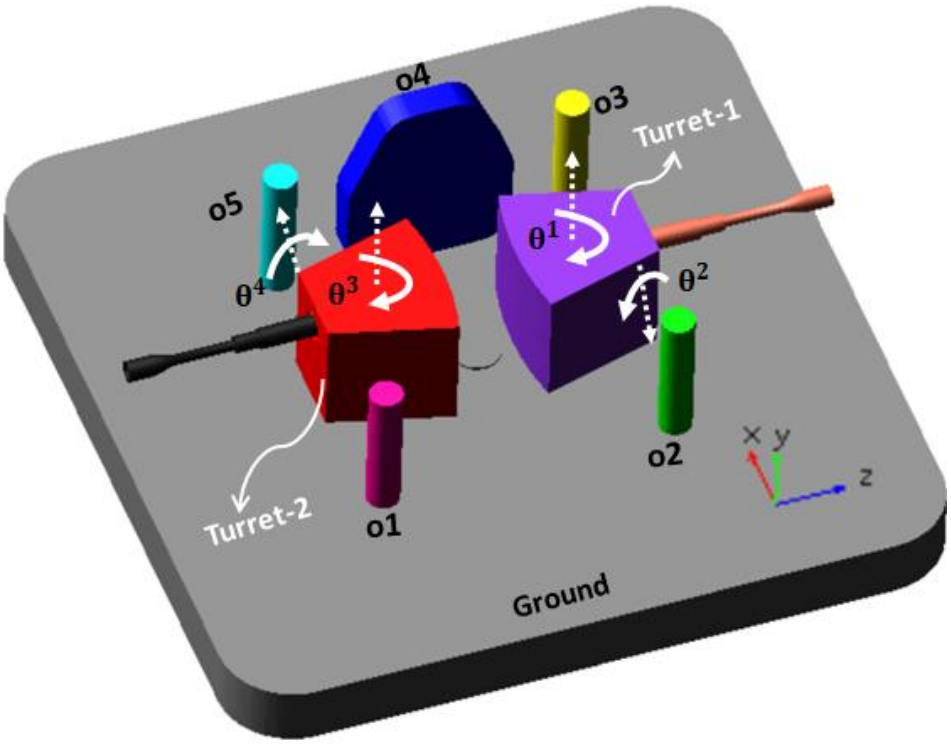

Figure 3.9. The simulation model of double-turret system with stationary obstacles

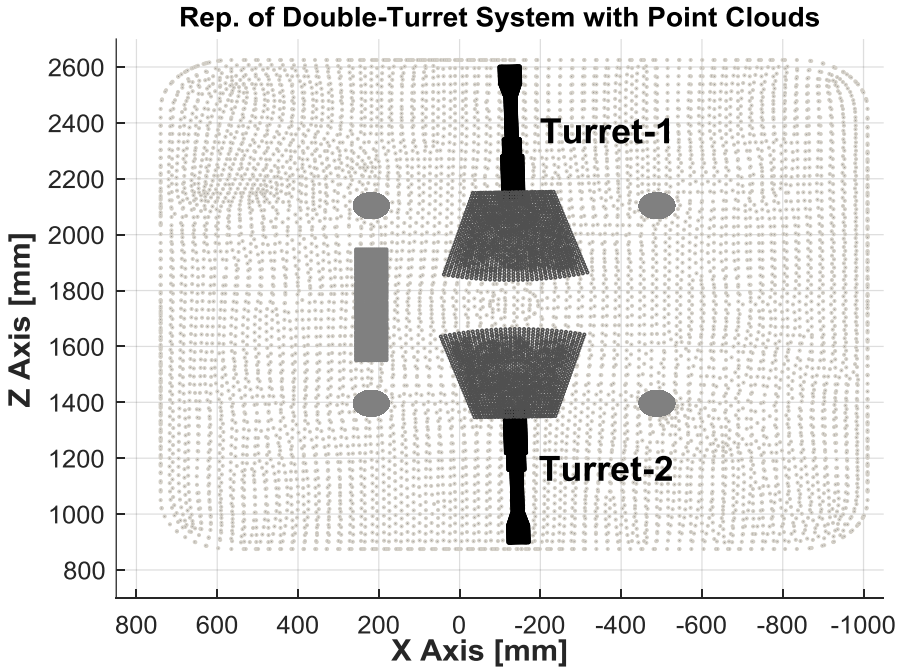

Figure 3.10. The point cloud model of double-turret system with stationary obstacles

The turret has 2-DOF, one for traverse and the other for elevation. In this example, there exists 4-DOF because of two turrets in the common environment. Also, there are five stationary obstacles around these two turrets. The positions of the stationary obstacle are adjusted so that the turrets can make a full rotation with the appropriate elevation angle on the traverse axis. The turrets are identical and their height is 305 mm. The heights of identical stationary obstacles of o1, o2, o3 and o5 350 mm and the height of obstacle of o4 is 485 mm. Therefore, stationary obstacles do not limit the operation of the traverse axes ($\theta^1 \; and \; \theta^3$) when the ascending axes are brought to a suitable angle. However, in case of θ^2 and θ^4 are set to zero, then turrets cannot rotate fully on traverse because of stationary obstacles and another turret.

The 4-D configuration space of the example is obtained by the method mentioned in the section 2. In this 4-D C-Space, the elevation axes (θ^2, θ^4) start from - 10°, end at 60° with a step size of 5°. The traverse axes (θ^1, θ^3) start from 0°, end at 360° with a step size of 5°. In the method of path planning on the C-Space, the number of grids in C-Space has a great effect on the algorithm itself. If the grid is too big, the precision of planning will decrease. If the grid is small, the calculation payload will increase. A reasonable grid decomposition should be based on some optimum criterion [54]. The 4-D C-Space can be represented by a certain number of 3-D configuration spaces. For this example, it can be obtained with fifteen different 3-D C-Spaces for each value of elevation axis of turret-2 (θ^4) in the example of double-turret system. The 3-D representations of this 4-D C-Space are given for 0° and 60° elevation of turret-2 in Figure 3.11 and Figure 3.12, respectively.

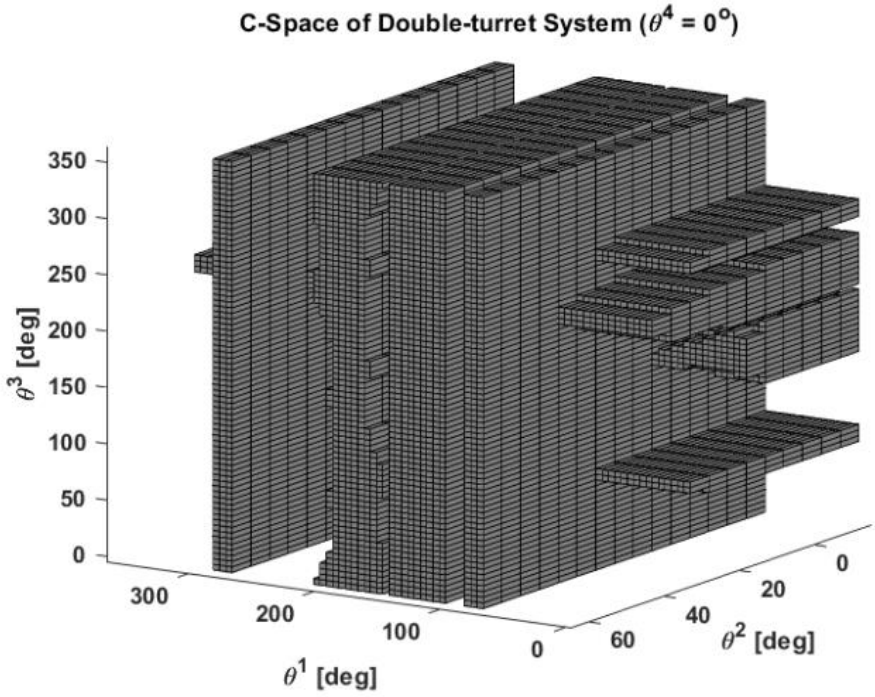

Figure 3.11. 3-D C-Space of double-turret system ($\theta^4 = 0^0$)

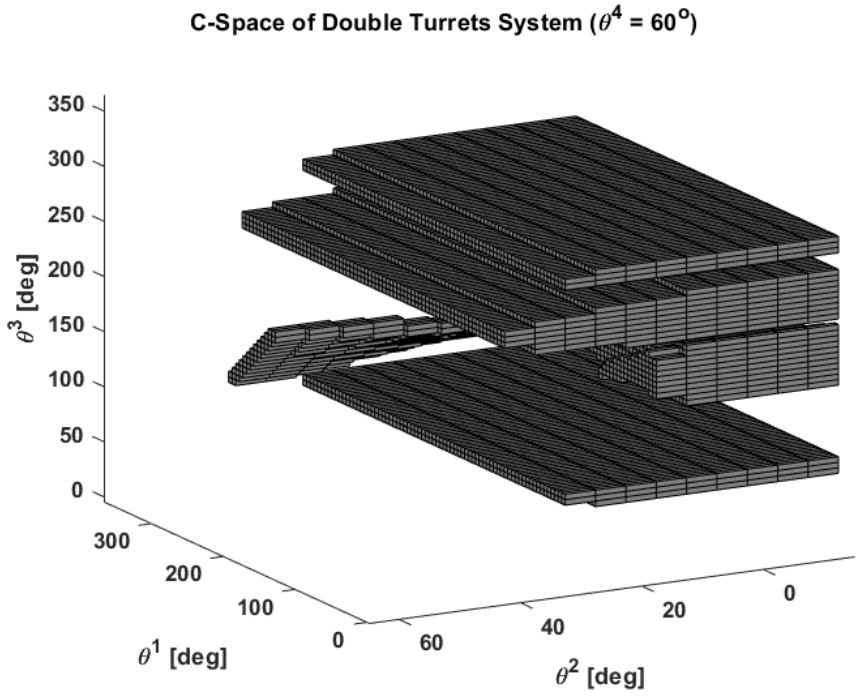

Figure 3.12. 3-D C-Space of double-turret system ($\theta^4 = 60^0$)

3.2.1 Random Simulation Runs

Since it is not possible to plan motions between all possible starting and target positions of the system, random positions were determined and the performance of the system was obtained statistically. A total of six different sequences were determined for path planning. Different configuration spaces, such as rectangular, circular and torus shaped, are used in these sequences. Although a rectangular C-Space has only one path planning option, there are three different options in circular shaped and nine different options in torus shaped C-Spaces. As mentioned in the introduction, sometimes a fast calculated but a long way, sometimes a slow calculated but a short way may be preferred depending on the application. Because the point to focus on here is the sum of calculation and operation time. The algorithm starts searching according to the first defined sequence order. Since there are different numbers of motion planning options according to the C-Space type in the sequence in which the search is made, there are three different converging options depending on when the search will end. They can be called as fast, medium and optimum. In the fast option, the algorithm ends when the first successful path is achieved. In the medium option, after the algorithm scans all the options in that instant sequence, it ends by choosing the shortest path between the options. If it does not find a suitable route in all the options of the sequence it is in, it moves to the next sequence determined and continues the same operation in this sequence. In the optimum option, all the sequences and all the options of these sequences are scanned and the optimum solution is found.

Sequences and options of these sequences should be placed in a suitable order in order for fast and medium converging options to find fast solutions as specified. Otherwise, all converging options will have to scan all sequences, so a quick solution may not be found. Therefore, random simulation runs were performed to determine sequences' order. A thousand random start and target positions were used to measure the performance of the sequences. It was paid attention that these locations are not in the disabled areas in 4-D C-Space. Because if the starting and target positions were

in the disabled area, the sequence performance could not be measured since there would be no suitable way. As a result of random simulations, successful path planning results in the sequences were obtained as shown in the Figure 3.13. With the help of all sequences, successful path planning was made between all random start and target positions. The sequences can be sorted as 1, 2, 5, 4, 3 and 6 according to their success.

In addition, it was found in sequence 2 for nineteen locations where no suitable route was found in the first sequence. This has shown that only the first two sequences are sufficient for path planning between all random positions. However, this result does not decrease the importance of other options for finding the optimum solution. The performance of the sequences depends on the position and shape of the stationary obstacles and the system itself. Because the obtained 4-D C-Space contains all possible motion scenarios including worst cases, collision free motion is guaranteed.

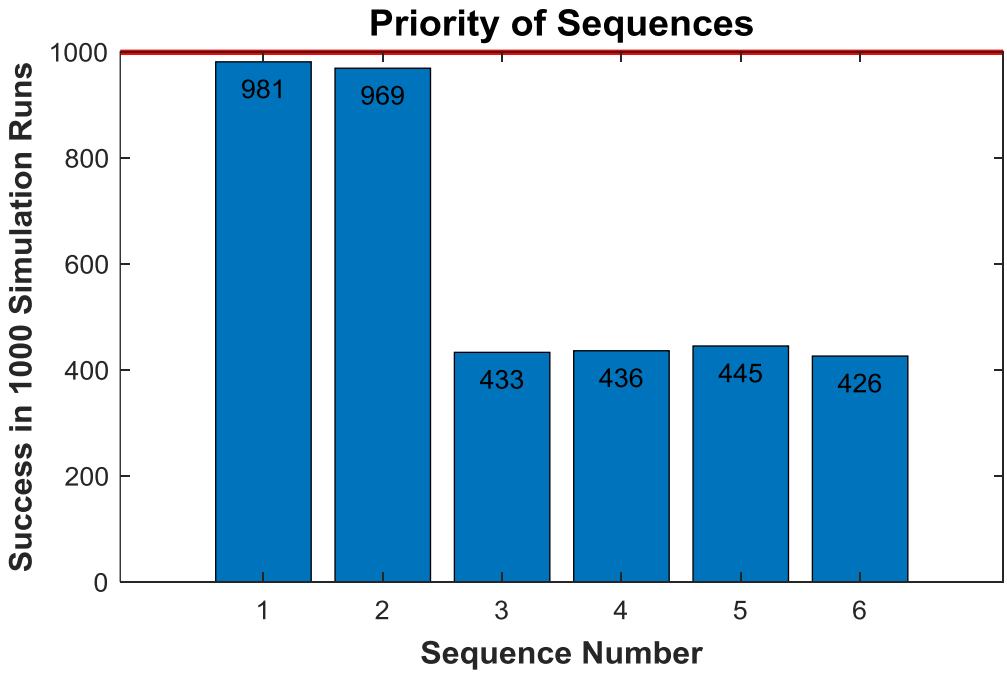

Figure 3.13. Successful path planning results in the sequences

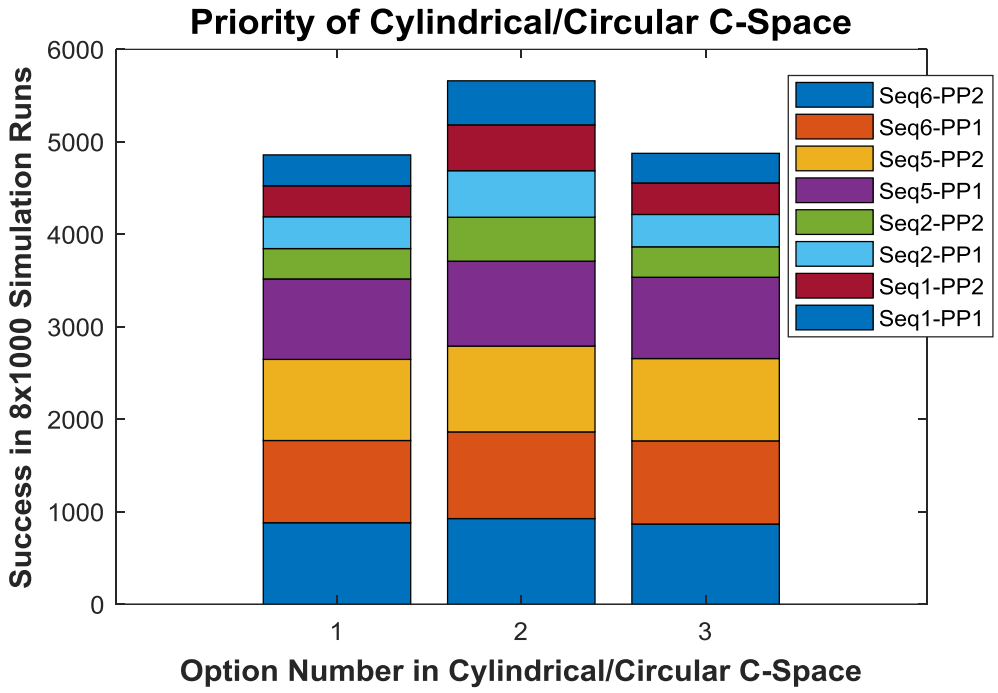

Figure 3.14. Successful path planning results in the options of sequences that have circular type of C-Space

Circular shaped C-Space is used in eight out of twelve configuration spaces of six sequences. There are three different options in circular shaped C-Space. The ordering of these options is also important for the fast-converging option. As a result of random simulations, successful path planning results in the options of sequences which have circular shaped C-Space are obtained as shown in the Figure 3.14. The options in circular shaped C-Space can be sorted as 2, 1 and 3 according to their success. Likewise, torus-shaped C-Space is used in two out of twelve configuration spaces of six sequences. There are nine different options in torus shaped C-Space. The ordering of these options is also important for the fast-converging option. As a result of random simulations, successful path planning results in the options of sequences which have torus C-Space are obtained as shown in the Figure 3.15. The options in torus C-Space can be sorted as 5, 6, 2, 8, 4, 7, 3, 1 and 9 according to their success.

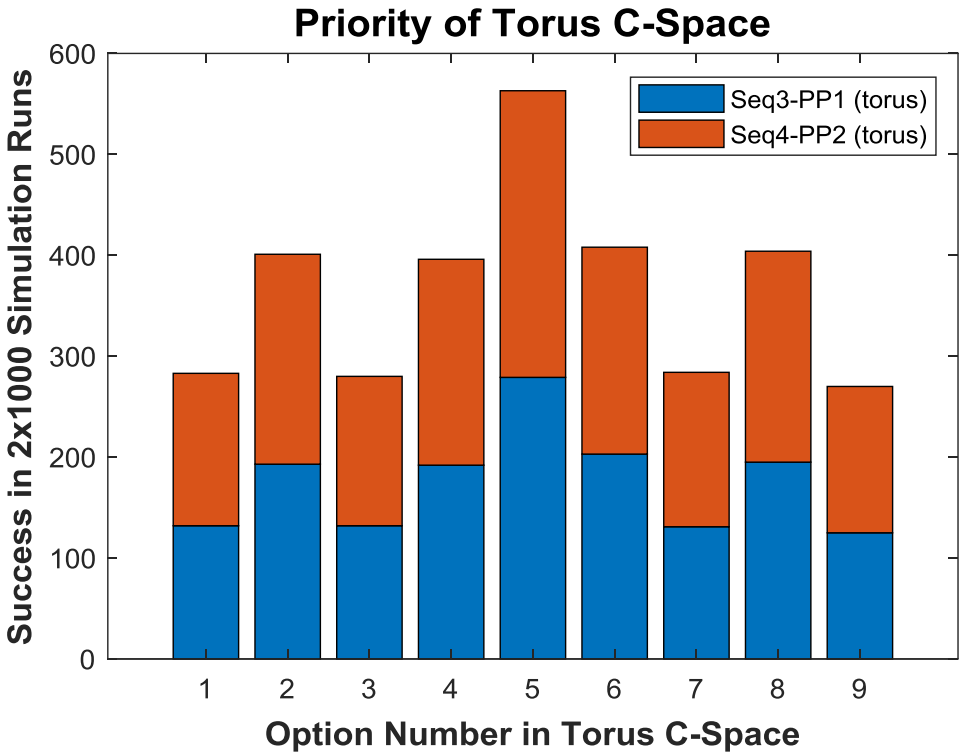

Figure 3.15. Successful path planning results in the options of sequences that have torus type of C-Space

3.2.2 Case Study

With the help of random simulations, the performances of the sequences and their options were statistically measured. So, sequence and option orders have been updated. Then, it is possible to see the differences between the converging options by planning a path between a sample pair of start and target positions which are given in Eqs. (3.16) and (3.17), respectively.

$$\theta_s = [\,\theta_s^1, \theta_s^2, \theta_s^3, \theta_s^4] = [40°, -10°, 320°, 50°] \tag{3.16}$$

$$\theta_t = [\,\theta_t^1, \theta_t^2, \theta_t^3, \theta_t^4] = [300°, 55°, 50°, -5°] \tag{3.17}$$

3.2.2.1 Result for Fast Converging Option

When planning a path with fast converging option using the starting and target positions given in the Eqs. (3.16) and (3.17), the results are obtained as in Figure 3.16. The first successful paths of both driven axes were found in approximately 0.05 seconds in the first option of the 1st sequence using circular shaped C-Spaces.

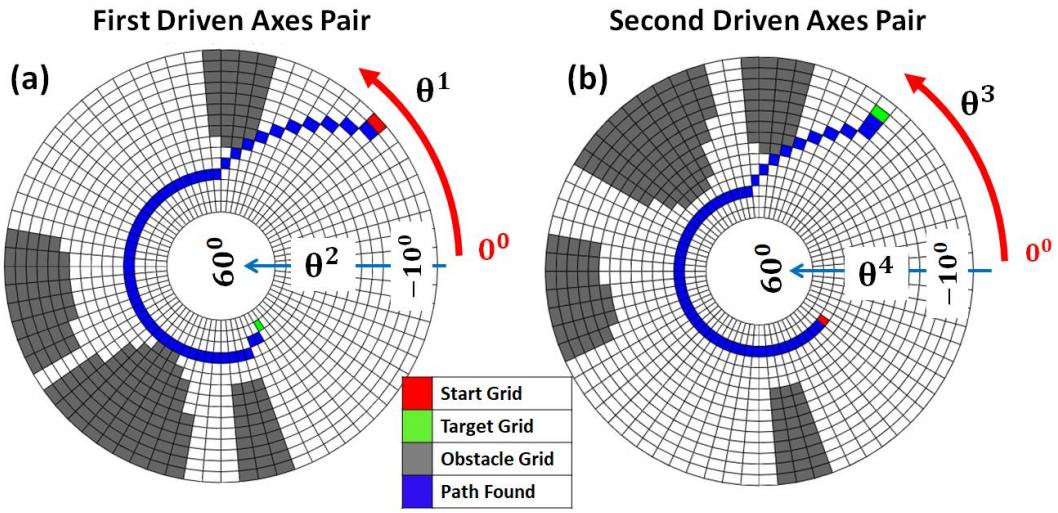

Figure 3.16. A sample path planning using fast converging option (a) first driven axes pair, (b) second driven axes pair

After obtaining the variations of the axis positions relative to each other, their variations over time can be achieved as shown in Figure 3.17 using section 3.1.2. In order to drive double-turret system, the absolute changes of positions are converted into incremental position changes by resetting starting positions as zero.

The traverse and elevation motion profiles should be created according to angular speed, acceleration or time requirements after obtaining grid changes. If the angular velocities of the axes are set to 5°/s, the time required to pass one grid is 1 second. In this case, the system can reach the target position from the specified start position in 109 seconds. The turrets will reach the target position without any collision if the system performs simultaneous traverse and elevation rotations as shown in Figure 3.17.

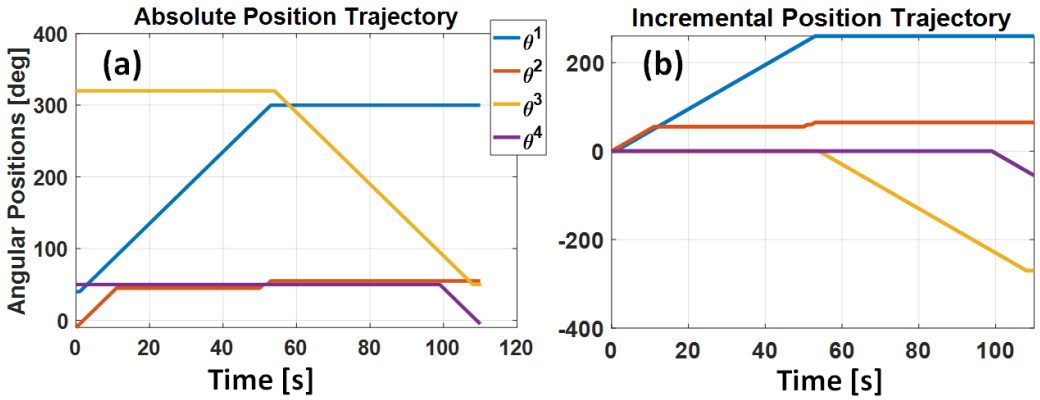

Figure 3.17. The change of position profiles concerning path planning result of fast converging option (a) absolute change, (b) incremental change

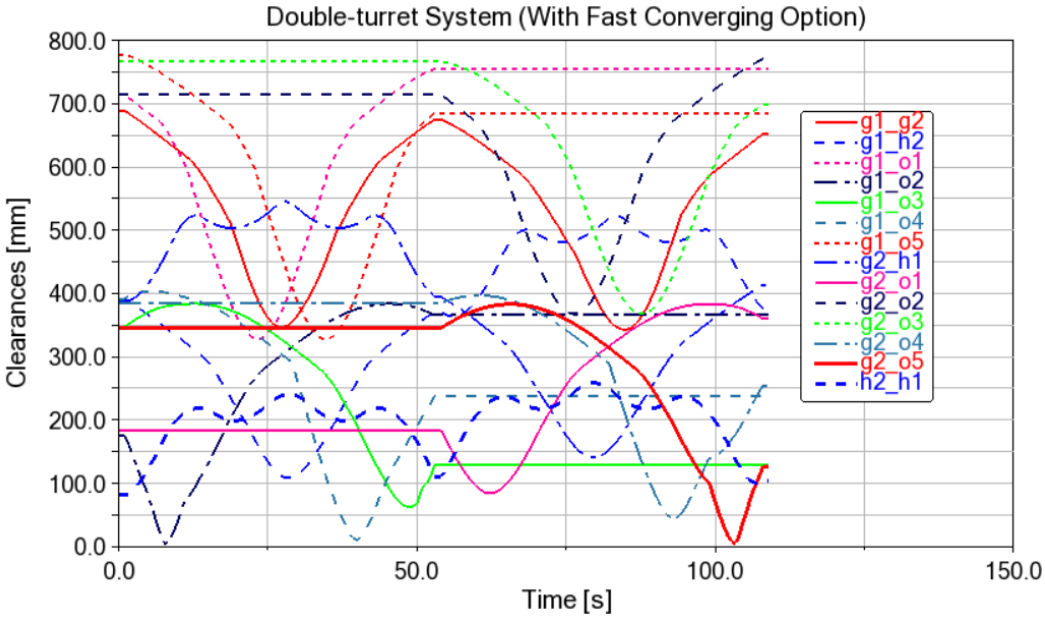

Figure 3.18. The change of minimum clearances between bodies in fast converging option

Once the motion profiles are obtained, the rotations of the double-turret system were simulated in the model. As a result of this simulation, the changes of minimum clearances between bodies are obtained as shown in Figure 3.18. In the Figure 3.18, the letters g, o and h represent guns, obstacles and hulls respectively. The minimum

clearance is between g1-o2 and also g2-o5 which is 5.9 mm and no collision occurs as expected. The safe clearance value can be defined by safe distance between point clouds, δ during obtaining C-Space.

3.2.2.2 Result for Medium Converging Option

When planning a path with medium converging option using the starting and target positions given in the Eqs. (3.16) and (3.17), the results are obtained as in Figure 3.19. The successful paths of both driven axes for medium converging option were found in approximately 0.45 seconds in the second option of the 1st sequence using circular shaped C-Spaces.

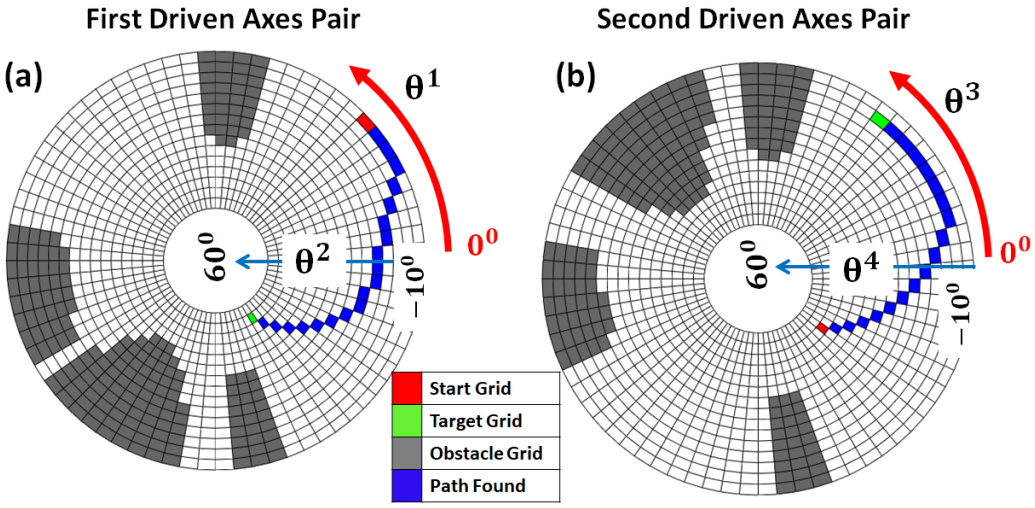

Figure 3.19. A sample path planning using medium converging option (a) first driven axes pair, (b) second driven axes pair

After obtaining the variations of the axis positions relative to each other, their variations over time can be achieved as shown in Figure 3.20 using section 3.1.2. In order to drive double-turret system, the absolute changes of positions are converted into incremental position changes by resetting starting positions as zero. Because absolute position changes are obtained by taking 360 modes of fully rotatable axes, it is not suitable for driving the turrets.

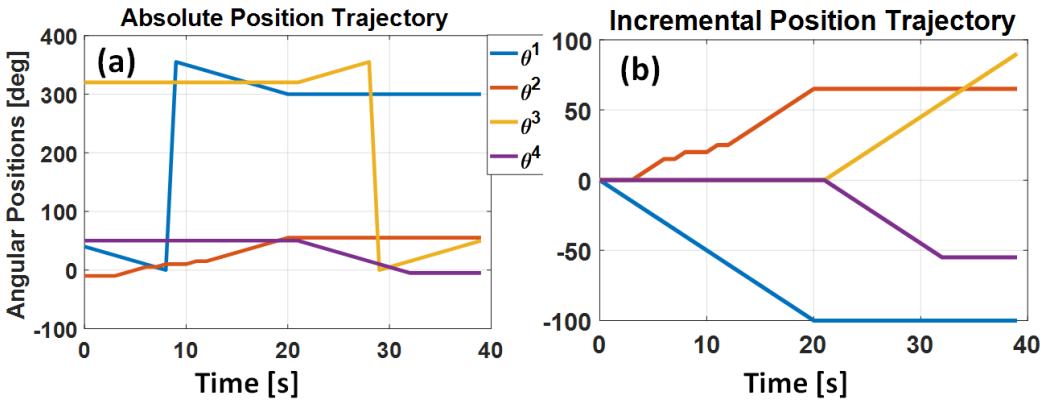

Figure 3.20. The change of position profiles concerning path planning result of medium converging option (a) absolute change, (b) incremental change

If the angular velocities of the axes are set to 5°/s, the time required to pass one grid is 1 second. In this case, the system can reach the target position from the specified start position in 39 seconds. The turrets will reach the target position without any collision if the system performs simultaneous traverse and elevation rotations as shown in Figure 3.20. Once the motion profiles are obtained, the rotations of the double-turret system were simulated in the model. As a result of this simulation, the changes of minimum clearances between bodies are obtained as shown in Figure 3.21. The minimum clearance between bodies is greater than 82.3 mm and no collision occurs as expected.

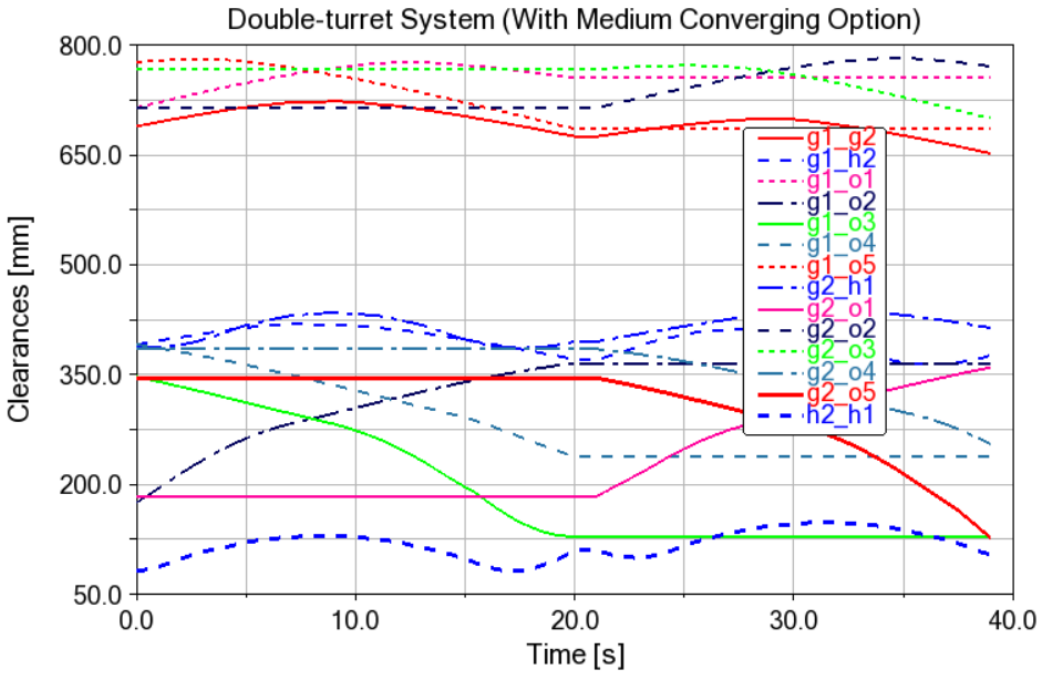

Figure 3.21. The change of minimum clearances between bodies in medium converging option

3.2.2.3 Result for Optimum Converging Option

When planning a path with optimum converging option using the starting and target positions given in the Eqs. (3.16) and (3.17), the results are obtained as in Figure 3.22. The successful path for optimum converging option was found in approximately 23.8 seconds in the third option of the 3rd sequence using torus and rectangular shaped C-Spaces. Since in this optimum converging option all sequences and their options are scanned, the time required to find the optimum path is quite high as expected.

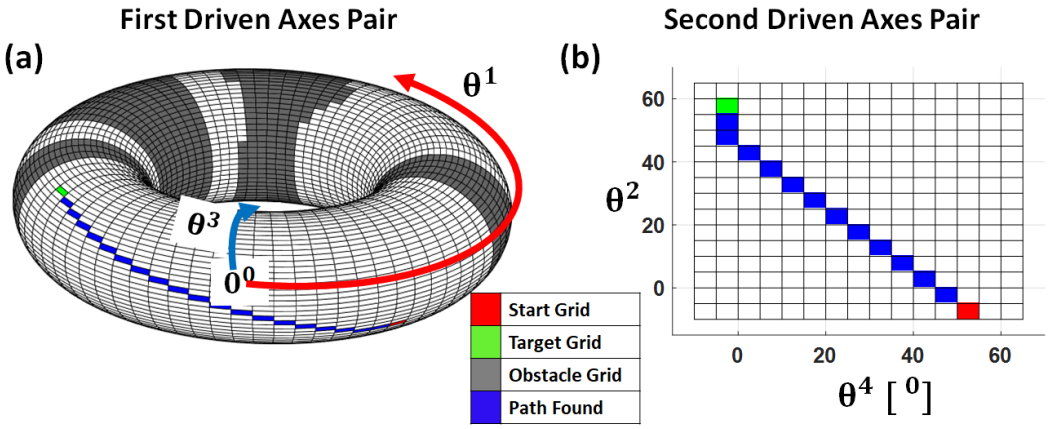

Figure 3.22. A sample path planning using optimum converging option (a) first driven axes pair, (b) second driven axes pair

After obtaining the variations of the axis positions relative to each other, their variations over time can be achieved as shown in Figure 3.23 using section 3.1.2. In order to drive double-turret system, the absolute changes of positions are converted into incremental position changes by resetting starting positions as zero. Because absolute position changes are obtained by taking 360 modes of fully rotatable axes, it is not suitable for driving the turrets.

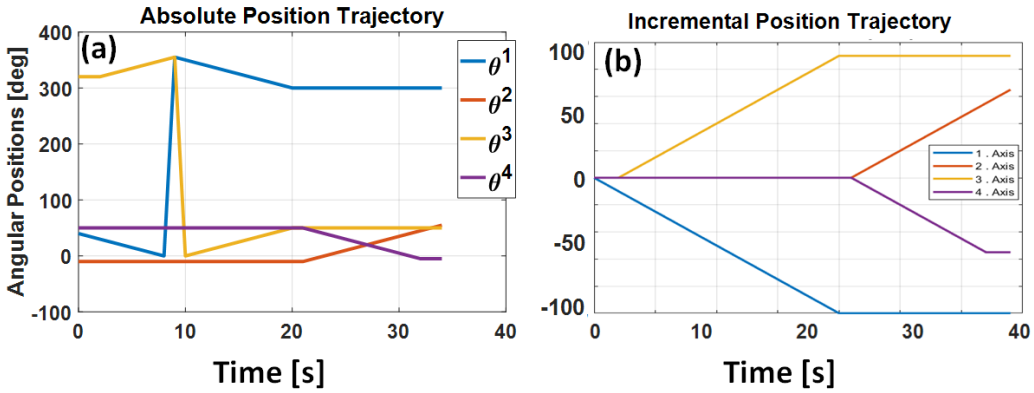

Figure 3.23. The change of position profiles concerning path planning result of optimum converging option (a) absolute change, (b) incremental change

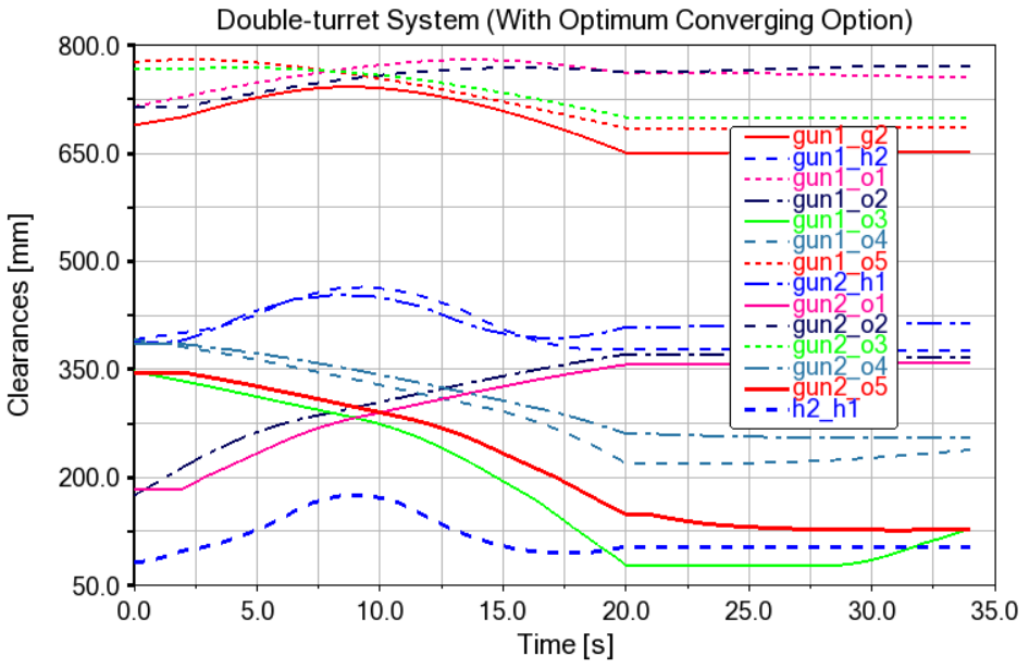

Figure 3.24. The change of position profiles concerning path planning result of optimum converging option (a) absolute change, (b) incremental change

If the angular velocities of the axes are set to 5°/s, the time required to pass one grid is 1 second. In this case, the system can reach the target position from the specified start position in 34 seconds. The turrets will reach the target position without any collision if the system performs simultaneous traverse and elevation rotations as shown in Figure 3.23. Once the motion profiles are obtained, the rotations of the double-turret system were simulated in the simulation model. As a result of this simulation, the changes of minimum clearances between bodies are obtained as shown in Figure 3.24. The minimum clearance between bodies is greater than 77.8 mm and no collision occurs as expected.

3.2.2.4 Comparison of Converging Options

The same 1000 random start and target positions were used to compare the converging options among themselves. As a result, grid-based average path lengths and standard deviations were obtained as shown in Figure 3.25. As expected, the best

option in terms of mean and standard deviation is optimum, then medium and fast options. But the medium and optimum options are quite similar.

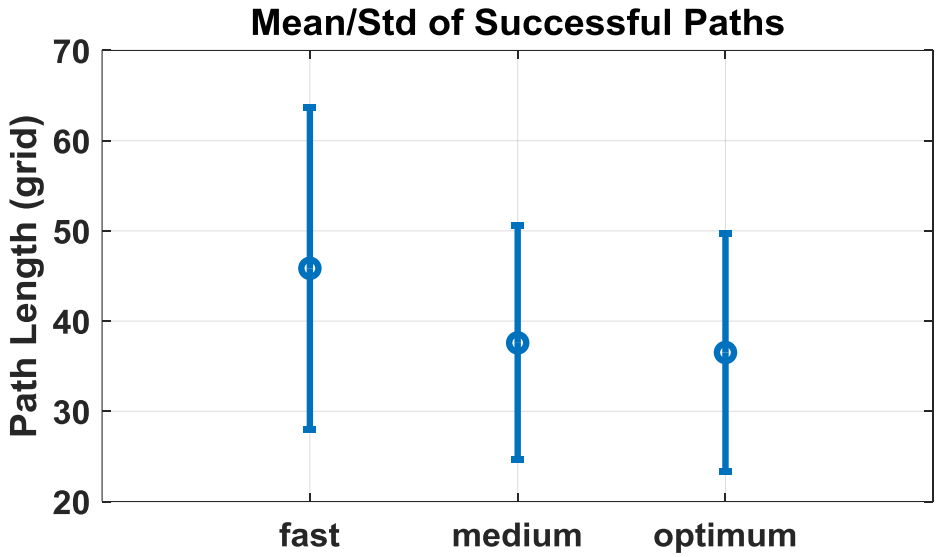

Figure 3.25. Average path lengths of converging options in 1000 random runs

Average calculation times for the 1000 random simulations for all converging options are given in Table 3.3. All the simulations were executed on an Intel Core i7 at 2.00 GHz running Windows 10. The time required to find a more optimal solution is gradually increasing. However, it is seen that medium converging option is the best option within the scope of processing time and path length. But of course, the selection of the converging option depends on the application.

Table 3.3 Average CPU time for path planning

Converging Options	**Average CPU Time (s)**
Fast	0.0517
Medium	0.4567
Optimum	23.5874

When the results of 1000 random simulations run before are examined in detail, the sequence numbers with the most optimum solution in the optimum converging option were 1, 2, 4, 3, 5 and 6, respectively. Since some of the random simulations have more than 1 optimum option, the total number in Figure 3.26 (a) exceeds 1000. As can be seen in the Figure 3.26 (b), 374 of 1000 random simulations only had one optimal solution, while 554 of them had 2 optimum solutions at the same time and so on.

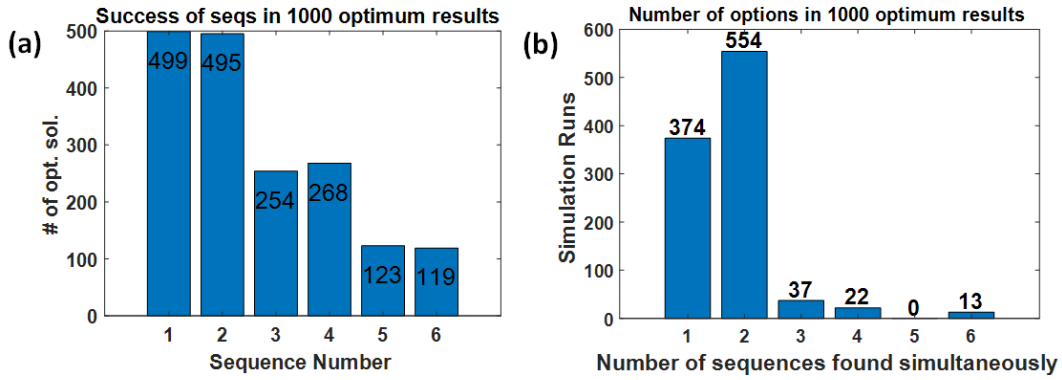

Figure 3.26. Detailed analysis on the optimum converging option (a) performance of sequences (b) number of sequences found simultaneously

3.2.3 Comparison with the Existing Method Proposed for Dual-arm Robot

The algorithm proposed for the dual-arm robot [20], which is most similar to the system studied on, and the method described in all details within the scope of this study was compared. In dual-arm robot work, instead of obtaining a 4-D configuration space, each arm is taken small steps, and as a result of these steps, a 2-D configuration space is produced again each time. It is tried to solve the same problem with the proposed algorithm within the scope of the study. In this context, the dual-arm robot given in Figure 3.1 is modelled with the given dimensions, and obtained the 4-D configuration space of $22 \times 26 \times 22 \times 26$ by point cloud

method. Then, the simulation in the dual-arm robot study is repeated with the proposed algorithm. The operating limits for joints are given in Eqs. (3.18) and (3.19). The start and target positions of all joints are given in Eqs. (3.20) and (3.21), respectively.

$$\theta^1 \in [80°, 185°], \qquad \theta^2 \in [-25°, -150°] \tag{3.18}$$

$$\theta^3 \in [-5°, 100°], \qquad \theta^4 \in [25°, 150°] \tag{3.19}$$

$$\theta_s = [\,\theta_s^1, \theta_s^2, \theta_s^3, \theta_s^4] = [95°, -70°, 70°, 120°] \tag{3.20}$$

$$\theta_t = [\,\theta_t^1, \theta_t^2, \theta_t^3, \theta_t^4] = [110°, -120°, 85°, 80°] \tag{3.21}$$

The operating limits for the dual-arm robot show that any axis is not fully-rotated. In this case, fast and medium converging options are the same. Fast and medium converging options stop the solution as they first find a solution in the 3rd sequence. Since the optimum converging option looks at all sequences, it finds solutions in both of the 3rd and 4th sequences and gives the conclusion that the optimum solution is in the 3rd sequence.

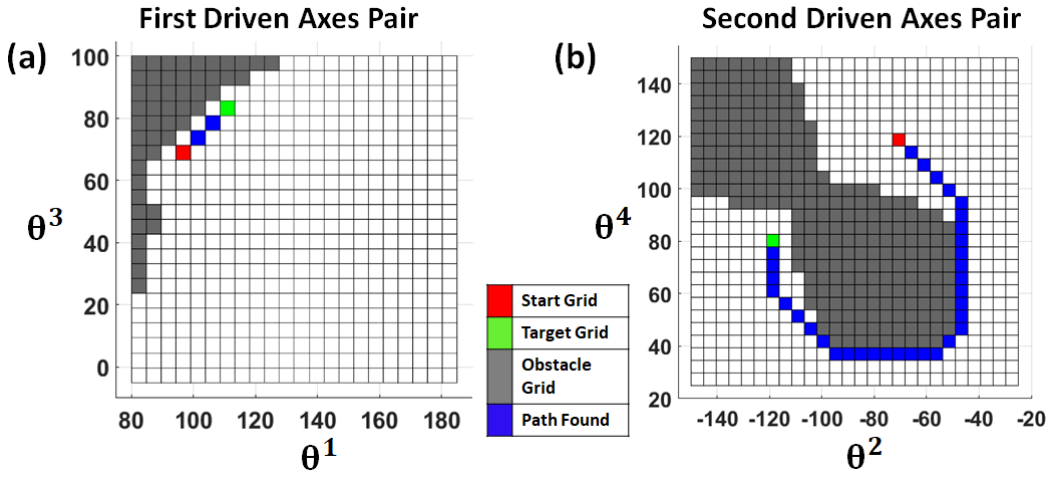

Figure 3.27. A sample path planning for dual-arm robot (a) first driven axes pair, (b) second driven axes pair

After obtaining the variations of the axis positions relative to each other, the simulation of dual-arm robot is done as shown in Figure 3.28. The collision-free motion planning of dual-arm robot is achieved by the simultaneous movement of both arms. UL, LL, UR and LR represent upper-left, lower-left, upper-right and lower-right, respectively.

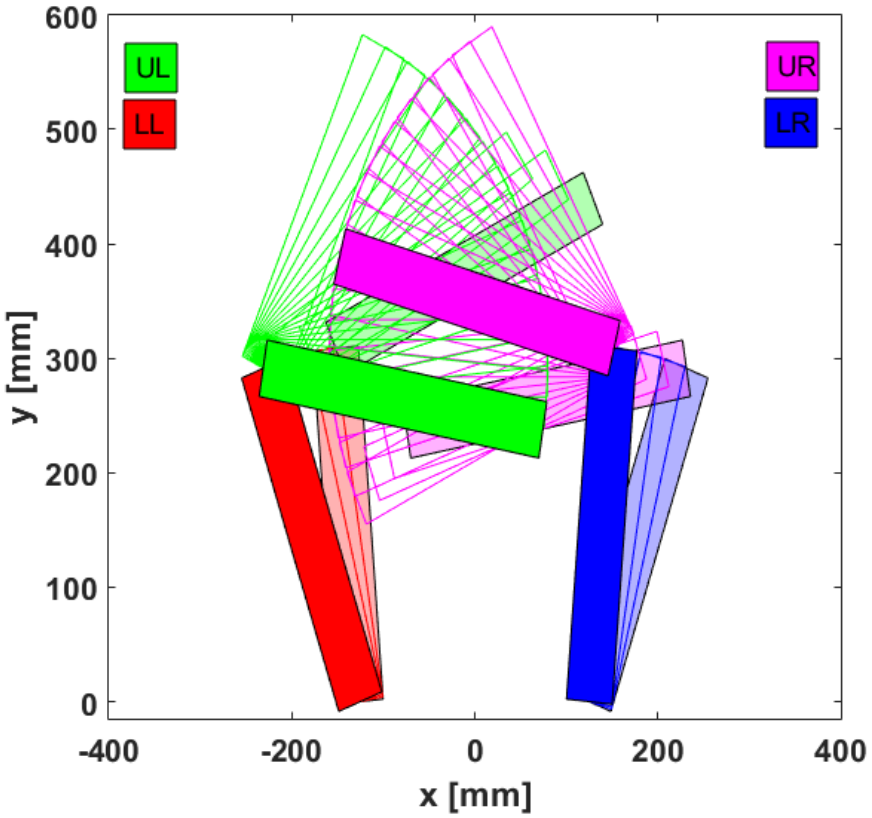

Figure 3.28. Simulation of the dual-arm robot where two arms move simultaneously.

After obtaining the variations of the axis positions relative to each other, their variations over time can be achieved as shown in Figure 3.29 using section 3.1.2.

If the angular velocities of the axes are set to 5°/s, the time required to pass one grid is 1 second. In this case, the system can reach the target position from the specified start position in 40 seconds. Once the motion profiles are obtained, the rotations of the dual-arm robot are simulated in the simulation model. As a result of this simulation, the changes of minimum clearances between bodies are obtained as

shown in Figure 3.30. The minimum clearance between bodies is greater 6.3 mm and no collision occurs as expected.

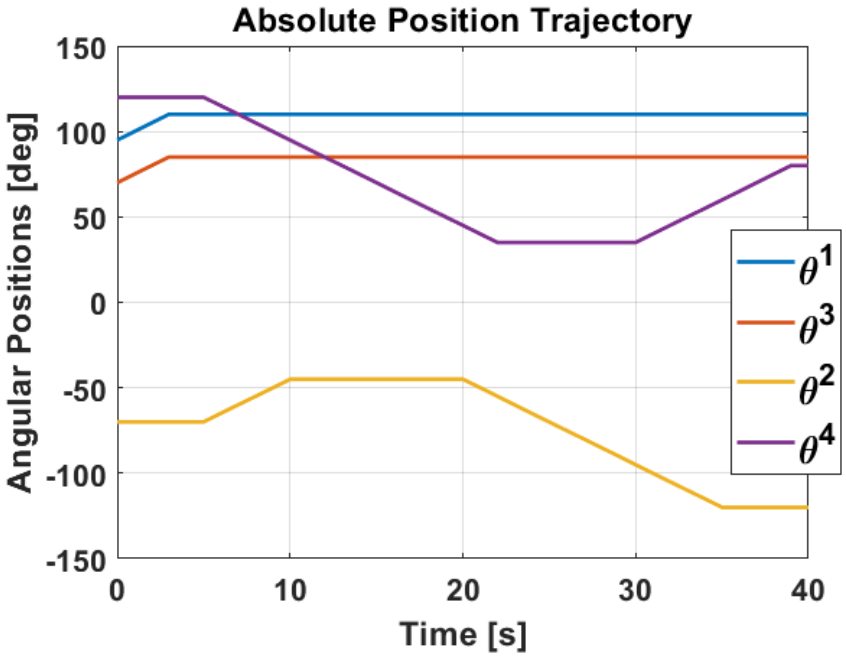

Figure 3.29. The change of position profiles concerning path planning result for dual-arm robot

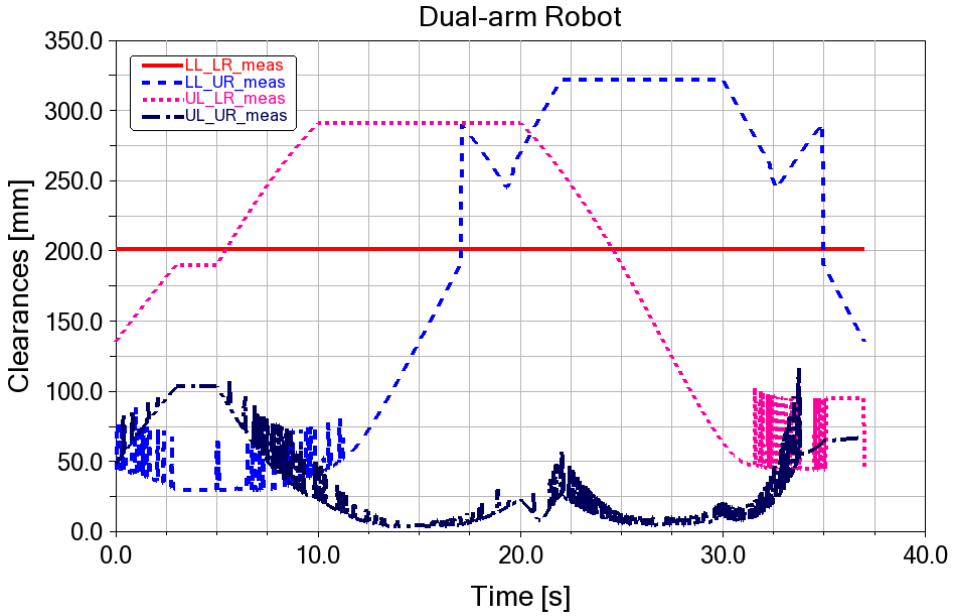

Figure 3.30. The change of minimum clearances between bodies for dual-arm robot

In order to realize this simulation in the dual-arm robot study, the configuration space is calculated approximately 60 times in real time and made 60 times path planning [20]. It is stated that each discrete C-Space constructing took 12ms and the C-Space obstacle boundaries decomposing took 6ms, not including the time taken by the movement itself. Moreover, C-space construction was comparatively faster as each arm of the dual-arm robot was simply modeled as rectangular. However, it may not be possible to apply the old method directly in real-time since it will take a lot of time to create a C-space using the volumes of more complex arms. However, within the scope of the study, with the help of 4-D C-space, the C-space does not need to be calculated each time, thus saving the mentioned calculation time. Moreover, since the 4-D configuration space is obtained once using point clouds, the exact same model can be used, as parts of the robot do not need to be modeled with familiar shapes. In addition, since there are 6 sequences for any situation, path planning can be tried a maximum of 6 times for non-fully rotated axes. In addition, since there are 6 sequences for any situation, path planning can be tried a maximum of 6 times. Finally, it is overlooked that fully rotated axes can move cw or ccw using the old method. However, with the proposed method, with the axes being fully rotated, every situation is considered and the optimum path is planned.

3.3 Handling of Configuration Space of Aligned Multiple-turret System

More than one gun turret can operate on the same platform. While there are millions of different configurations even in 4-D space for a double-turret system, billions or even trillions of different configurations can be defined for systems with 3 and more turrets. Increasing the degrees of freedom naturally means increasing the size of the configuration space exponentially. Aligned multiple-turret system can be illustrated as in Figure 3.31.

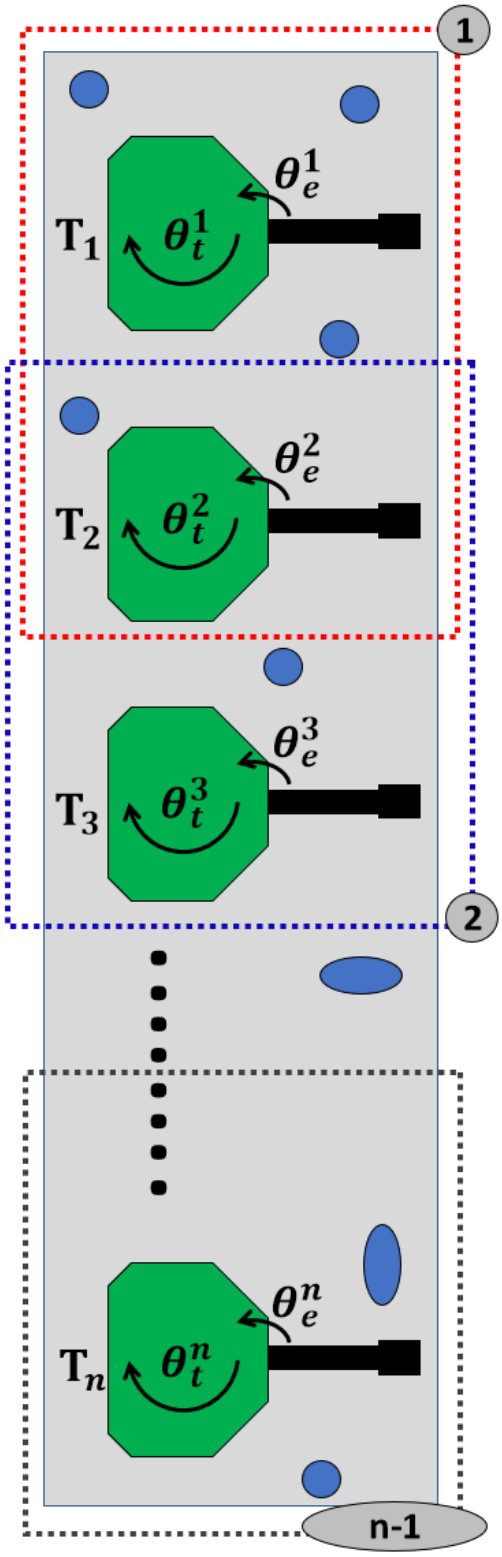

Figure 3.31. Aligned multiple-turret system

If (n) number of turrets were able to interact with each other, an n-dimensional configuration space could be mentioned. However, since an individual turret interacts only with neighboring turrets, motion planning can be made by obtaining (n-1) number of 4-D configurations space. One of the 4-D configuration space includes the entire configuration space with given angle ranges of two adjacent turrets. At the stage of obtaining this 4-D space, all other turrets except 2 related turrets are considered as if they do not exist. All fixed obstacles around are always taken into account when obtaining all configuration spaces. The 4-D configuration space of first and second turrets (T_1, T_2) can be named as $\left(C_{1,2}\right)^{fix}$ whose dimensions are d_t^1, d_e^1, d_t^2 and d_e^2. Once all fixed 4-D configuration spaces are obtained, live 4-D configuration spaces are obtained by super-positioning using the instantaneous positions of the adjacent turrets. When obtaining live configuration spaces in the aligned multiple-turret system shown in Figure 3.31, superposition is made as shown in Eq. (3.22). However, for the first and last double turrets the Eq. (3.22) is updated as in Eqs. (3.23) and Eq. (3.24), respectively.

$$\left(C_{n,n+1}\right)^{live} = SuperPos\left\{\left(C_{n-1,n}\right)^{fix}, \left(C_{n,n+1}\right)^{fix}, \left(C_{n+1,n+2}\right)^{fix}\right\} \tag{3.22}$$

$$\left(C_{1,2}\right)^{live} = SupPos\left\{\left(C_{1,2}\right)^{fix}, \left(C_{2,3}\right)^{fix}\right\} \tag{3.23}$$

$$\left(C_{n-1,n}\right)^{live} = SupPos\left\{\left(C_{n-2,n-1}\right)^{fix}, \left(C_{n-1,n}\right)^{fix}\right\} \tag{3.24}$$

Algorithm-3.3: SuperPos

In: $(C_{1,2})^{fix}$, $(C_{2,3})^{fix}$, $(C_{3,4})^{fix}$ $\theta_t^1, \theta_e^1, \theta_t^4, \theta_e^4$

Out: $(C_{2,3})^{live}$

1 $Temp_{12}(d_t^2 \times d_e^2)$ = 2-D section of $(C_{1,2})^{fix}$ for θ_t^1, θ_e^1

2 $Temp_{34}(d_t^3 \times d_e^3)$ = 2-D section of $(C_{3,4})^{fix}$ for θ_t^4, θ_e^4

3 $(C_{2,3})^{live} = (C_{2,3})^{fix}$ →4-D Configuration Spaces

4 **for** $i = 1$ *to* d_t^3

5 **for** $j = 1$ *to* d_e^3

6 $(C_{2,3})^{live}(:,:,i,j) = OR\left\{(C_{2,3})^{fix}(:,:,i,j), Temp_{12}\right\}$

7 **end for**

8 **end for**

9 **for** $i = 1$ *to* d_t^2

10 **for** $j = 1$ *to* d_e^2

11 $(C_{2,3})^{live}(i,j,:,:) = OR\left\{(C_{2,3})^{fix}(i,j,:,:), Temp_{34}\right\}$

12 **end for**

13 **end for**

14 **return** $(C_{2,3})^{live}$

The Algorithm-3.3 converts fix configuration space of a turret pair to live configuration space by super positioning and it expects three different 4-D configuration spaces in size of $d_t^1 \times d_e^1 \times d_t^2 \times d_e^2$ and current positions of adjacent turrets which are $\theta_t^1, \theta_e^1, \theta_t^4$ and θ_e^4. For instance, in order to obtain live configuration space of second and third turrets $(C_{2,3})^{live}$, firstly the 2-D section of adjacent configuration spaces are obtained for given current positions ($\theta_t^1, \theta_e^1, \theta_t^4$ and θ_e^4) as shown in Line 1 and Line 2. Between Lines 4 to 7, 2-D sections of one of the adjacent 4-D configuration space are superposed with related part of 4-D configuration space of concerned turret pair. Then, the same is repeated for other adjacent 4-D configuration space between Lines 9 to 13. Finally, it returns to live configuration

space $\left(C_{2,3}\right)^{live}$. The live configuration space of the respective turret pair is the same as long as the positions of adjacent turrets do not change. If any axis of adjacent turrets moves, the live configuration space needs to be recreated by super positioning.

3.4 Simulation and Verification of Multiple-turret System

In the scope of this study a triple-turret system is considered as an example. The simulation and point cloud models of the triple-turret system are given in Figure 3.32 and Figure 3.33, respectively.

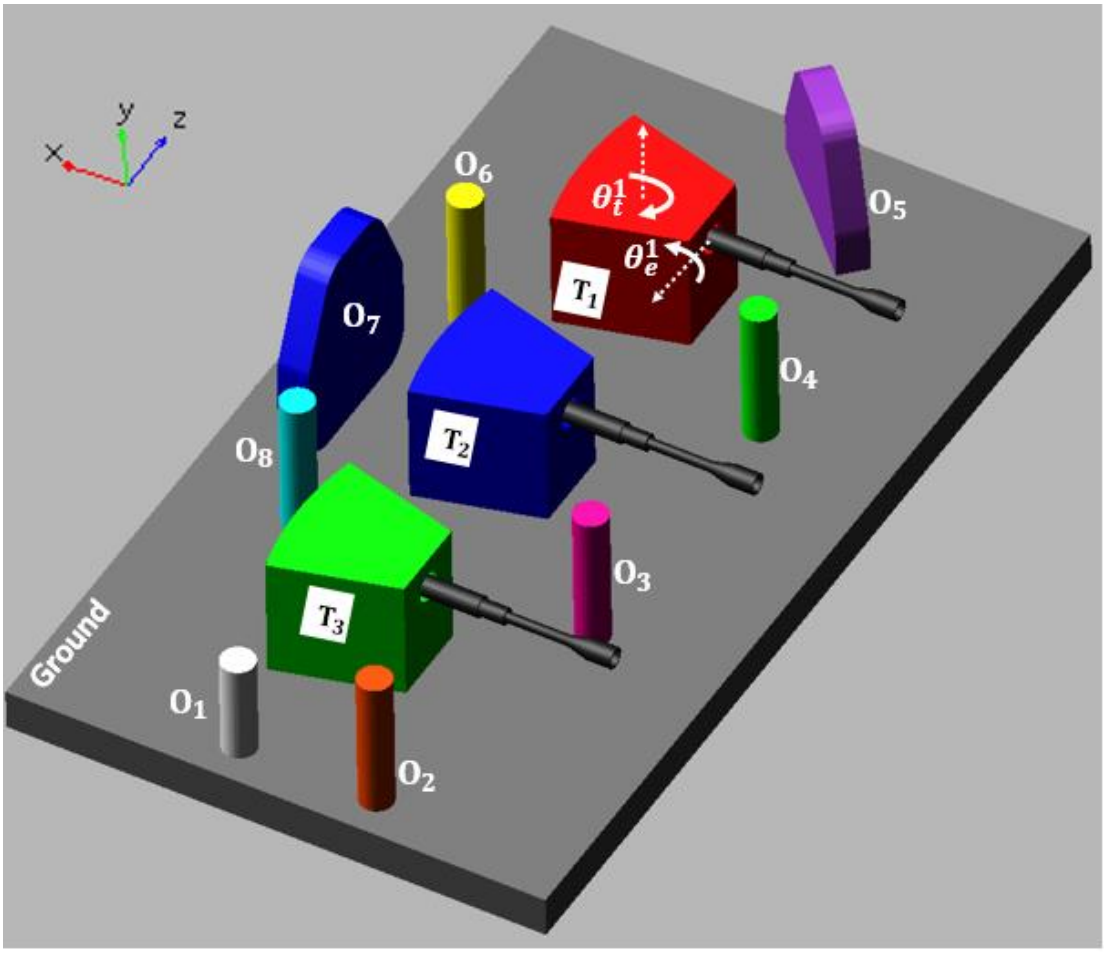

Figure 3.32. The simulation model of triple-turret system with stationary obstacles

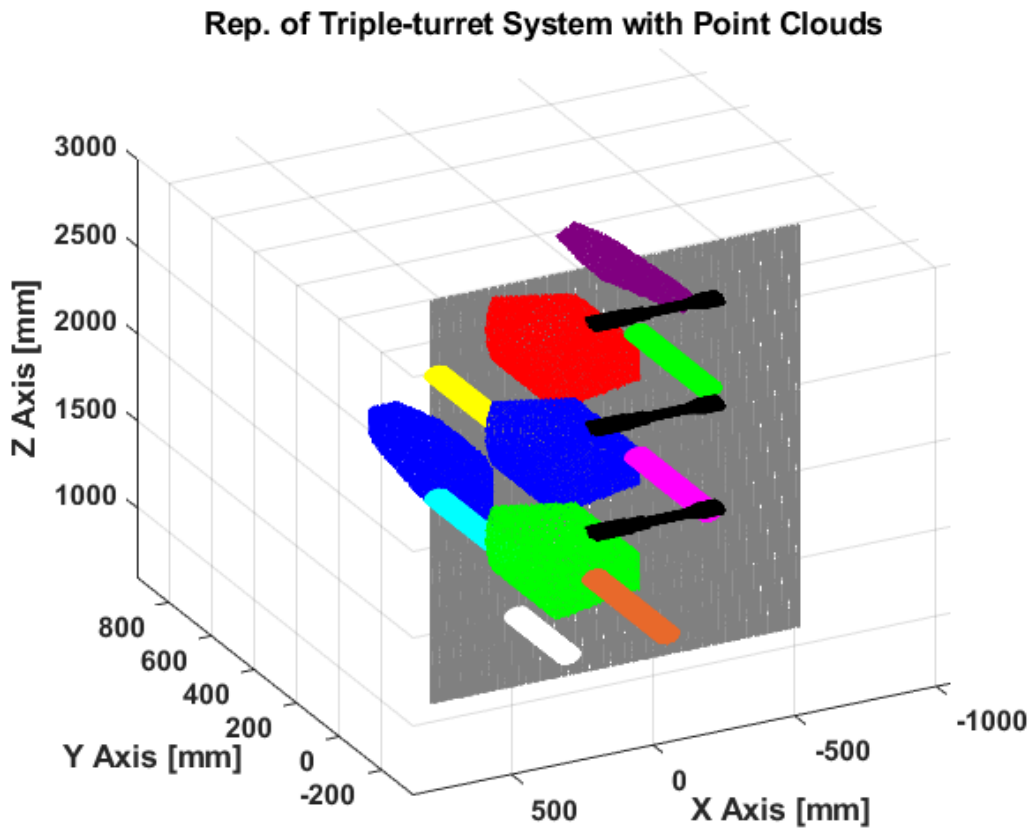

Figure 3.33. The point cloud model of triple-turret system with stationary obstacles

A turret has 2-DOF, one for traverse and the other for elevation. In this example, there exists 6-DOF because of three turrets in the common environment. Also, there are eight stationary obstacles around these three turrets. The positions of the stationary obstacle are adjusted so that the turrets can make a full rotation with the appropriate elevation angle on the traverse axis. The turrets are identical and their height is 305 mm. The heights of identical stationary obstacles of O2, O3, O4, O6, and O8 350 mm and the heights of obstacles of O1, O5 and O7 are 250 mm, 330 mm and 485 mm, respectively. Therefore, stationary obstacles do not limit the operation of the traverse axes (θ_t^1 , $\theta_t^2 and\ \theta_t^3$) when the ascending axes are brought to a suitable angle. However, in case of θ_e^1, θ_e^2 and θ_e^3 are set to zero, then turrets cannot rotate fully on traverse because of stationary obstacles and other turret. The fix 4-D

configuration spaces of the turret pairs 1-2 and 2-3 of the triple-turret system are obtained by the method mentioned in the section 2. In this 4-D C-Space, the elevation axes (θ_e^1, θ_e^2 *and* θ_e^3) start from - 10°, end at 60° with a step size of 5°. The traverse axes (θ_t^1, θ_t^2 *and* θ_t^3) start from 0°, end at 355° with a step size of 5°. In the method of path planning on the C-Space, the grid number of C-Space has great effect on the algorithm itself. If the grid is too big, the precision of planning will decrease. If the grid is small, the calculation payload will increase. A reasonable grid decomposition should be based on some optimum criterion [54]. A 4-D C-Space can be illustrated by a certain number of 3-D configuration spaces. For the example of triple turret system, fifteen different 3-D C-Spaces can be obtained for each value of elevation axis of turret-2 (θ_e^2) for 1-2 turret pair and for each value of elevation axis of turret-3 (θ_e^3) for 2-3 turret pair. The 3-D representations of the 4-D fix C-Space of 1-2 turret pair and 2-3 turret pair are given in Figure 3.34 and Figure 3.35, respectively.

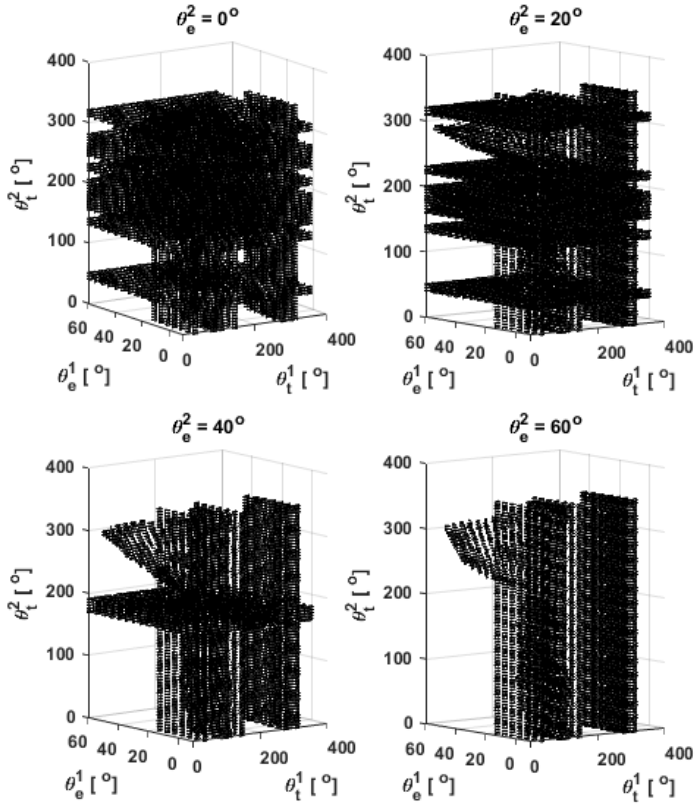

Figure 3.34. 3-D C-Spaces of 1-2 turret pair of the triple-turret system

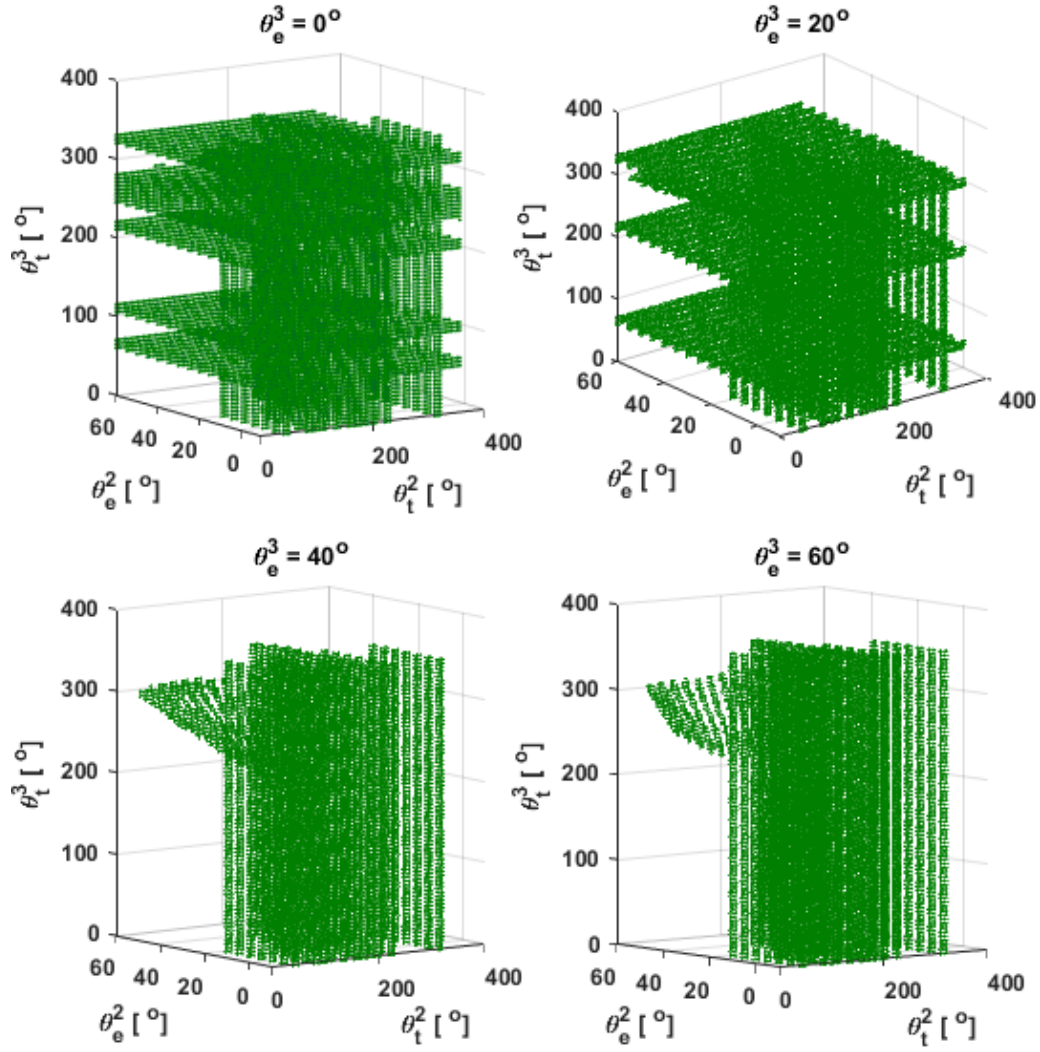

Figure 3.35. 3-D C-Spaces of 2-3 turret pair of the triple-turret system

For aligned multiple-turret systems, it is aimed to make motion planning by choosing two adjacent turrets from all existing turrets. Therefore, for triple-turret system there are 4 different options in total. It can be examined these 4 options under the title of the uppersequence as given in Table 3.4. For example, when 1st uppersequence is used, first 1st and 2nd turrets are driven simultaneously. After they arrived to their target positions and then 3rd turret is driven to its target. Looking at the first driven turret pair of the 1st uppersequence, it can be seen that the dimension of C-Space is

4-D due to turret pair 1-2. For first driven turret pair, the proposed 4-D path planning algorithm is applied and then for 3rd turret 2-D path planning is adequate. Converging options are still valid. The uppersequence table should be updated if the number of turrets on platform is changed.

Table 3.4 Information about uppersequence

Uppersequence No	First Driven Turret Pair		Second Driven Turret Pair	
	Turret Pair	Dimension of C-Space	Turret Pair	Dimension of C-Space
1	1-2	4-D	3	2-D
2	3	2-D	1-2	4-D
3	2-3	4-D	1	2-D
4	1	2-D	2-3	4-D

3.4.1 Case Study

In order to see the differences between the converging options, a sample path is planned between arbitrary chosen start and target positions which are given in Eq. 35 and 36, respectively.

$$\theta_s = [\,\theta_t^1, \theta_e^1, \theta_t^2, \theta_e^2, \theta_t^3, \theta_e^3\,] = [0°, 0°, 0°, 0°, 0°, 0°] \tag{3.25}$$

$$\theta_t = [\,\theta_t^1, \theta_e^1, \theta_t^2, \theta_e^2, \theta_t^3, \theta_e^3] = [60°, 40°, 240°, 10°, 240°, 0°] \tag{3.26}$$

3.4.1.1 Result for Fast Converging Option

When planning a path with fast converging option using the starting and target positions given in the Eqs. (3.25) and (3.26), the results are obtained as in Figure 3.36. The first successful paths of both driven turret pair were found in 0.05 seconds in the first option of the 1st sequence of 1st uppersequence.

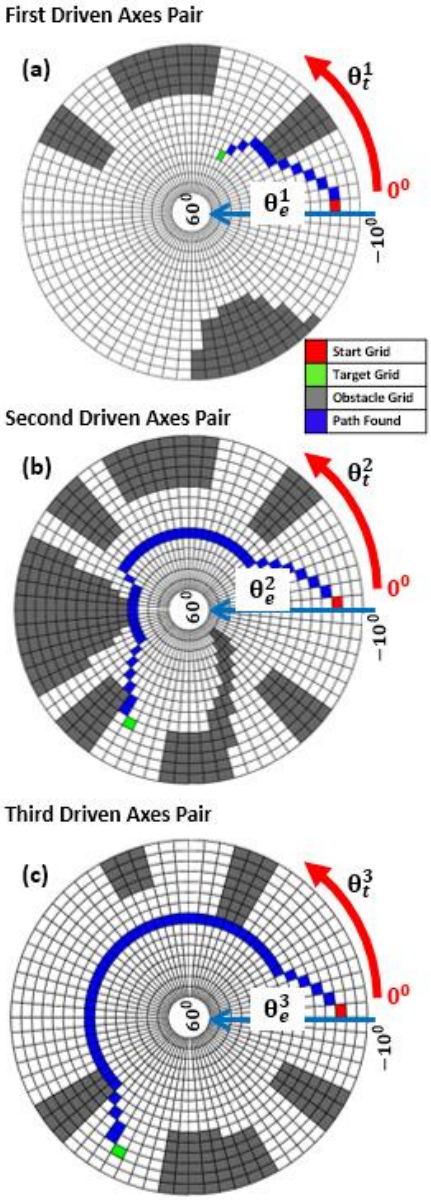

Figure 3.36. A sample path planning using fast converging option (a) first driven axes pair, (b) second driven axes pair, (c) third driven axes pair

After obtaining the variations of the axis positions relative to each other, their variations over time can be achieved as shown in Figure 3.37 using section 3.1.2. In order to drive triple-turret system, the absolute changes of positions are converted into incremental position changes by resetting starting positions as zero.

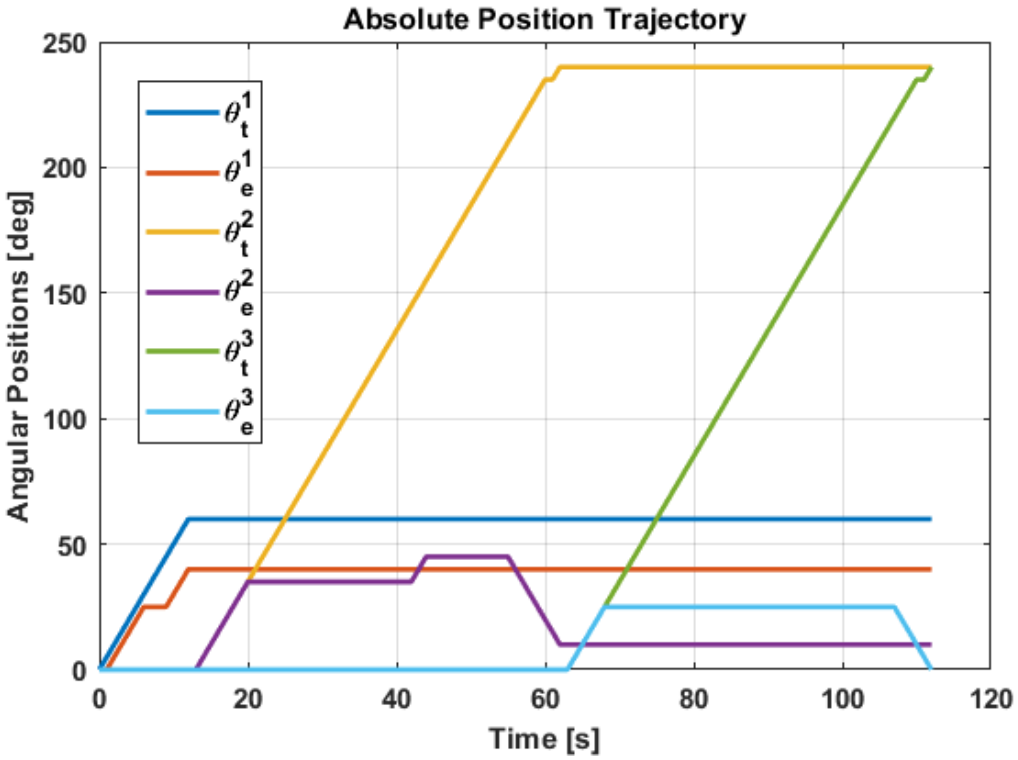

Figure 3.37. The change of position profiles concerning path planning result of fast converging option

The traverse and elevation motion profiles should be created according to angular speed, acceleration or time requirements after obtaining grid changes. If the angular velocities of the axes are set to 5°/s, the time required to pass one grid is 1 second. In this case, the system can reach the target position from the specified start position in 113 seconds. The turrets will reach the target position without any collision if the system performs simultaneous traverse and elevation rotations as shown in Figure 3.37. Once the motion profiles are obtained, the rotations of the triple-turret system were simulated in the model. As a result of this simulation, the changes of minimum clearances between bodies are obtained as shown in Figure 3.38. The abbreviation "Ba", "Hu" and "O" represent Barrels, Hulls and obstacles, respectively. The minimum clearance is between barrel of 1st turret (Ba^1) and fourth obstacle (O_4) which is 6.7 mm and no collision occurs as expected. The safe clearance value can be defined by safe distance between point clouds, δ during obtaining C-Space.

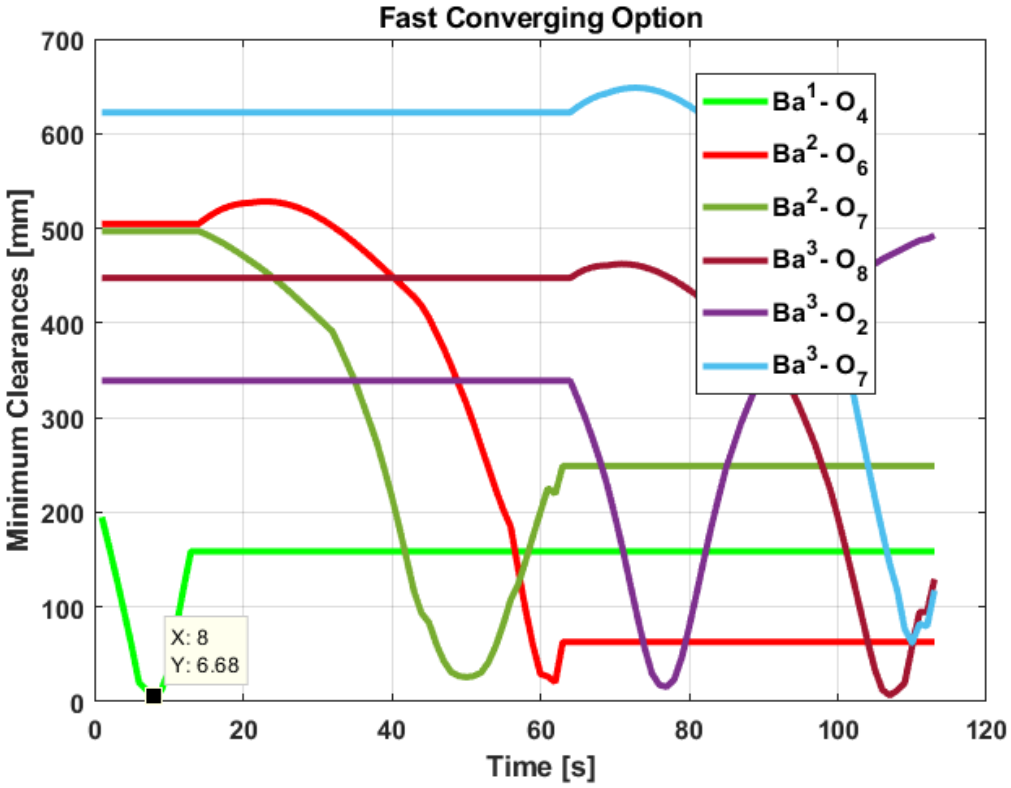

Figure 3.38. The change of minimum clearances between bodies in fast converging option

3.4.1.2 Result for Medium Converging Option

When planning a path with medium converging option using the starting and target positions given in the Eqs. (3.25) and (3.26), the results are obtained as in Figure 3.39. The successful paths of three driven axes for medium converging option were found in 0.45 seconds in the second option of the 1st sequence of 1st uppersequence.

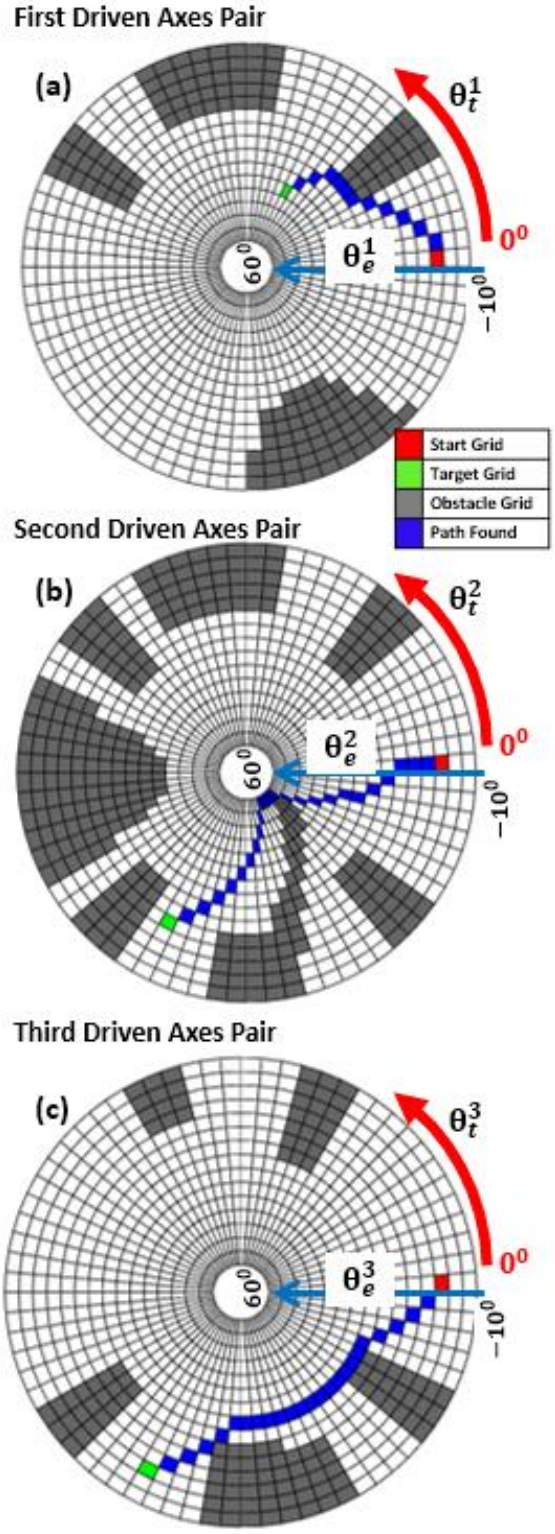

Figure 3.39. A sample path planning using medium converging option (a) first driven axes pair, (b) second driven axes pair, (c) third driven axes pair

After obtaining the variations of the axis positions relative to each other, their variations over time can be achieved as shown in Figure 3.40 using section 3.1.2. In order to drive triple-turret system, the absolute changes of positions are converted into incremental position changes by resetting starting positions as zero. Because absolute position changes are obtained by taking 360 modes of fully rotatable axes, it is not suitable for driving the turrets.

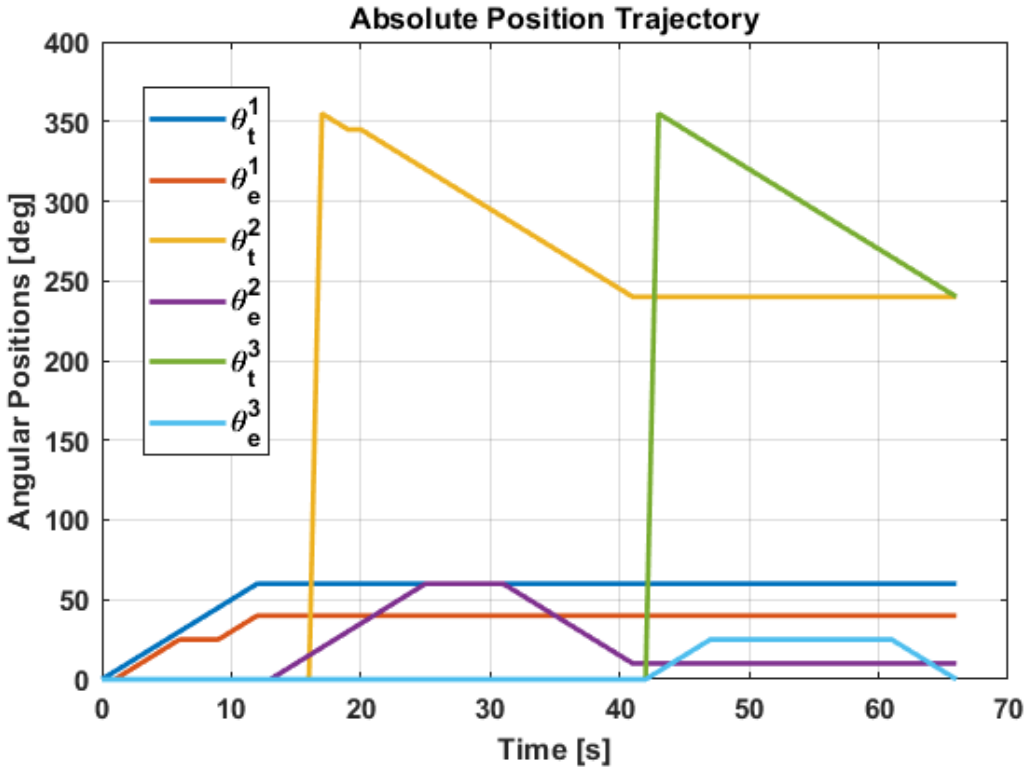

Figure 3.40. The change of position profiles concerning path planning result of medium converging option

If the angular velocities of the axes are set to 5°/s, the time required to pass one grid is 1 second. In this case, the system can reach the target position from the specified start position in 65 seconds. The turrets will reach the target position without any collision if the system performs simultaneous traverse and elevation rotations as shown in Figure 3.40. Once the motion profiles are obtained, the rotations of the triple-turret system were simulated in the model. As a result of this simulation, the

changes of minimum clearances between bodies are obtained as shown in Figure 3.41. The minimum clearance between bodies is greater 6.5 mm and no collision occurs as expected.

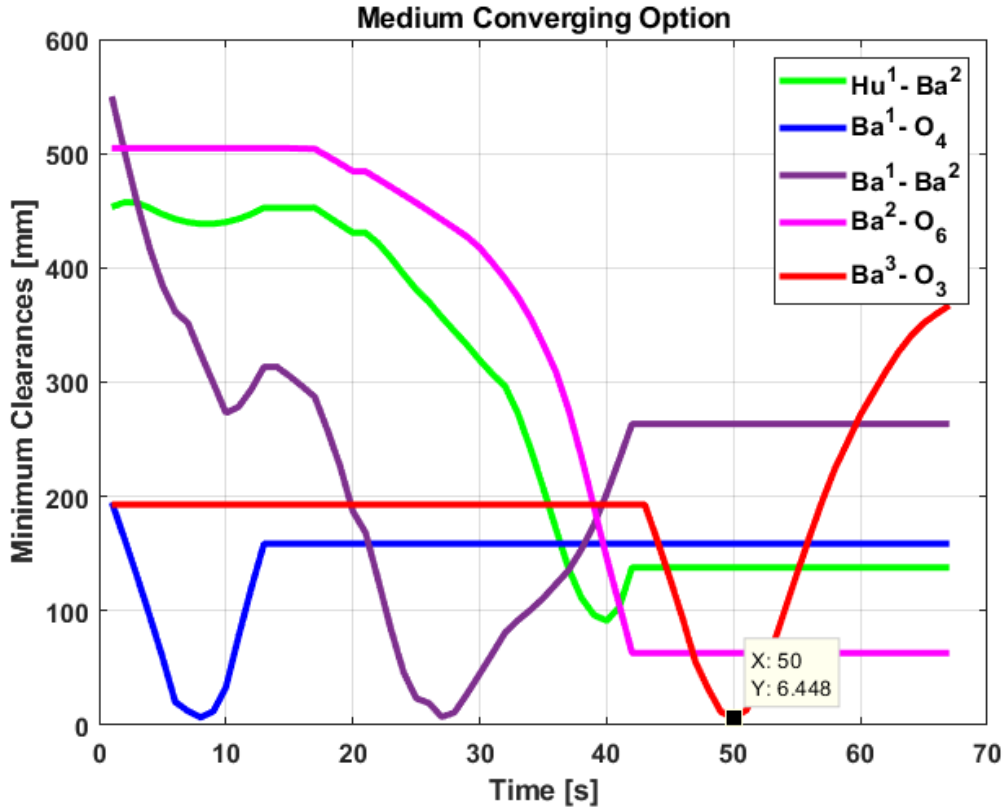

Figure 3.41. The change of minimum clearances between bodies in medium converging option

3.4.1.3 Result for Optimum Converging Option

When planning a path with optimum converging option using the starting and target positions given in the Eqs. (3.25) and (3.26), the results are obtained as in Figure 3.42. The successful path for optimum converging option was found in 27.6 seconds in the first option of the 2nd sequence of 1st uppersequence. Since in this optimum converging option all sequences and their options are scanned, the time required to find the optimum path is quite high as expected.

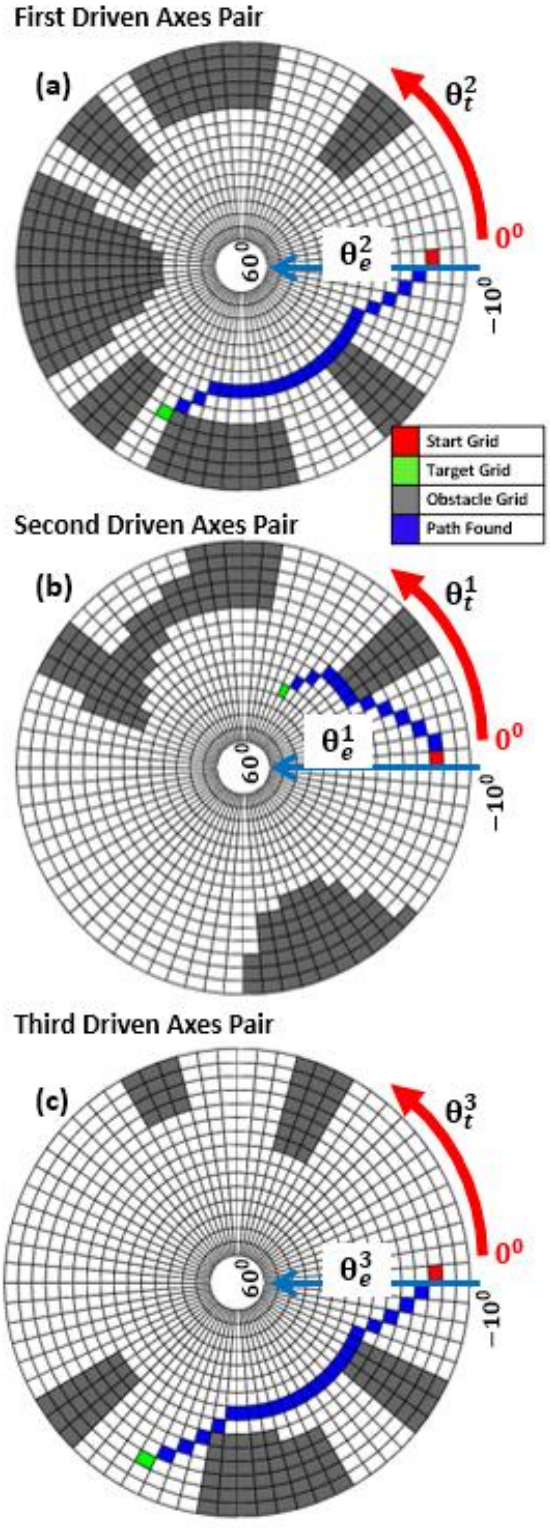

Figure 3.42. A sample path planning using optimum converging option (a) first driven axes pair, (b) second driven axes pair, (b) third driven axes pair

After obtaining the variations of the axis positions relative to each other, their variations over time can be achieved as shown in Figure 3.43 using section 3.1.2. In order to drive triple-turret system, the absolute changes of positions are converted into incremental position changes by resetting starting positions as zero.

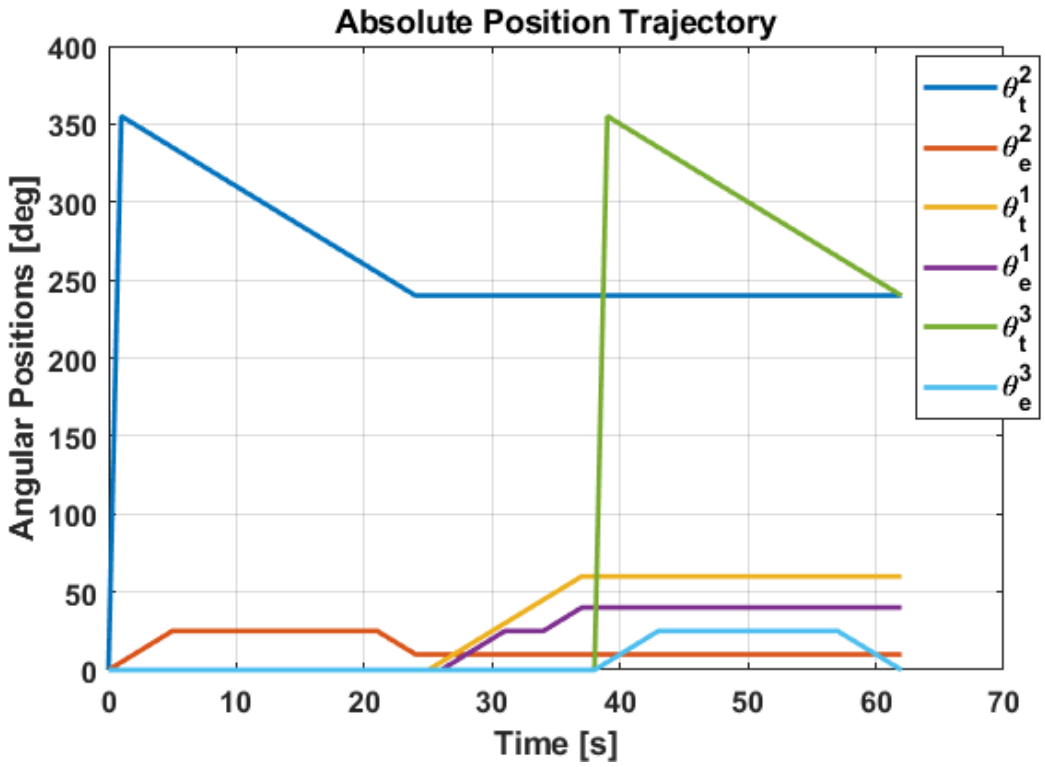

Figure 3.43. The change of position profiles concerning path planning result of optimum converging option

If the angular velocities of the axes are set to 5°/s, the time required to pass one grid is 1 second. In this case, the system can reach the target position from the specified start position in 63 seconds. The turrets will reach the target position without any collision if the system performs simultaneous traverse and elevation rotations as shown in Figure 3.43. Once the motion profiles are obtained, the rotations of the triple-turret system were simulated in the simulation model. As a result of this simulation, the changes of minimum clearances between bodies are obtained as

shown in Figure 3.44. The minimum clearance between bodies is greater 5.9 mm and no collision occurs as expected.

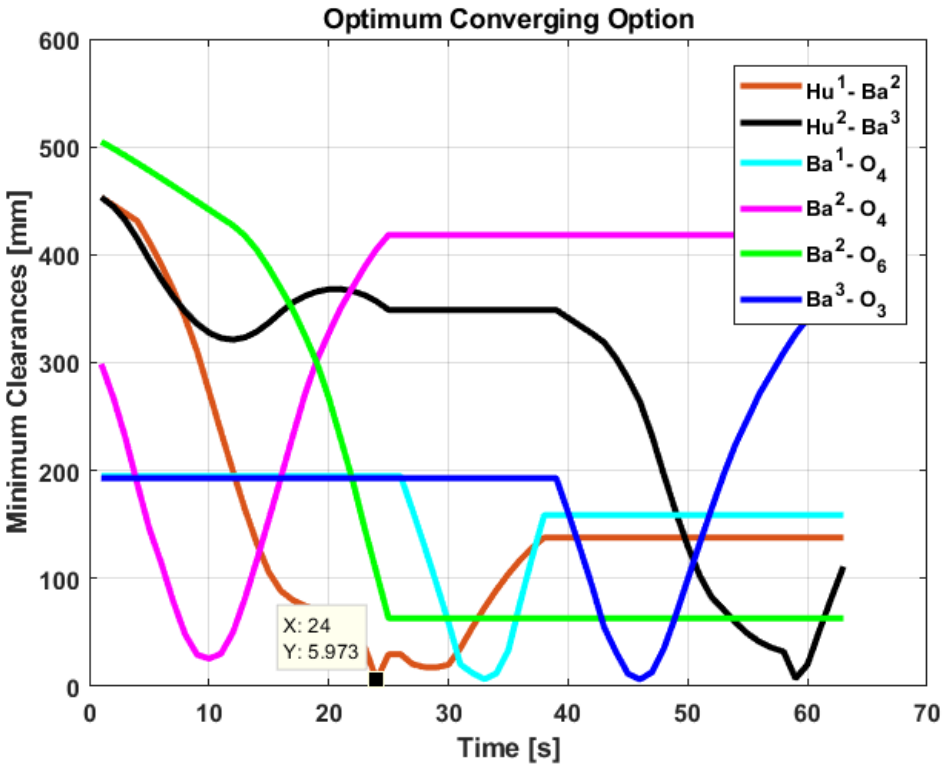

Figure 3.44. The change of minimum clearances between bodies in optimum converging option

3.5 Conclusion

The proposed algorithm attempts to search a path on three different types of C-Spaces named as rectangular, circular and torus shaped in three converging options which are fast, medium and optimum depending on the application. So that, a collision free motion planning can be carried out for double-turret system operating in a common workspace. Firstly, 4-D configuration space of the double-turret system is obtained by using the method of intersection of point clouds where the bodies in the system are meshed and converted into points. As C-Spaces are obtained with

point clouds, it is easier to obtain the configuration space of any system at any dimension with no restrictions on the shape of the system parts. This shows that this proposed algorithm can be applied to all kinds of systems, not only horizontally articulated systems which are generally used in former studies. Then, with the help of the proposed algorithm, 4-D path planning problem was realized as 2-D + 2-D by using six sequences and their options. Thanks to random simulations, the sequence and the options of these sequences are provided in an appropriate order and also proposed algorithm is tried on various input/output combinations. A sample path planning was made to examine the performance of the algorithm and thus the converging options. The results obtained for three different converging options were simulated on the model of the double-turret system and it was observed that there was no collision between any bodies.

CHAPTER 4

COLLISION AVOIDANCE FOR MULTIPLE-TURRET SYSTEM

With the development of robotic systems, most military systems have now become remotely controllable, and this has significantly reduced the need for military personnel [7]. As the number of remote-controlled systems on military vehicles increase, autonomous driving of the systems and the use of collision avoidance algorithms are inevitable. Although there are no works directly focusing on the path planning and collision avoidance of turrets in the literature, algorithms for the braking and overtaking needs of similar robotic systems, Unmanned Aerial Vehicles and autonomous vehicles can be guiding. Navigation can be examined under two different headings; one is global path planning and the other is local collision avoidance. Global path planning algorithms create a group of waypoints, from a start position to a target position, passing through obstacles in a working space and configuration space, while local collision avoidance takes a given waypoint assignment as a local goal to avoid obstacles [21, 22]. Path planning has pulled in a great deal of consideration since the 1970s [23]. Different algorithms have been accounted for over four decades. These algorithms incorporate potential field method [24], heuristic search method [25] and graph technique [26].

Since it is important for missile system launchers to move from one location to another rather than a trajectory, global path planning algorithms are used, while local collision avoidance algorithms are used for the turrets of armored ground vehicles. For this reason, real-time collision avoidance algorithms are especially focused on.

Velocity Obstacle, Set-Based Guidance, Vector Field Histogram and Dynamic Window Approach are all local methods. Borenstein and Koren recommended in 1991 a technique for collision avoidance for versatile robots called Vector Field

Histogram (VFH) [27]. It was introduced as a solution on the basic issues of the Potential Field strategies, introduced by Khatib in 1985 [28]. In 1997, Fox, Burgard and Thrun introduced the Dynamic Window Approach (DWA); another technique for collision avoidance for versatile robots [29]. The Dynamic Window is a diminished speed space containing only the speeds reachable inside a short measure of time and that guarantees a way where the robot is capable to stop securely. An objective function is controlled by consolidating the speed of the robot, the advancement towards the objective and the separation between the robot and the hindrance. It brings about a harmony between keeping a rapid motion towards the objective and performing sly moves to dodge deterrents. The translational and rotational speeds are picked by maximizing the objective function. Since most robots have some sort of limitations on speed and acceleration, DWA is particularly decent since it handles this automatically [30]. Velocity Obstacle (VO) was first published by Fiorini and Shiller in 1998 [31]. VO is a movement arranging strategy for robots in extreme situations. It aims to find all speeds for which the robot will crash into a deterrent sooner or later. These speeds will make confined regions in the speed space molded like cones, one for each obstacle.

Potential filed method, reactive methods, deformable virtual zones, velocity potential field and acceleration velocity obstacle can be cited as examples of local collision avoidance algorithms where intervention to speed and even torque loops is inevitable for implementation [32-37].

These proposed collision avoidance algorithms for autonomous vehicles [38-42] are not easy to apply directly to turrets with two different independent axes, as these algorithms perform their maneuvers by braking or steering [43-46].

Most of the algorithms mentioned above are concerned with taking systems from a starting point to a target point collision free in dynamic or static environments. Therefore, there is always a position to which the system is directed. In addition, the implementation of these algorithms requires frequent intervention in the control loops of the systems, especially the torque loop, which is the innermost loop. For

this reason, it is not possible to add these algorithms as add-on to ready-to-use systems.

Motivated by the above problems, it is focused on the new collision avoidance algorithm named as End Damping Algorithm which provides systems to be driven both in speed and position control modes more efficiently and safely.

4.1 End Damping Algorithm

Gun turret system in Figure 4.1 has 2 degrees of freedom, one for elevation and the other for traverse axis. The gun turret system is expected to provide flexibility and help the user without colliding different systems on the vehicle.

Figure 4.1. Stabilized remote controlled gun turret system

The system architecture with the End Damping algorithm can be shown as follows. The system model consists of the position loop (1), the end damping algorithm (2)

and the closed box (ready-to-use) system (3) itself. The closed box system can be any system for which speed is commanded. Turret systems are generally used in speed mode, but position loop is used when bringing the turret to home position or video tracing issues. A collision avoidance algorithm must be used to avoid colliding obstacles while the system is controlling the position. For this reason, the end-control-damping algorithm is placed between the position controller and the rest of the system as seen in Figure 4.2. The commands given to the axes are used as speed command if position control is disabled, and position command if enabled. Proposed algorithm works as collision avoidance when the system is working in speed mode and as path planning in position control mode. While operating in both modes, motion takes place within the speed, acceleration and jerk limits.

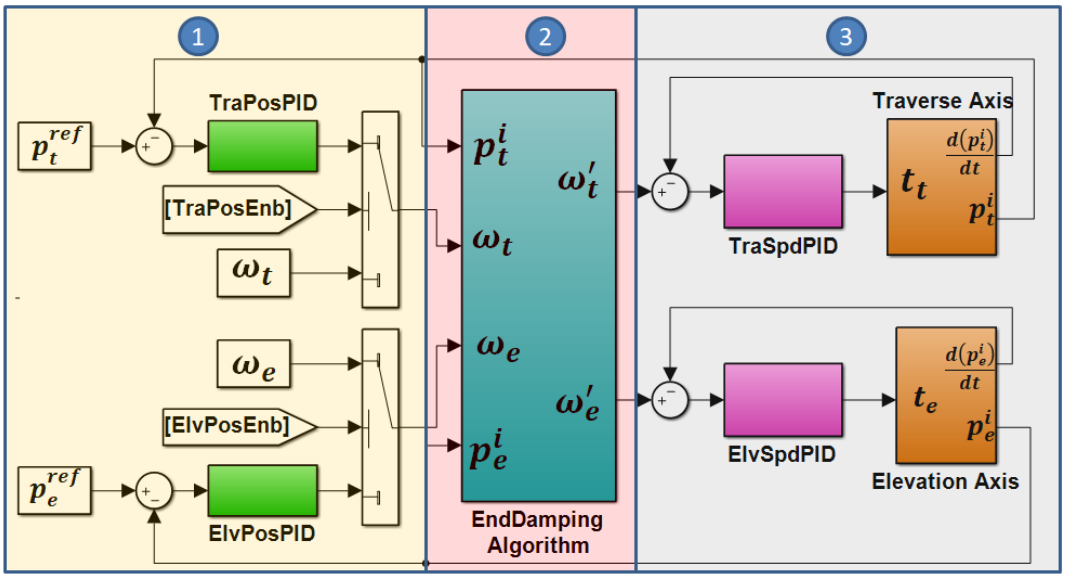

Figure 4.2. System architecture

Stabilization and normal operating modes are available according to the requirements. Most of the system's working life is in speed control mode. Therefore, collision avoidance algorithms are needed for these systems rather than path planning. It is possible to show the 2 degree of freedom working space as a 2-dimensional configuration space (C-space) as follows.

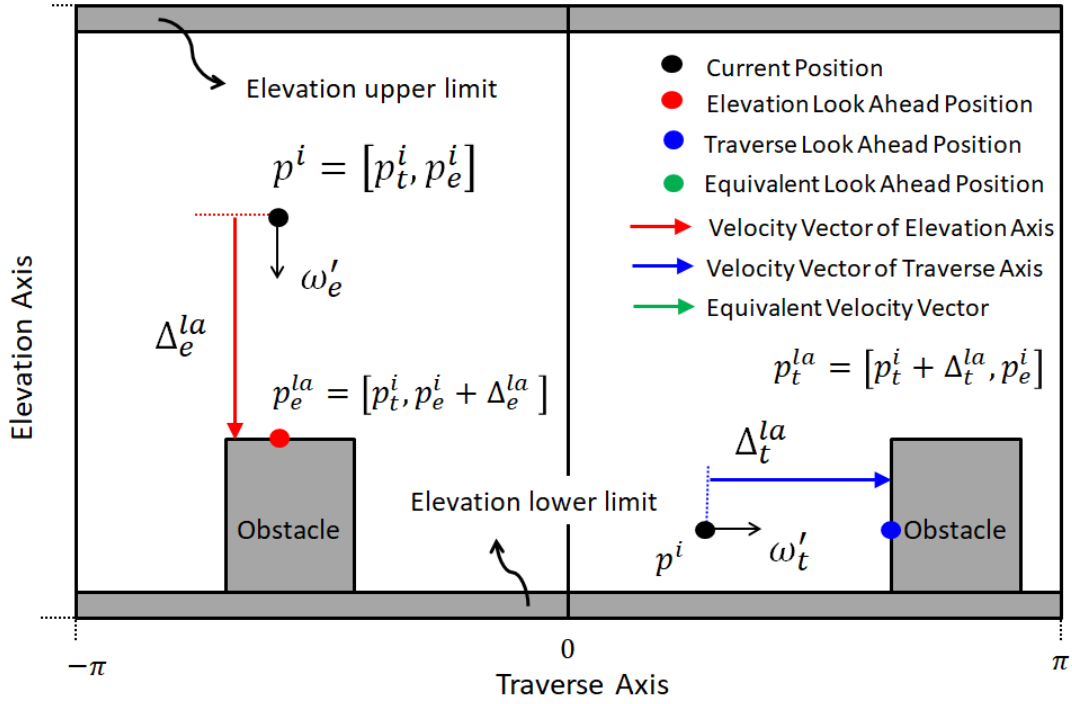

Figure 4.3. Sample C-space of gun turret system

As can be seen from configuration space, while there is an upper and lower limit on the elevation axis, the traverse axis can take a full turn ($n \times 2\pi$). The obstacles in the C-space are expressed as Cartesian based rectangles so that they can be expressed with a small number of parameters. Since the C-space represents the turret's movements on the vehicle, all obstacles are adhered to the upper/lower elevation limits. All obstacle are adjacent to bottom and top which named as lower-hang and upper-hang. In such systems, there is not much opposition to these situations.

The main purpose of this study is that the current turret position, which is shown as black dot in Figure 4.3, never enters the lower/upper limits of elevation axis and the obstacles expressed as Cartesian and to overcome the obstacles. But at the same time, while doing this, speed, acceleration and jerk limits specified in the performance requirements of the system must be taken into account. Another critical point is that while the system is avoiding the obstacle or slowing down in order not to interfere with the obstacle, it should listen to the new commands from the user instantly and

apply the new command if the incoming commands do not pose any danger to the system. It is aimed not to interfere with the user as much as possible while ensuring the movement of the system without collision. Thus, the user will not have the feeling of losing control. This can be done by listening to new commands from the user at every stage of the algorithm.

We can show the instantaneous location of the system as a point in the C-space as in Eq. (4.1).

$$p^i = [p_t^i \quad , \quad p_e^i] \tag{4.1}$$

In the C-space, look ahead distances on each axis are calculated as in Eq. (4.2) and (4.3) in order not to collide the upper/lower limits and obstacles. Look ahead distance is the distance the system travels when it starts to slow down until it stops. The function named "StopDis" used below gives the stopping distance of a system whose initial speed and acceleration are given by adhering to the acceleration and jerk limits. The details of the function $StopDis$ are given between Eqs. (4.10) and (4.20).

$$\Delta_e^{la} = StopDis(\omega_e, \alpha_e) + sgn\,(\omega_e) \cdot \delta_e \tag{4.2}$$

$$\Delta_t^{la} = StopDis(\omega_t, \alpha_t) + sgn\,(\omega_t) \cdot \delta_t \tag{4.3}$$

By adding look ahead distances to the current position of the turret, look ahead positions are obtained like Eq. (4.4) and (4.5).

$$p_e^{la} = [p_t^i \quad , \quad p_e^i + \Delta_e^{la}] \tag{4.4}$$

$$p_t^{la} = \left[mod\left(p_t^{int} + 180\,,\, 360\right) - 180 \quad , \quad p_e^i\right] \tag{4.5}$$

where $p_t^{int} = p_t^i + \Delta_t^{la}$

Since the turret can rotate fully on the traverse axis, the position of the turret is always in the range of $\pm 180°$, by taking the mode as in Eq. (4.5). Because the space we show as rectangular C-space is actually circular and $+180°$ and $-180°$ represent the same position.

The red dot shown in Figure 4.3 shows the elevation axis look ahead position and it is updated instantly. When the red dot comes into contact with any obstacle, the system will start to slow down on the elevation axis. The same applies to the traverse axis look ahead position shown as blue dot. Avoiding phase starts when the blue point comes into contact with any obstacle. The axes can be handled separately in this way. However, this process alone is not enough to give the turret collision-free mobility. For this reason, the equivalent look ahead position caused by both axes should also be considered. The equivalent look ahead position is shown with a green dot in Figure 4.4.

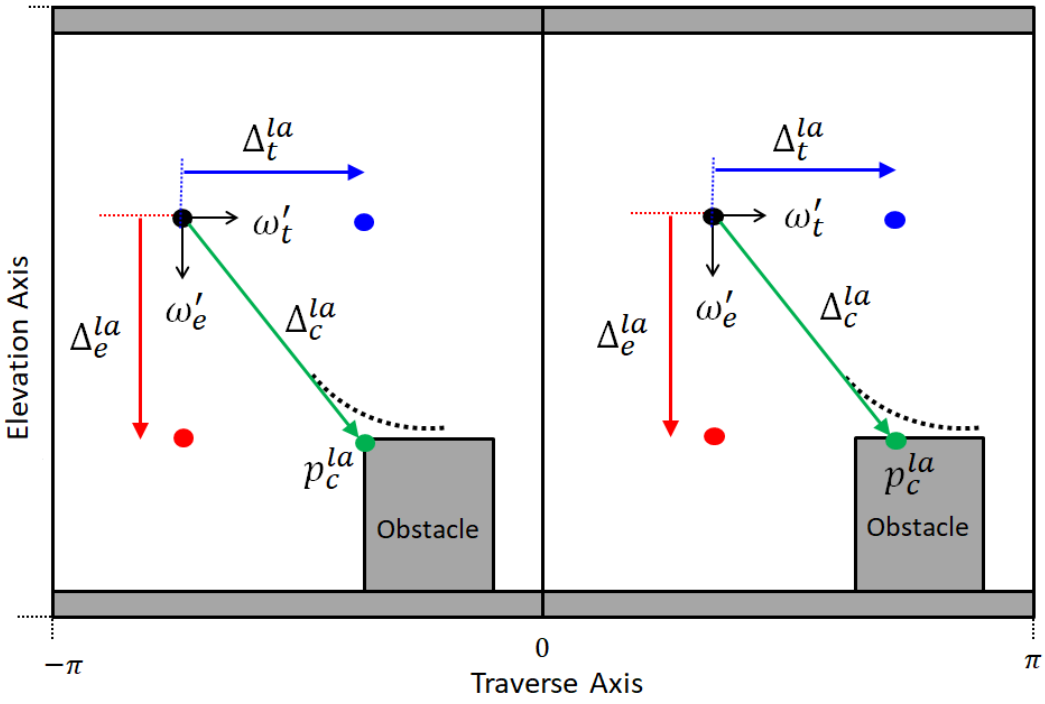

Figure 4.4. Illustrations of the equivalent look ahead position and converging to elevation axis

Normally, if the equivalent look ahead position had not been processed, only the red and blue dots could not see the obstacle in the figure, so the turret could enter the obstacle from the corner. As can be seen in Figure 4.4, when the equivalent look ahead position (green point) touches the corner or above the obstacle, the turret will only slow down on the elevation axis and avoid the obstacle by passing tangentially to the obstacle as seen in Figure 4.5.

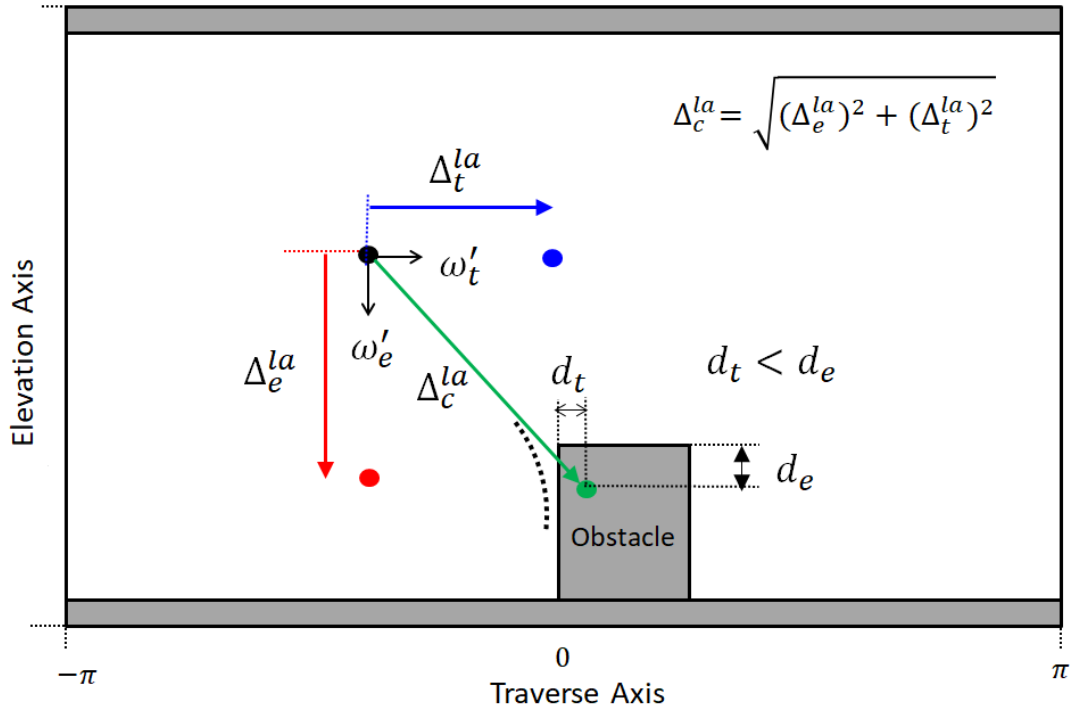

Figure 4.5. Illustrations of the equivalent look ahead position and converging to traverse axis

If the equivalent look ahead position enters the obstacle from the side as shown, the turret will only slow down on the traverse axis and pass tangential to the obstacle. If the turret is still rotated to the right (towards the obstacle), then the turret will first slow down on the traverse axis and then begin to avoid the obstacle by elevating. The axis on which the turret will cross the obstacle is related to which surface the equivalent look ahead position is closer to. If it closes to the side surfaces of the obstacle, deceleration starts on the traverse axis; in the case of closing to upper surface then deceleration occurs on elevation axis.

The turret avoiding a Cartesian based rectangular shaped obstacle as given in Figure 4.6. Thus, when the traverse or equivalent look-ahead position comes into contact with the obstacle, the turret will begin to avoid the obstacle. If the actual equivalent speed of the turret is increased or the maximum rate of deceleration is decreased, the turret will avoid the obstacle with a larger radius. The reason for this is that in this case the look ahead distance will be greater, and the presence of the obstacle will be detected earlier.

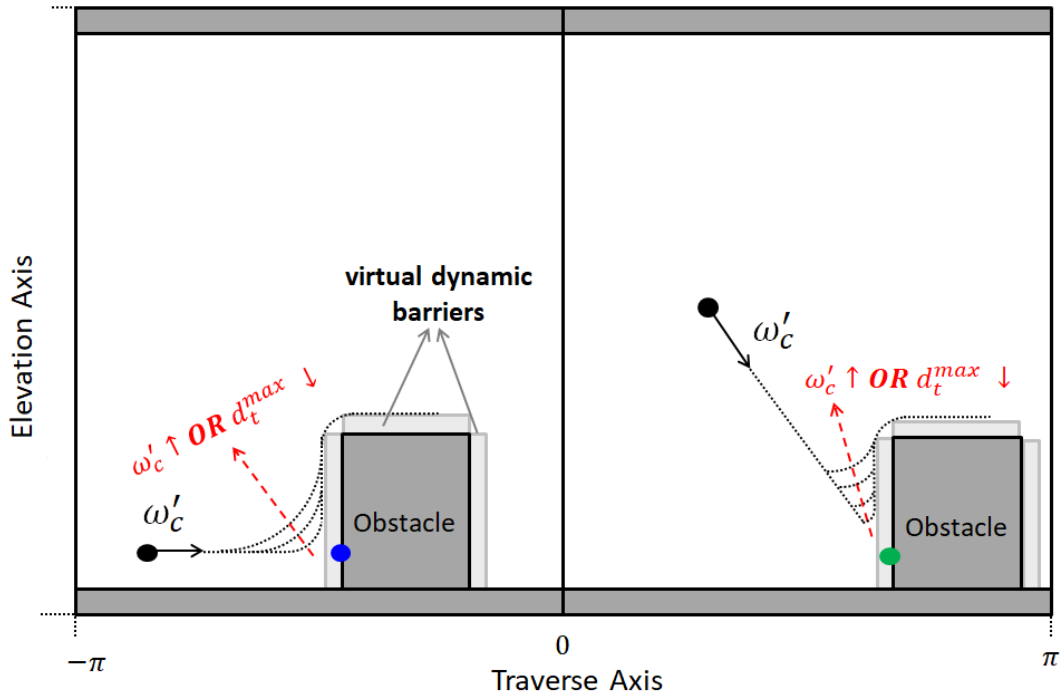

Figure 4.6. Avoiding a rectangular shaped obstacle and illustration of the virtual barriers

An artificial elevation speed command is required for the black dot, which expresses the current position of the turret in the C-space, to avoid the obstacles. This speed command should be calculated as in Eq. (4.6) in order to prevent serious overshoots of the system. We can find the speed value of stopping distance of δ_e and initial acceleration is assumed to be zero as follows.

$$\omega_a = (StopDis)_{\delta_e}^{-1} \quad for\ a_a = 0 \tag{4.6}$$

When the turret crosses the obstacles by using the avoiding speed in Eq. (4.6), the actual position trajectory will form just around the virtual dynamic barriers without moving too far from the obstacle as in Figure 4.6.

4.1.1 Worst Cases

Handling all worst scenarios as possible is critical for the robustness of the algorithm. We can examine the worst scenarios that may occur in such systems under 4 subtitles. The precautions taken for these subtitles will play a role in shaping the frame of the algorithm.

4.1.1.1 Narrow width obstacles

It is stated that the equivalent look ahead position should also be used so that the turret does not enter the obstacle from the corner. However, just taking this action does not guarantee that the turret will not collide with all Cartesian based obstacles. The distance traveled by the turret until it slows down and stops on the traverse axis at maximum speed expresses as maximum look ahead distance. If the width of one of the obstacles is less than this distance then the collision will occur.

As can be seen on the left side of Figure 4.7, the equivalent look-ahead position sees the obstacle, but on the right side, the collision will occur because the turret cannot see the obstacle anymore. If the width of the obstacle is greater than or equal to the value Δ_t^{la}, it would not be possible for the turret to collide the obstacle. However, considering the narrow width obstacles, it is obvious that only one equivalent look ahead position will not be sufficient. The number of equivalent look ahead position can be calculated as follows.

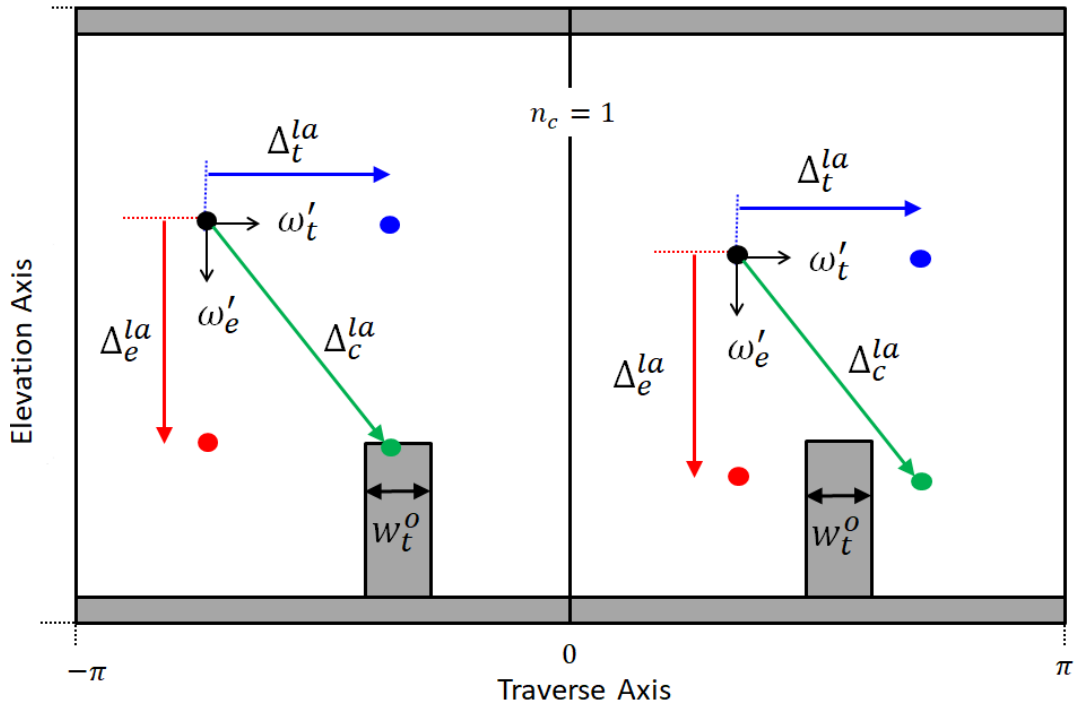

Figure 4.7. The relation between Δ_e^{la} and w_t^o

$$n_c = \left\lceil \frac{\max(\Delta_t^{la})}{w_t^o} \right\rceil \tag{4.7}$$

where

$$\max(\Delta_t^{la}) = StopDis\left\{\left(\omega_{max} - \frac{a_{max}^2}{2j_{max}}\right), a_{max}\right\}$$

Equivalent ahead positions can be obtained as follows.

$$(p_c^{la})_{n=1\ldots..n_c} = \left[mod\left((p_c^{int})_{n=1\ldots..n_c} + 180\,,\, 360\right) - 180 \quad , \quad p_e^i\right] \tag{4.8}$$

where

$$(p_c^{int})_{n=1\ldots..n_c} = p_t^i + \Delta_t^{la} \cdot \left(\frac{n_c - n + 1}{n_c}\right)$$

In Figure 4.7, the obstacle cannot be seen by a single equivalent ahead vector, now it can be seen by using more than one equivalent look ahead position in Figure 4.8 and no-collision is guaranteed.

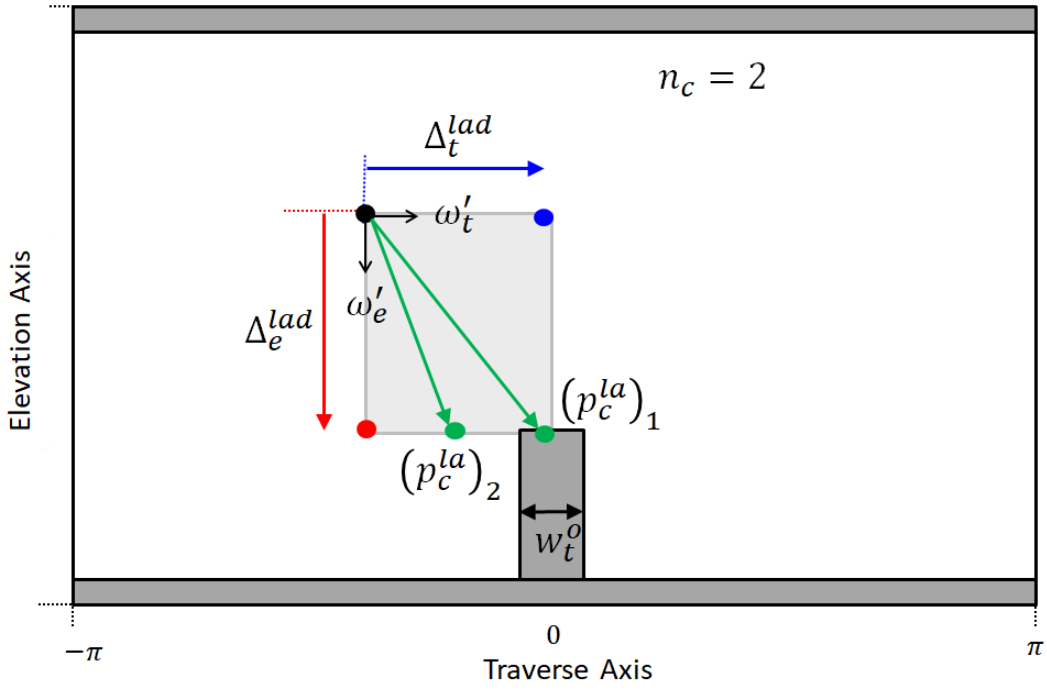

Figure 4.8. Illustration of 2 equivalent look ahead vectors

4.1.1.2 Adjoining obstacles

The Cartesian expression of the obstacles in the C-space causes the growth of small obstacles, which causes a decrease in free space. Therefore, non-Cartesian obstacles in C-space can be defined as side-by-side Cartesian obstacles to take up less space.

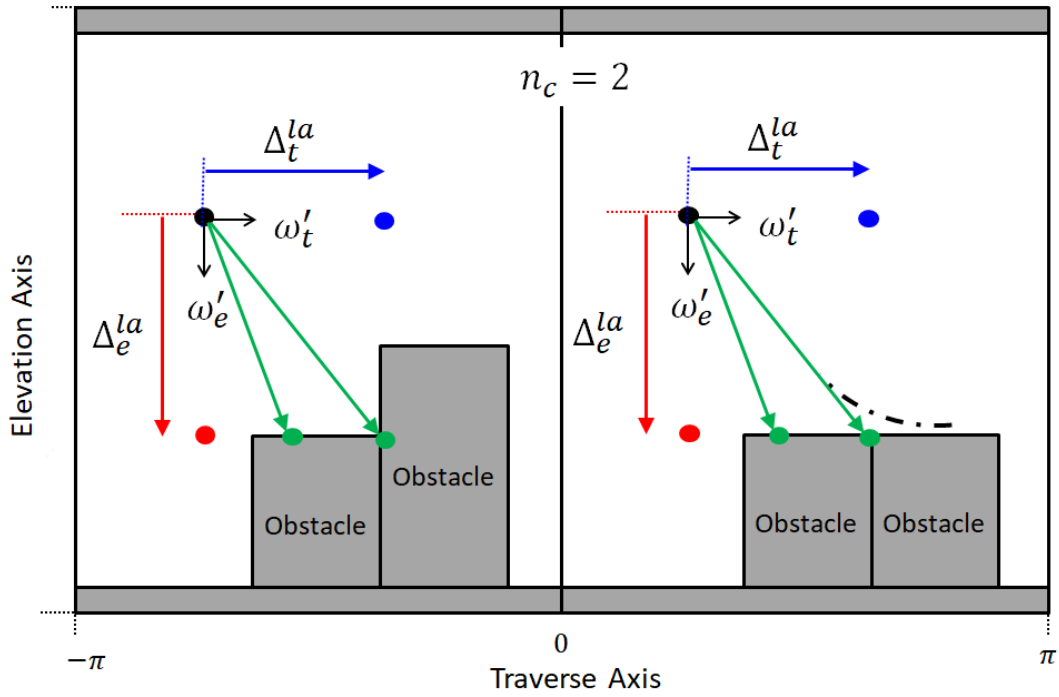

Figure 4.9. Illustration of two adjoining obstacles

A few precautions have to be taken so that the obstacles next to each other would not cause problems for the equivalent ahead positions. In the case shown on the left of Figure 4.9, both look ahead positions interfered with the obstacles. Since one of them touches the upper surface of the obstacle, deceleration is desired on the traverse axis since the other one touches the side surface of another obstacle. In this case, the system will start to slow down in both axes. To the right of the same figure, there are two side by side obstacles at the same height. This situation can be experienced when a large Cartesian based obstacle is found in 180° transition. Otherwise, a single obstacle could be expressed instead of 2 obstacles anyway. In this case, the turret will slow down on the rising axis and will be tangent to the obstacle.

4.1.1.3 Low speed motion

While the turret is trying to avoid the obstacle, it passes tangential to the traverse axes and then moves to a tangent position to the upper surfaces. Precautions should be taken in order for the system to experience a problem similar to the local minimum at the moment of full transition in low speed movements.

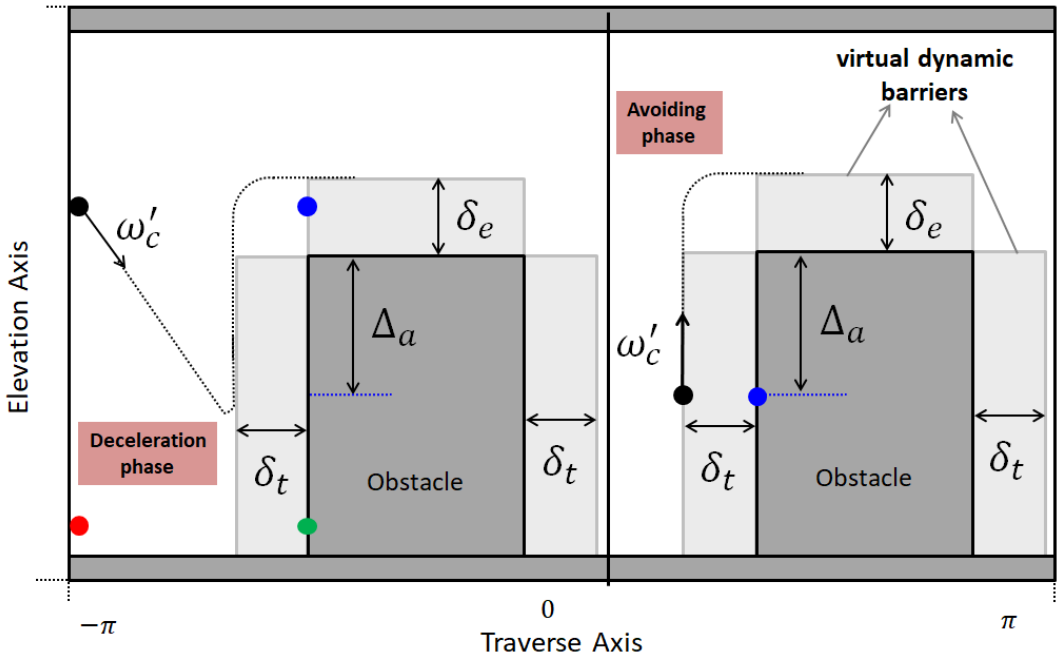

Figure 4.10. Illustration of two adjoining obstacles

As shown on the left side of Figure 4.10, when the turret starts the deceleration phase, there is a distance Δ_a that it can avoid. If this distance Δ_a is less than height of dynamic barrier of elevation axis (δ_e), the system will experience a local minimum. This situation occurs during low speed movements.

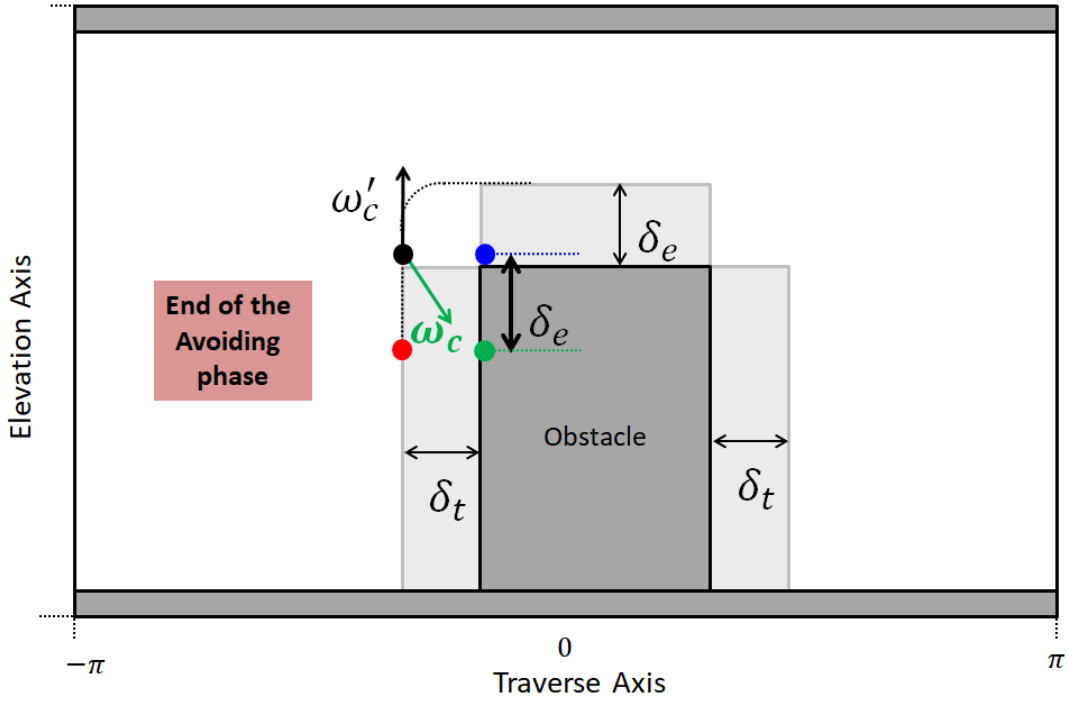

Figure 4.11. End of the avoiding phase

At low speeds, when the turret is at the end of the avoiding phase, local minimum is experienced when the actual speed (ω_c') is upward and the user speed command (ω_c)) is towards the obstacle, as in Figure 4.11. This situation, which occurs especially at low speeds, is solved by updating the selection decision of the axis to be converged.

The symbol d_e is the distance of the equivalent ahead position to the upper surface of the obstacle and d_t is the distance to the side axis to which it is close, shown in Figure 4.5. In the normal case, where $d_e > d_t$, the turret will converge on the traverse axis, otherwise it will converge to the elevation axis. However, since this situation creates a local minimum problem for low speeds, the axis to be converged will be selected according to the condition in Eq. (4.9) in order to avoid local minimum problem. The value ε is so small and approximately can be selected as 10% of δ_e.

$$ConvergTo = \begin{cases} Elevation, if \;\; d_e < d_t + \delta_e + \varepsilon \\ Traverse, if \;\; d_e > d_t + \delta_e + \varepsilon \end{cases} \tag{4.9}$$

4.1.1.4 Emergency case

Some of the obstacles refer to hatches on the vehicle. These can be active or inactive depending on the situation. While the turret is standing exactly where the hatch is, if the hatch is opened, the obstacle will be active, so the current position of the turret will remain inside the obstacle. If we express this situation as an emergency case, then according to the lower-hang or upper-hang state of the obstacle, it will move the current position to the safe zone as in Figure 4.12 by rising or depressing on the elevation axis.

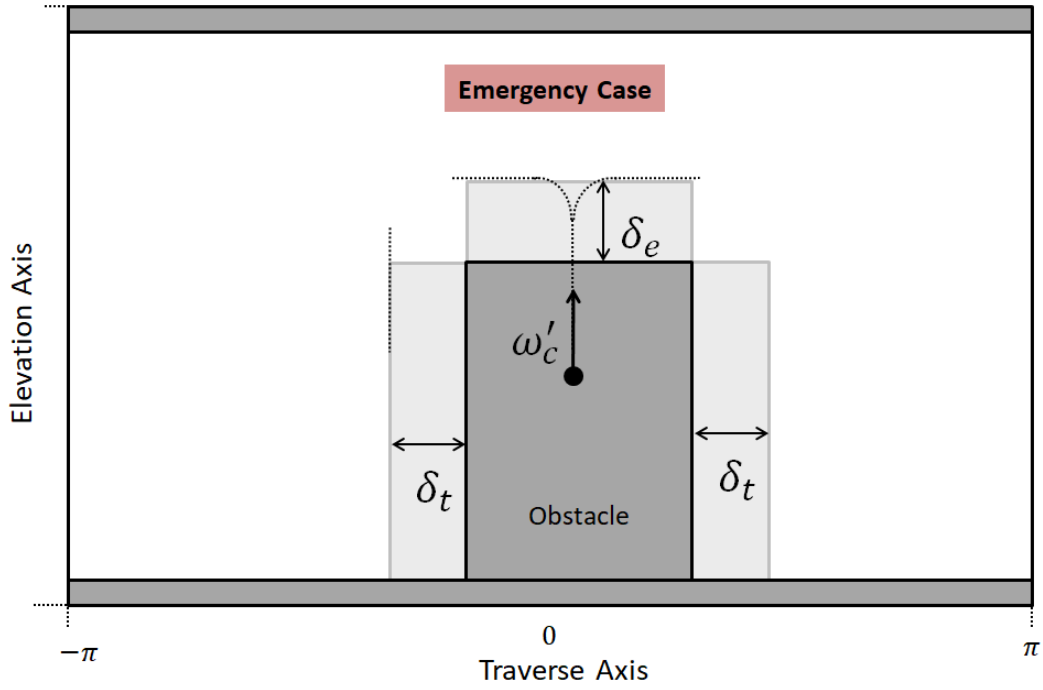

Figure 4.12. Emergency case

4.1.2 Source codes

The main algorithm (Algorithm-4.1) starts by initializing kinematic parameters, coordinates of the obstacles and 4 inputs; ω_t, ω_e, p_t^i, p_e^i which represent traverse and elevation speed commands, instantaneous traverse and elevation positions, respectively. Firstly, the number of equivalent look ahead vectors (n_c) is calculated and according to this number, all look ahead positions ($p_e^{la}, p_e^{la}, (p_c^{la})_{n=1\ldots..n_c}, p^i$) are obtained in Line 1. Then, control point matrices are calculated using the look ahead positions obtained between Lines 2-11. The function named ***CollisionDet*** used in Lines 3, 4, 5, 10 is given in detail in Algorithm-4.2. Converging axis of the system is determined by processing control point matrices between Lines 12-22. The function named ***ConvAxis*** used in Line 19 is given in detail in Algorithm-4.3. The remaining Lines are for determining the action to be taken. Between Lines 23-27, it is checked whether the system is in emergency state or not. Then, using the converging axis and control point matrices, the velocity commands in both axes are shaped. The function named ***SpeedShaper*** used between Lines 26-41 is given in detail in Algorithm-4.4.

Algorithm-4.1: End Damping Algorithm

In: $\omega_t, \omega_e, p_t^i, p_e^i$

Out: ω_t', ω_e'

1 $calculate\ n_c, p_e^{la}, p_e^{la}, (p_c^{la})_{n=1\ldots..n_c}, p^i$

initiate $\omega_{t_{pre}}' = \omega_{e_{pre}}' = a_{t_{pre}}' = a_{e_{pre}}' = 0$

2 **for** $i = 1\ \ to\ \ n_o \rightarrow number\ of\ obstacles$

3 $ElvChk(i) = \boldsymbol{CollisionDet}\left(p_e^{la}, O(i)\right)$

4 $TraChk(i) = \boldsymbol{CollisionDet}\left(p_t^{la}, O(i)\right)$

5 $EmrgChk(i) = \boldsymbol{CollisionDet}\left(p^i, O(i)\right)$

6 **for** $j = 1\ \ to\ \ n_c \rightarrow number\ of\ equ.look\ ahead\ vectors$

7 **if** $(i)\ is\ element\ of\ 'ElvLimits'$

$EquivChk(j,i) = 0; \quad GlobalChk(j,i) = 0;$

else

$EquivChk(j,i) = \boldsymbol{CollisionDet}\left((p_c^{la})_{n=j}, O(i)\right)$

$GlobalChk(j,:) = EquivChk(j,:) - ElvChk(i) - TraChk(i)$

if max$(GlobalChk) == 1$

n = # of the obsts. that equ. look ahead vectors collide with

if $n > 1$ && $case\ of\ different\ obstacles\ have\ different\ converging\ axis$

$ConvOpt = 3;$

else

u = select one of the obst. the combinations collide with

equ = the equi. vectors intersects with obstacle 'u'

n_{equ}= number of equi. vectors intersects with obstacle 'u'

for $y = 1\ to\ \ n_{equ}$

$AxConv(y) = \boldsymbol{ConvAxis}\left((p_c^{la})_{n=equ(y)}, O(u)\right)$

$ConvOpt =$ max$(AxConv)$;

else

$ConvOpt = 0;$

if max$(EmrgChk) == 1$

find the obstacle that current position is inside

for lower-hang obs. $\omega_{emrg} = \omega_a$, otherwise $\omega_{emrg} = -\omega_a$;

$[\omega_t', a_t'] = \boldsymbol{SpeedShaper}\left(0, \omega_{t_{pre}}', a_{t_{pre}}', t_s\right)$

$[\omega_e', a_e'] = \boldsymbol{SpeedShaper}\left(\omega_{emrg}\omega_{e_{pre}}', a_{e_{pre}}', t_s\right)$

else

if max$(TraChk) == 1$ | ConvOpt == 1 | ConvOpt == 3

$[\omega_t', a_t'] = \boldsymbol{SpeedShaper}\left(0\omega_{t_{pre}}', a_{t_{pre}}', t_s\right)$

else

32 $[\omega_t', a_t'] = \boldsymbol{SpeedShaper}\left(\omega_t \omega_{t_{pre}}', a_{t_{pre}}', t_s\right)$

33 **if** $\max(TraChk) == 1$

34 find the obstacle that traverse vector collides with

35 for lower-hang obs. $\omega_{avo} = \omega_a$, otherwise $\omega_{avo} = -\omega_a$;

36 $[\omega_e', a_e'] = \boldsymbol{SpeedShaper}\left(\omega_{avo} \omega_{e_{pre}}', a_{e_{pre}}', t_s\right)$

37 **else**

38 **if** $\max(ElvChk) == 1 \mid$ ConvOpt $== 2 \mid$ ConvOpt $== 3$

39 $[\omega_e', a_e'] = \boldsymbol{SpeedShaper}\left(0 \omega_{e_{pre}}', a_{e_{pre}}', t_s\right)$

40 **else**

41 $[\omega_e', a_e'] = \boldsymbol{SpeedShaper}\left(\omega_e, \omega_{e_{pre}}', a_{e_{pre}}', t_s\right)$

42 $\omega_{e_{pre}}' = \omega_e'; \quad \omega_{t_{pre}}' = \omega_t'; \quad a_{e_{pre}}' = a_e'; \quad a_{t_{pre}}' = a_t';$

43 **return** $\omega_t' \ and \ \omega_e'$

The Algorithm-4.2 checks whether a given point is inside a Cartesian based rectangular shaped obstacle.

Algorithm-4.2: CollisionDet

In: p^i, O_1

Out: $lapChk$

1 $TraMax = \max\left(O_1(:,1)\right); \quad TraMin = \min\left(O_1(:,1)\right);$

$ElvMax = \max\left(O_1(:,2)\right); \quad ElvMin = \min\left(O_1(:,2)\right);$

2 **if** $p^i(1,1) \leq TraMax \ \& \ p^i(1,1) \geq TraMin \ \&\&$

$p^i(1,2) \leq ElvMax \ \& \ p^i(1,2) \geq ElvMin$

3 $lapChk = 1$

4 **else** $\rightarrow lapChk = 0$ **return** $lapChk$

The Algorithm-4.3 checks whether a given point is close to the side or top surface of a Cartesian based rectangular shaped obstacle, using the expression in Eq. (4.9), and decides the converging axis accordingly.

Algorithm-4.3: ConvAxis

In: p^i, O_1

Out: $AxConv$

1 $TraMax = \max(O_1(:,1))$; $TraMin = \min(O_1(:,1))$;

$ElvMax = \max(O_1(:,2))$; $ElvMin = \min(O_1(:,2))$;

2 $d_t = \min(|TraMax - p^i(1,1)|, |TraMin - p^i(1,1)|)$

$d_e = \min(|ElvMax - p^i(1,2)|, |ElvMin - p^i(1,2)|)$

3 **if** $d_e \leq d_t + \delta_e + \varepsilon$

4 $AxConv = 2$ (elevation)

5 **else** $\rightarrow AxConv = 1$ (traverse)

6 **return** $AxConv$

The Algorithm-4.4 enables the system to execute speed commands by complying with the given acceleration and jerk limits by using all kinematic parameters. The theoretical background of the algorithm used as a real time speed shaper is discussed in detail in the reference [67]. Speed limits are also added to the algorithms mentioned in this study. The symbols of ω_c, ω_c', $\omega_{c_{pre}}'$, a_c' and $a_{c_{pre}}'$ represent the instant raw speed command, shaped speed command, previous shaped speed command, acceleration command and previous acceleration command, respectively. Also, the symbols of ω_c^{max}, a_c^{max} and j_c^{max} present the maximum allowable speed, maximum allowable acceleration and maximum allowable jerk of the related axis, respectively. For simplicity, the maximum acceleration and deceleration limits, plus and minus jerk values are shown with the same symbol. An example speed planning is made by using Algorithm-4.4 as in Figure 4.13. As can be seen from the example, the $60°/s$ speed command is shaped by adhering to the specified speed, acceleration and jerk limits which are $57.3°/s$, $114.6°/s^2$ and $343.8°/s^3$, respectively.

Algorithm-4.4: SpeedShaper

In: $\omega_c, \omega'_{c_{pre}}, a'_{c_{pre}}, t_s$

Out: ω'_c, a'_c

if $\omega_c > \omega_c^{max}$

  $\omega_c = \omega_c^{max}$

elseif $\omega_c < -\omega_c^{max}$

  $\omega_c = -\omega_c^{max}$

$\omega_{dif} = \omega'_{c_{pre}} - \omega_c$

$d = j_{max} \cdot t_s^2 \; a_i = t_s \cdot a'_{c_{pre}}, \; y = a_i + \omega_{dif}$

$a_1 = \sqrt{d \cdot (d + 8|y|)}, \; a_2 = a_i + sgn(y) \cdot \frac{a_1 - d}{2}$

$sy = \frac{sgn(y+d) - sgn(y-d)}{2}, a = (a_i + y - a_2) \cdot sy + a_2$

$$sa = \frac{sgn(a+d) - sgn(a-d)}{2}$$

$$j_c = -j_c^{max} \cdot sa\left(\frac{a}{d} - sgn(a)\right) - j_c^{max} \cdot sgn(a)$$

$da = j_c \cdot t_s, a'_c = a'_{c_{pre}} + da$

if $a'_c > a_c^{max}$

  $a'_c = a_c^{max}$

elseif $a'_c < -a_c^{max}$

  $a'_c = -a_c^{max}$

$d\omega = a'_c \cdot t_s$

$\omega'_c = \omega'_{c_{pre}} + d\omega$

return ω'_c, a'_c

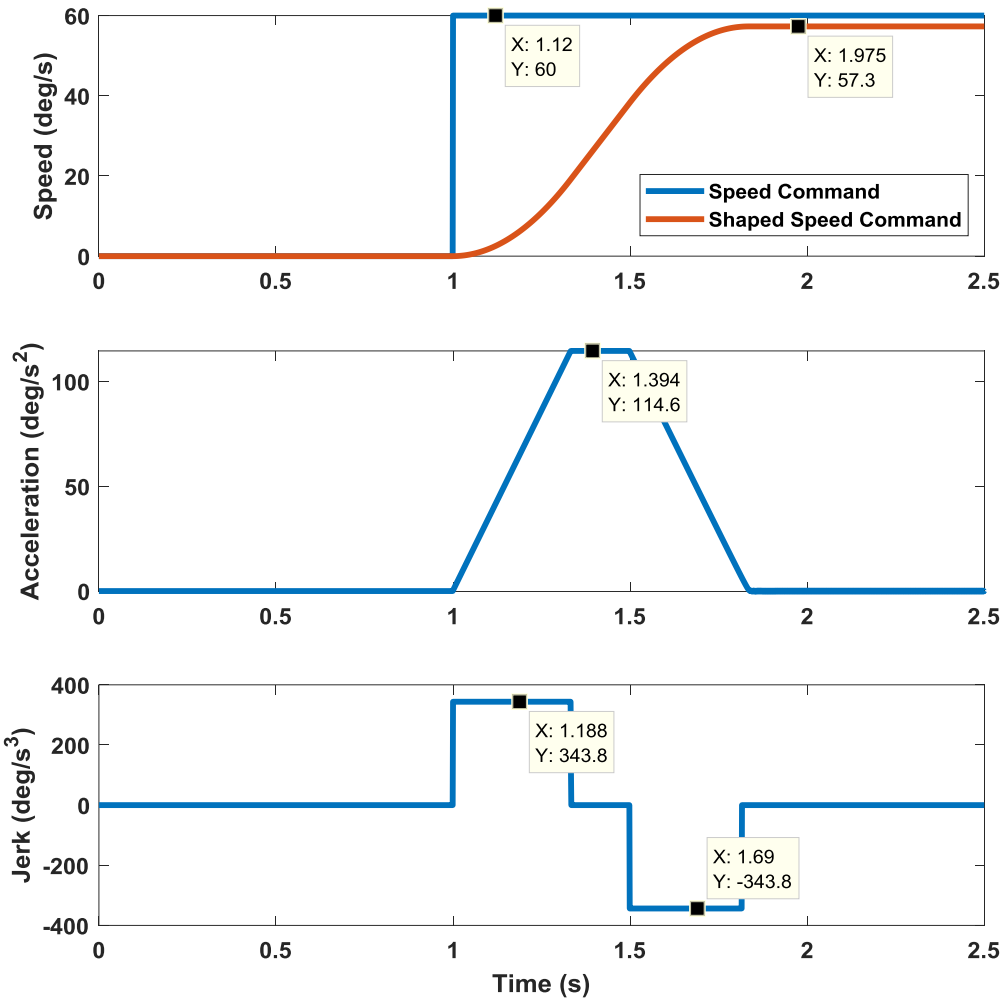

Figure 4.13. Sample speed shaping

After the speed command is shaped according to the desired limits, another important point is to calculate the stopping distance of an axis whose instantaneous velocity and acceleration are known. In the model where the speed is trapezoidal, the acceleration is step, the jerk is infinite, the stopping distance can be easily calculated as $\frac{(\omega_c')^2}{2a_c^{max}}$ which depends only on instant speed. However, in the model where acceleration is trapezoidal and jerk is step, the stopping distance is calculated as in Eq. (4.12) which depends not only on instant speed and also on instant acceleration. When instantaneous velocity is shown as ω_0 and instantaneous acceleration is shown as a_0, the change in velocity is calculated as in Eq. (4.10) to set the instantaneous

acceleration to zero. After the instantaneous acceleration is reset, the acceleration required to reset the instantaneous velocity is calculated as in Eq. (4.11).

$$\omega_i = sgn(a_0)\frac{a_0^2}{2j} \tag{4.10}$$

$$a_x = \begin{cases} -sgn(\omega_0)\dfrac{\sqrt{2a_0^2 + |4j\omega_0|}}{2}, & sgn(\omega_0) == sgn(\omega_0 + \omega_i) \\ -sgn(\omega_0 + \omega_i)\sqrt{|j(\omega_0 + \omega_i)|}, & sgn(\omega_0) \neq sgn(\omega_0 + \omega_i) \end{cases} \tag{4.11}$$

The stooping distance of an axis whose instantaneous velocity and acceleration are given as ω_0 and a_0 is obtained as in Eqs. (4.13) and (4.14) according to the conditions in Eq. (4.12). The symbols a_{max} and j denote acceleration and jerk limits, respectively.

$$\boldsymbol{StopDis}(\omega_0, a_0) = \begin{cases} d_1, & a_{max} \leq |a_x| \\ d_2, & a_{max} > |a_x| \end{cases} \tag{4.12}$$

$$d_1 = B_1 + B_2 \tag{4.13}$$

$$d_2 = \frac{a_0^3 - 3a_0^2 a_x + 3j\omega_0 a_0 A_4 + 3a_x^3 - 6j\omega_0 a_x A_4}{3j^2} \tag{4.14}$$

where

$$B_1 = \frac{sgn(\omega_0)(a_{max}^4 + 6j\omega_0 A_1 a_{max} + 6ja_o A_1 A_3 - 3jA_3^2)}{6j^2 a_{max}} \tag{4.15}$$

$$B_2 = \frac{(6j^2\omega_0 A_3 - 6j^2 A_3 A_2 + 3a_0 A_1^2 a_{max} - A_1^3 a_{max})}{6j^2 a_{max}} \tag{4.16}$$

$$A_1 = a_0 + a_{max}\, sgn(\omega_0) \tag{4.17}$$

$$A_2 = \frac{sgn(\omega_0)A_1^2}{2j} \tag{4.18}$$

$$A_3 = \left| \frac{sgn(\omega_0)(2a_0A_1 - a_{max}^2) + 2j\omega_0 - 2jA_2}{2j} \right| \tag{4.19}$$

$$A_4 = sgn(\omega_0 + \omega_i) \tag{4.20}$$

4.2 Simulation

A number of simulations are conducted to examine the algorithm's performance and behavior under different conditions. First, the worst conditions, then the behavior under different constant speed commands and custom noisy signals are examined.

4.2.1 Simulation of the Worst Cases

Handling all worst scenarios as possible is critical for the robustness of the algorithm. We can examine the worst scenarios that may occur in such systems under 4 subtitles. The precautions taken for these subtitles will play a role in shaping the frame of the algorithm.

4.2.1.1 Narrow width obstacles

It is stated that a single equivalent ahead position would not be sufficient for Narrow width obstacles. In Figure 4.14, such a situation is examined and the effect of

different n_c on obstacle avoidance is shown. If only one equivalent ahead position is used as in the upper figure of Figure 4.14, the collision occurred because the turret could not see the obstacle after a while. However, when the n_c number is calculated according to the Eq. (4.7) and the operations are done accordingly, as it can be seen from the lower figure of Figure 4.14, the collision did not occur because the turret always sees the obstacle with the help of 3 equivalent look ahead positions.

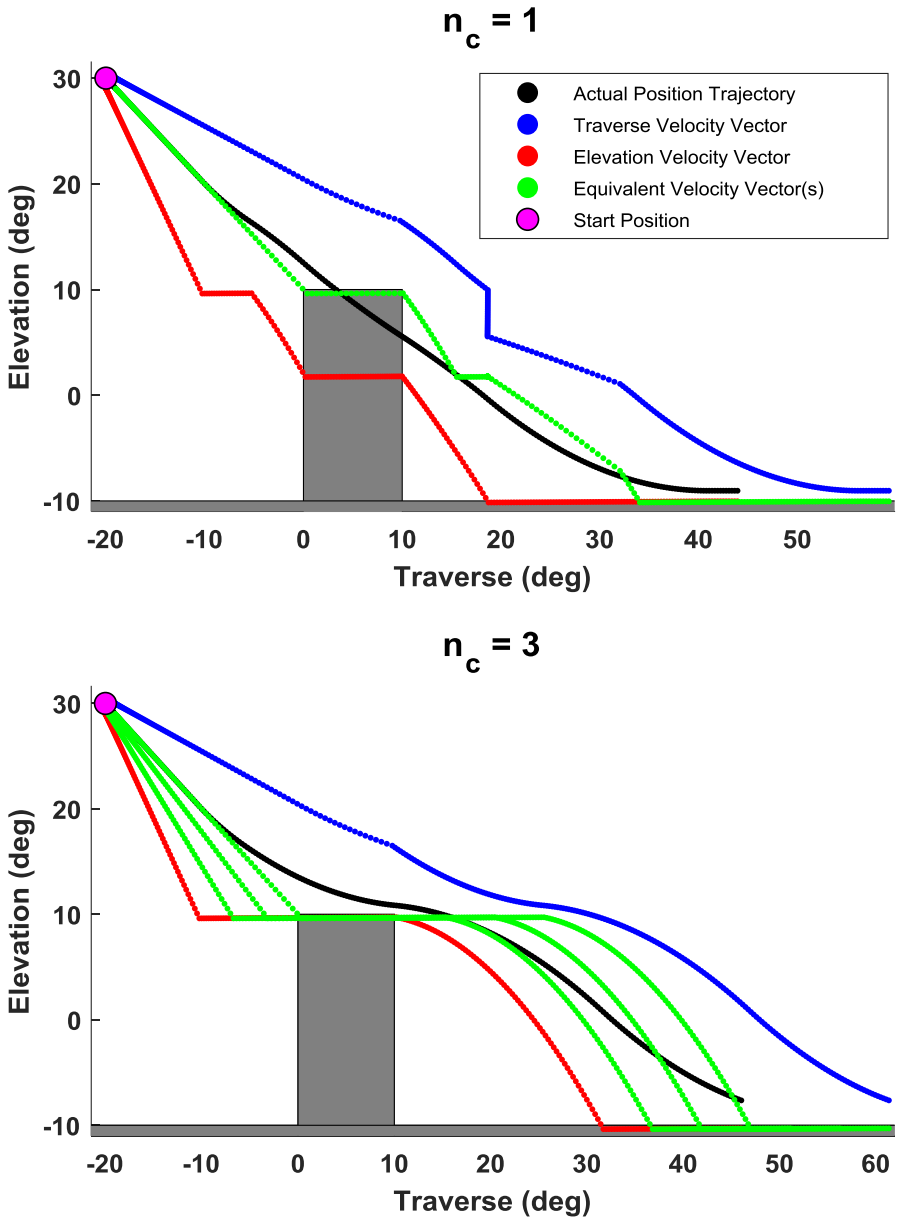

Figure 4.14. The effect of the number of n_c on avoiding the narrow width obstacles

4.2.1.2 Adjoining obstacles

It was critical as some action was required on both axes to overcome adjacent obstacles. In Figure 4.15, when $57.3°/s$ speed command is given to both axes, the results in case of double-acting active and inactive are given. Accordingly, by performing double-acting, the system cannot allow any collision in this case either.

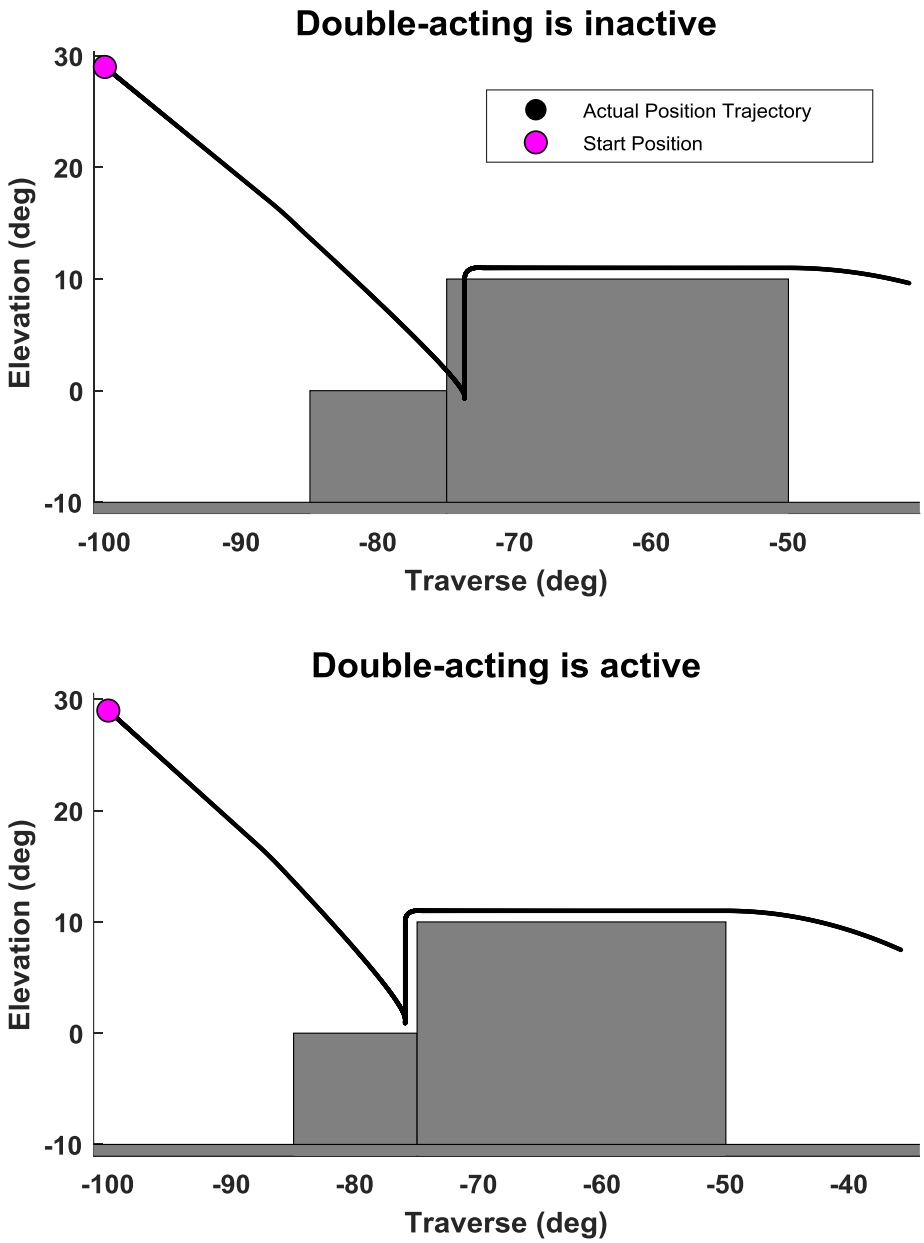

Figure 4.15. The effect of proposed solution for adjoining obstacles

4.2.1.3 Low speed motion

In the worst cases section, it is mentioned that low speed movements cause local minimum and precautions are taken for this. The effect of the precautions taken can be seen in the simulation as in Figure 4.16. While the precautions are inactive, in the case of sending $1°/s$ speed commands to both axes, the system encounters the local minimum and cannot avoid the obstacle, but with the proposed solution, the obstacle is avoided without experiencing any local minimum problem as it can be seen in Figure 4.16.

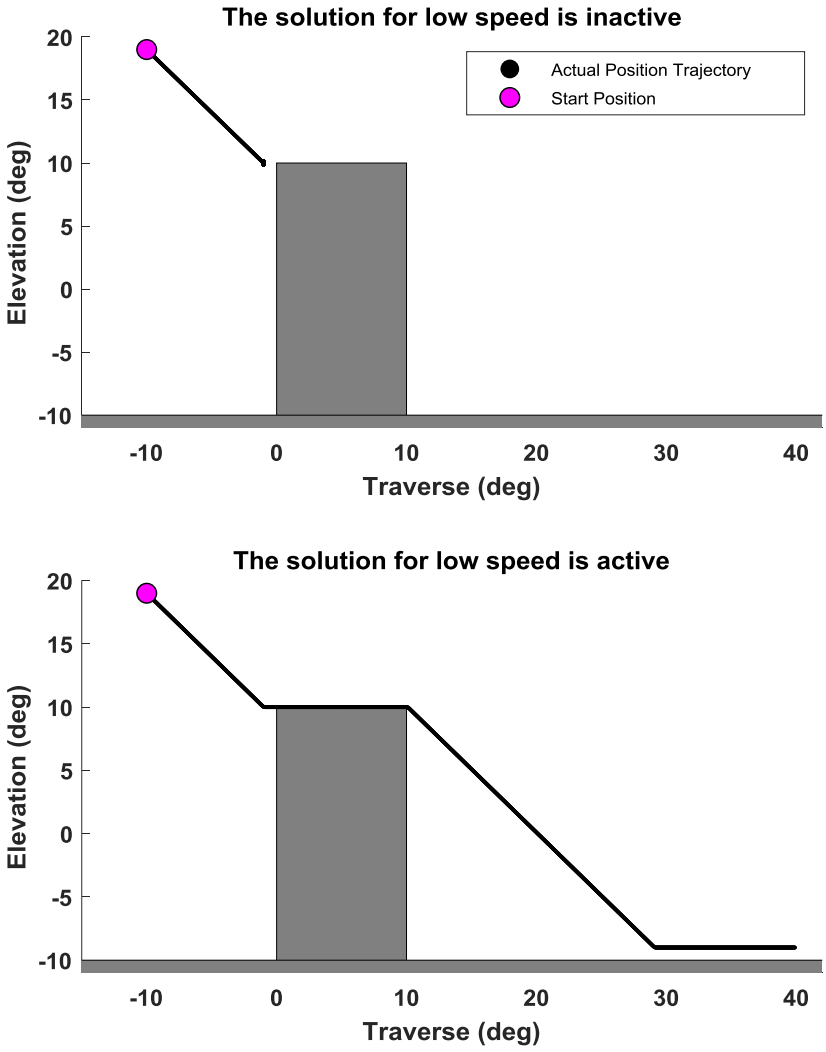

Figure 4.16. The effect of proposed solution for low speed motions

4.2.1.4 Emergency case

If the current position of the turret remains within the boundaries of the obstacle, the turret moves out of the obstacle by either ascending or descending on the elevation axis depending on the position of the obstacle. In Figure 4.17, although the user sends zero speed commands to both axes, the turret rises until it reaches a safe area and then stops.

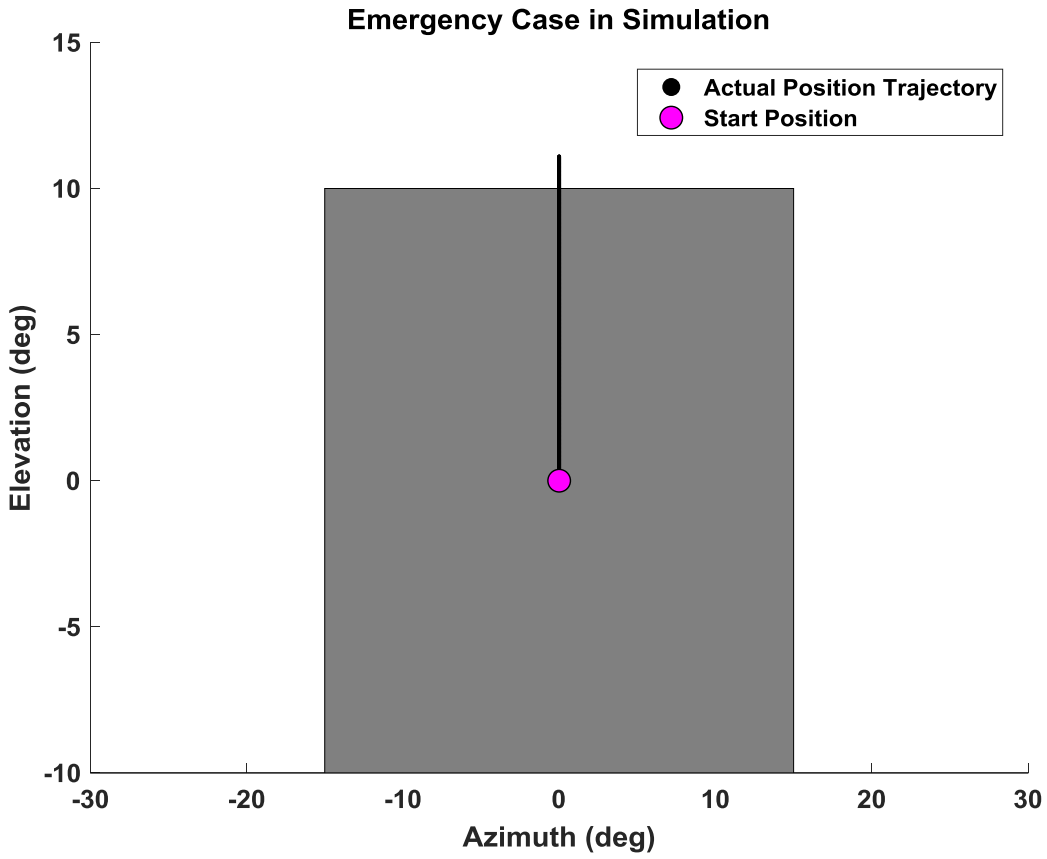

Figure 4.17. Moving out of the obstacle by ascending

4.2.2 Simulations with Different Step Speed Commands

In order to measure the performance of the system under different speeds, simulations are run by giving the following 7 different speed commands. While constant commands are given to the traverse axis, the speed commands for the elevation axis are created in the form of steps in order to avoid upper and lower-hang obstacles.

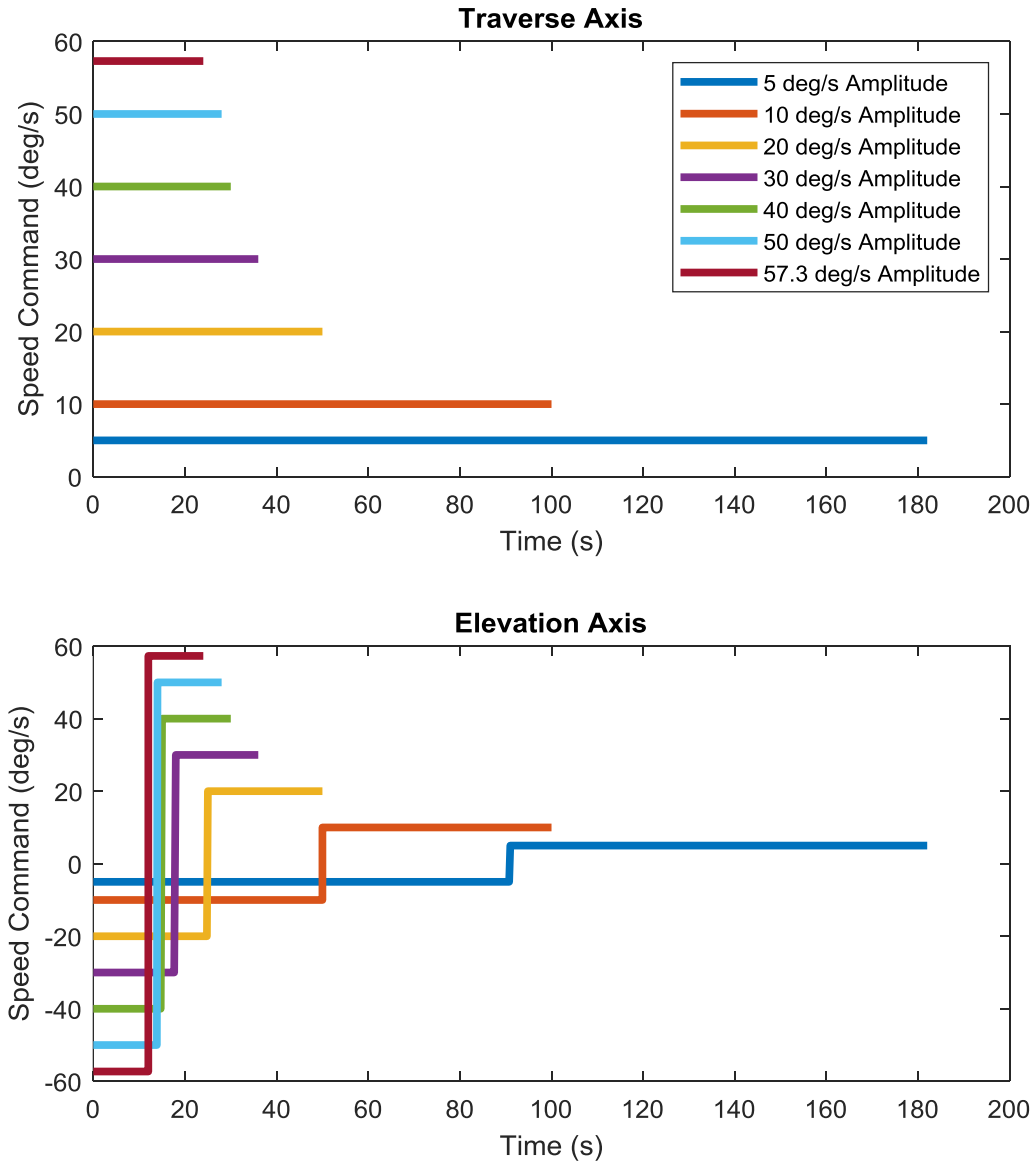

Figure 4.18. Speed commands of both axes

To show that the speed, acceleration and jerk limits are working successfully, for both axes they are limited to the values $57.3°/s$, $114.6°/s^2$ and $344°/s^3$, respectively. The first 5 seconds of the test performed at $57.3°/s$ speed given in Figure 4.18 is examined as in Figure 4.19. As can be seen in the graphs, the velocity, acceleration and jerk values in the axes are within the limits. The regions where the jerk values exceed the limits are generally the regions where the speed is close to zero. In these areas, friction is quite disruptive and a stick-slip effect is seen.

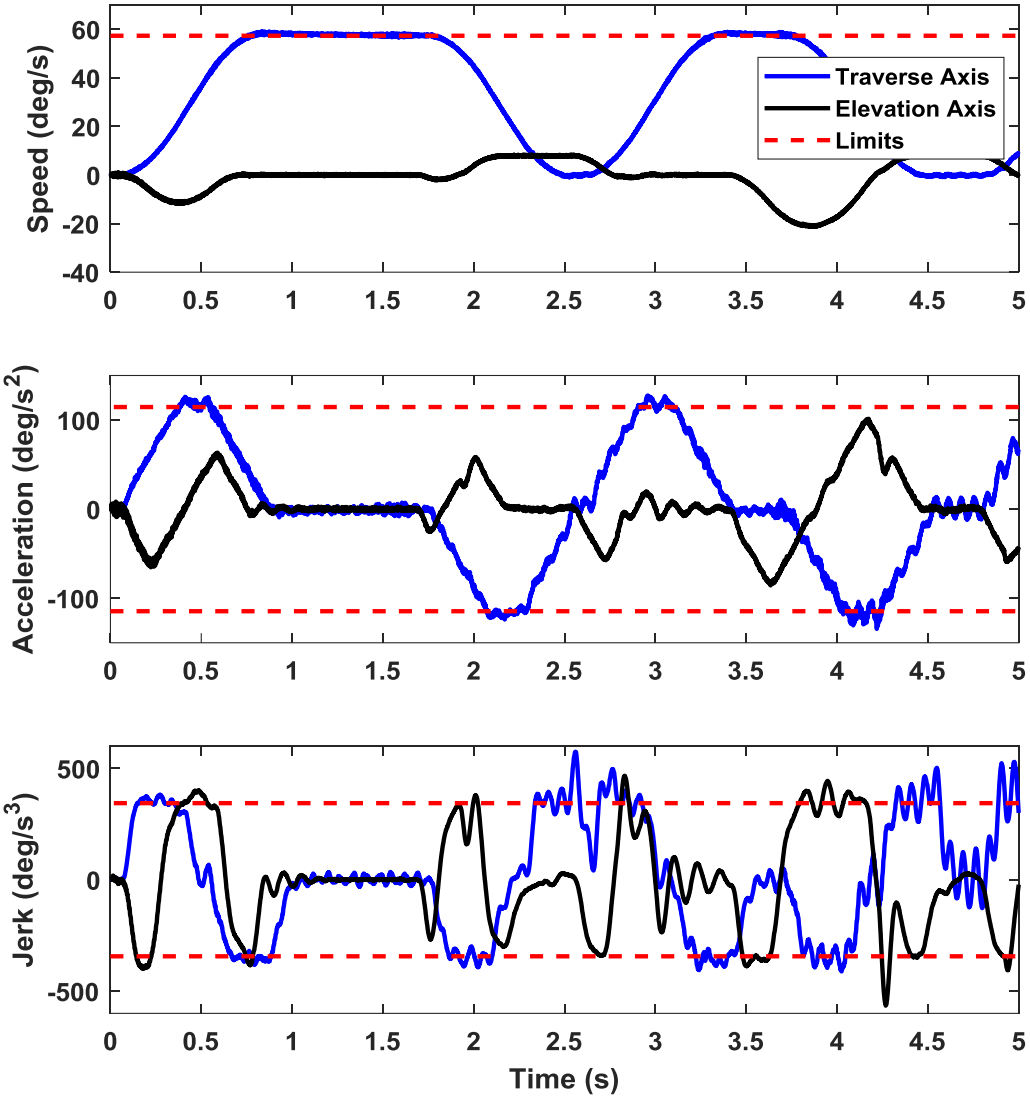

Figure 4.19. The changes of velocity, acceleration and jerk of both axes (for $344°/s^3$ jerk limit)

For the rest of the tests, the velocity, acceleration and jerk values of both axes are limited to the values $57.3°/s$, $114.6°/s^2$ and $2000°/s^3$, respectively. The jerk value is kept quite high to increase the stabilization performance of the turret. The first 5 seconds of the test performed at $57.3°/s$ speed given in Figure 4.18 is examined as in Figure 4.20. As can be seen in the graphs, the velocity, acceleration and jerk values in the axes are within the limits.

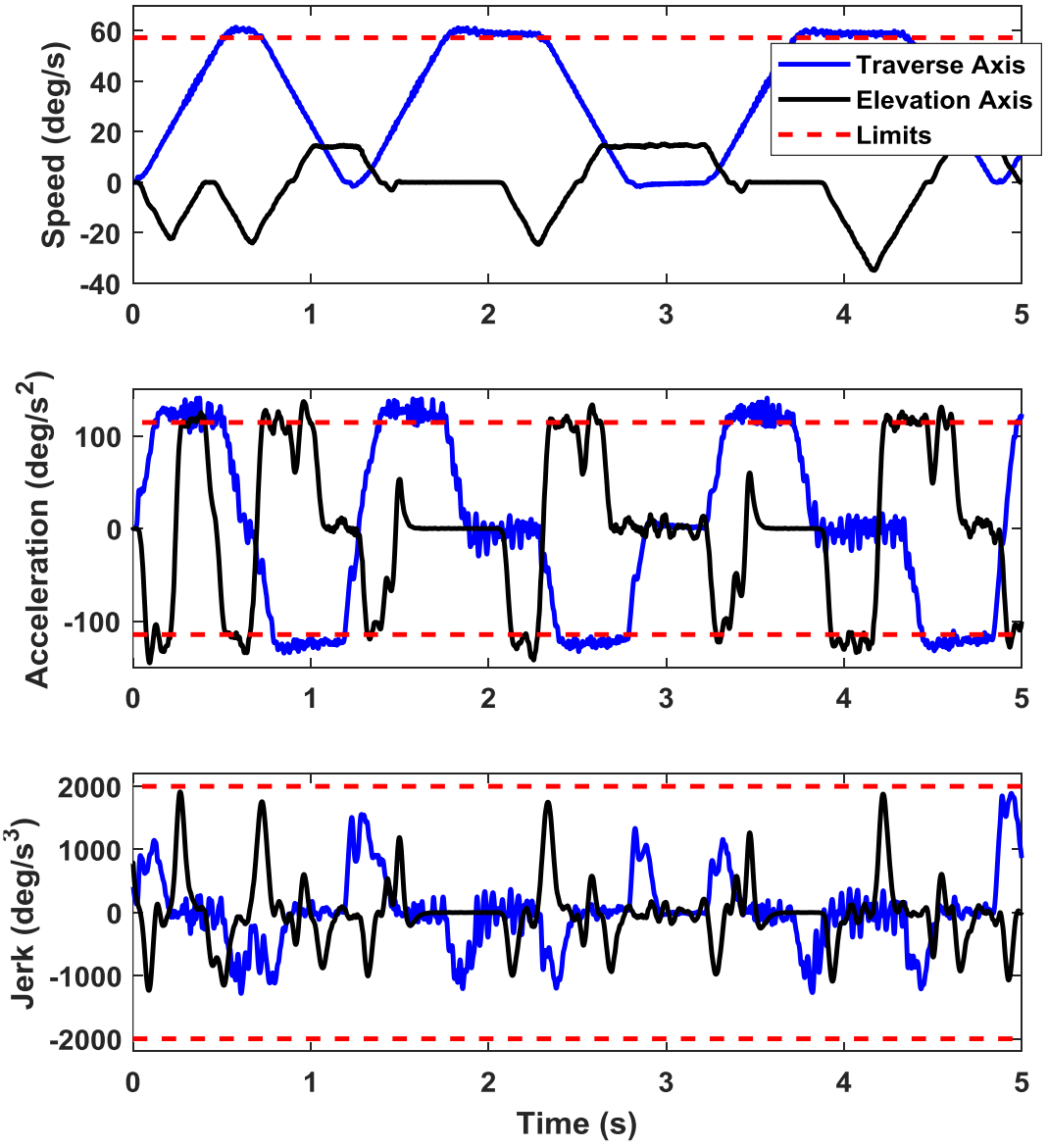

Figure 4.20. The changes of velocity, acceleration and jerk of both axes (for $2000°/s^3$ jerk limit)

The results obtained by giving the speed commands in Figure 4.18 are as in Figure 4.21. As can be seen, in all of the simulations made with 7 different speeds, the turret does not collide with the obstacles and continues its movement. When positive speed commands are given to the traverse axis, the turret approaches all obstacles from their left.

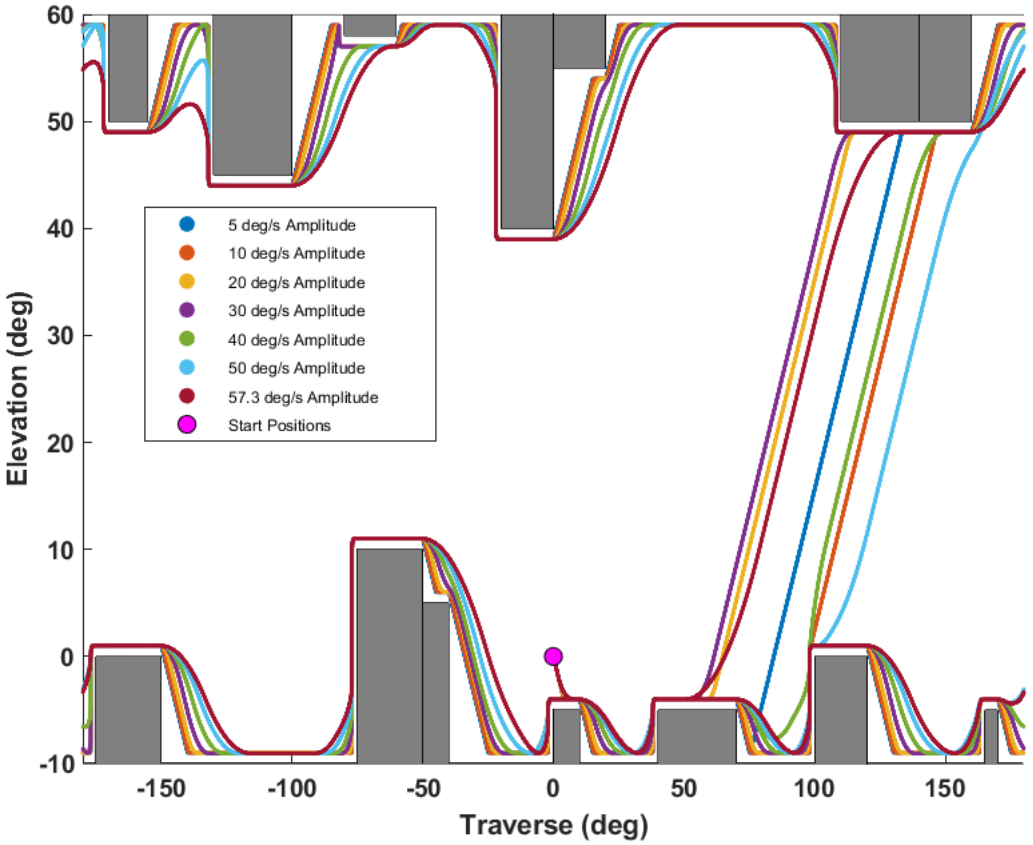

Figure 4.21. Actual position trajectories of turret for 7 different speed commands

In order to see the circumference of the upper two adherent obstacles seen in Figure 4.21, a zoomed view is given in Figure 4.22. As mentioned, while explaining the algorithm, the radius formed at different speeds under the same acceleration is different and this radius increases as the speed increases.

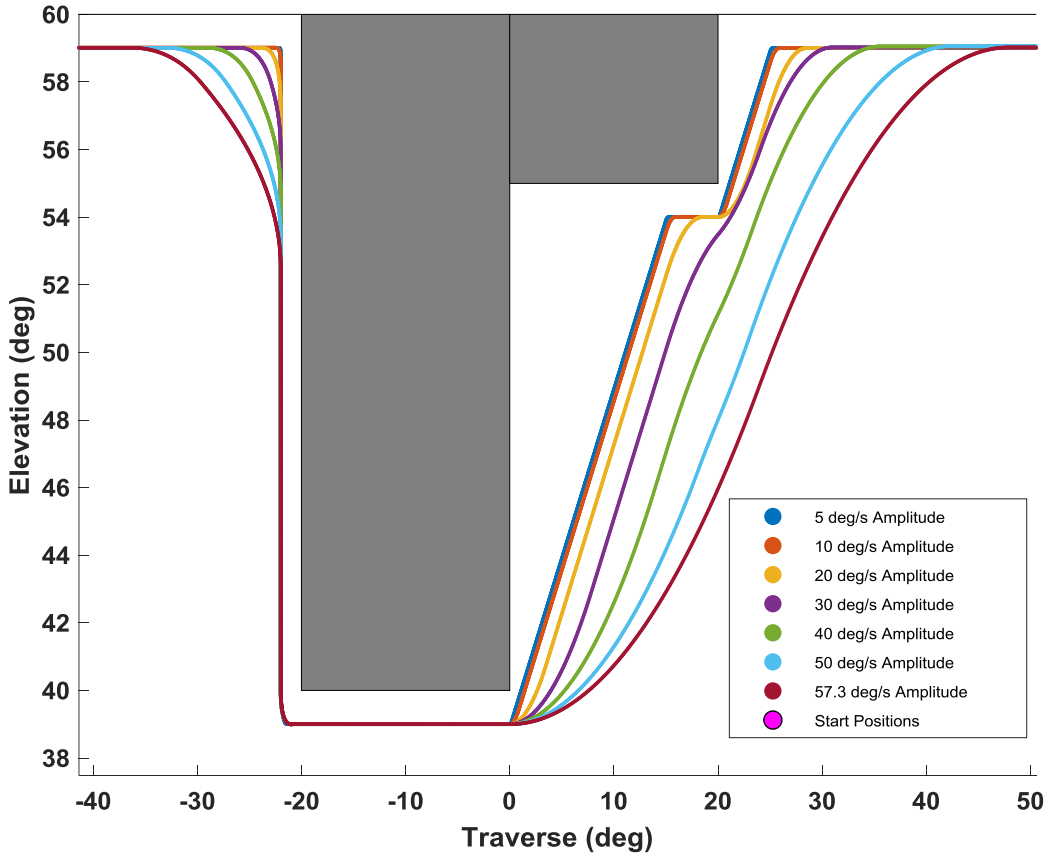

Figure 4.22. Actual position trajectories of turret for 7 different speed commands

To show how the speed commands given according to the obstacles and acceleration limits are shaped by the algorithm, the results with only $20°/s$ are given in Figure 4.23. The shaped speed commands are applied to the system.

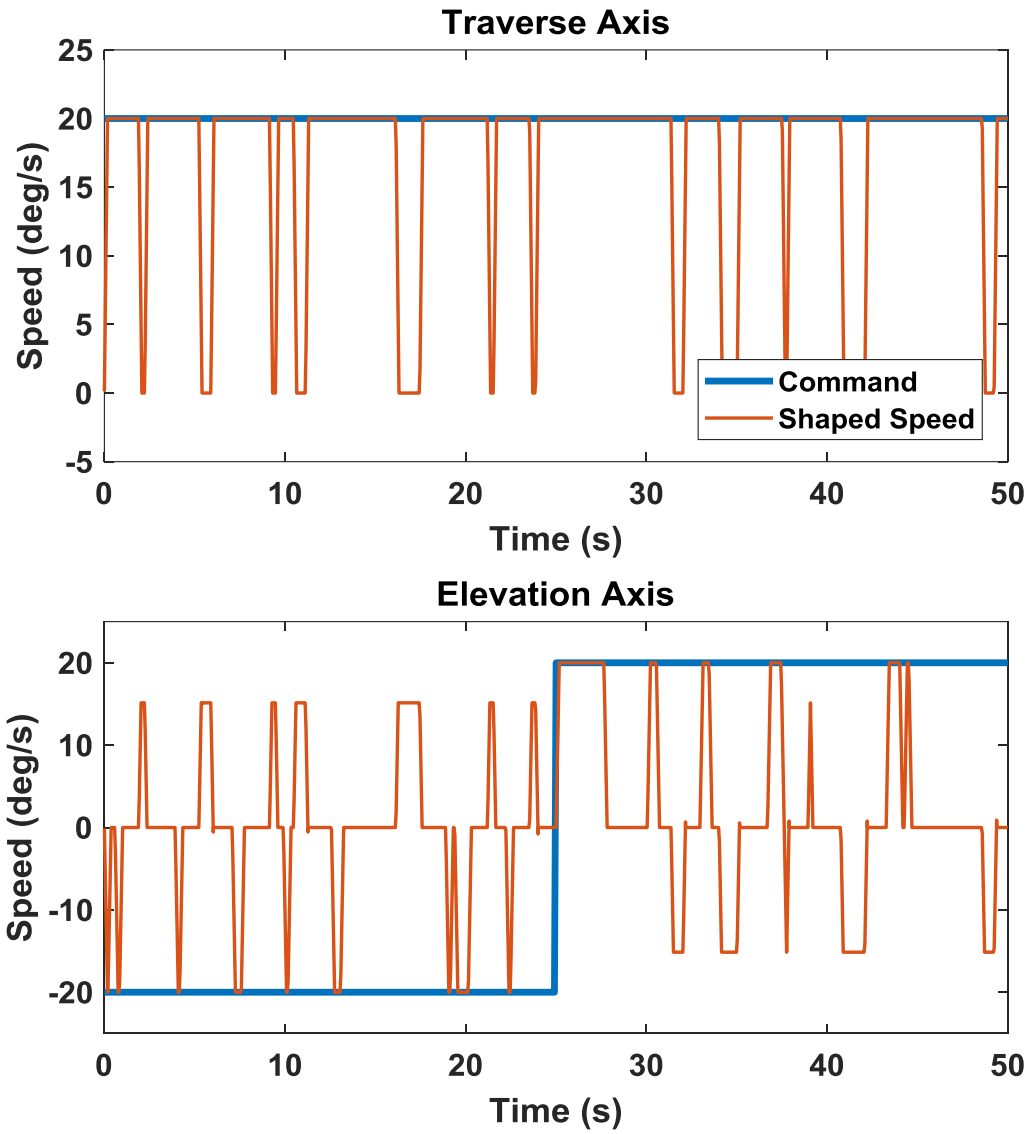

Figure 4.23. Speed commands and shaped speed commands of two axes for 20°/s

When the speed commands in Figure 4.18 are given, the turret approaches all obstacles from their left. In order to test the obstacles from their right, the elevation speed commands in Figure 4.18 are kept the same and only the traverse axis speed commands are given as negative and the tests are repeated. In this case, the turret does not collide with any obstacles.

In order to see the circumference of the upper two adherent obstacles seen in Figure 4.24, the zoomed view is given in Figure 4.25.

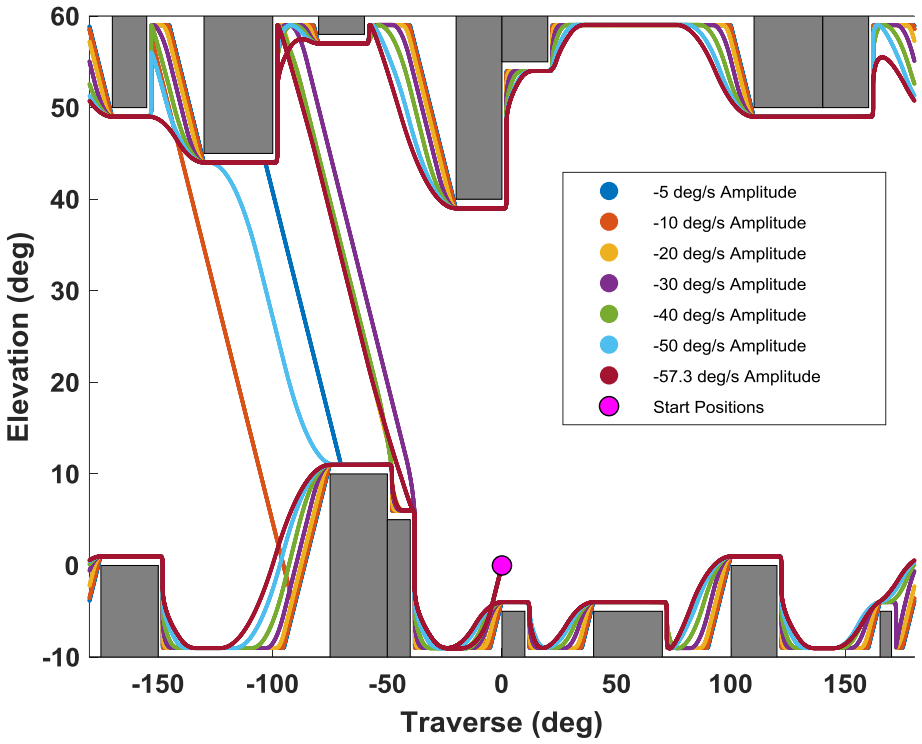

Figure 4.24. Actual position trajectories of turret for 7 different speed commands

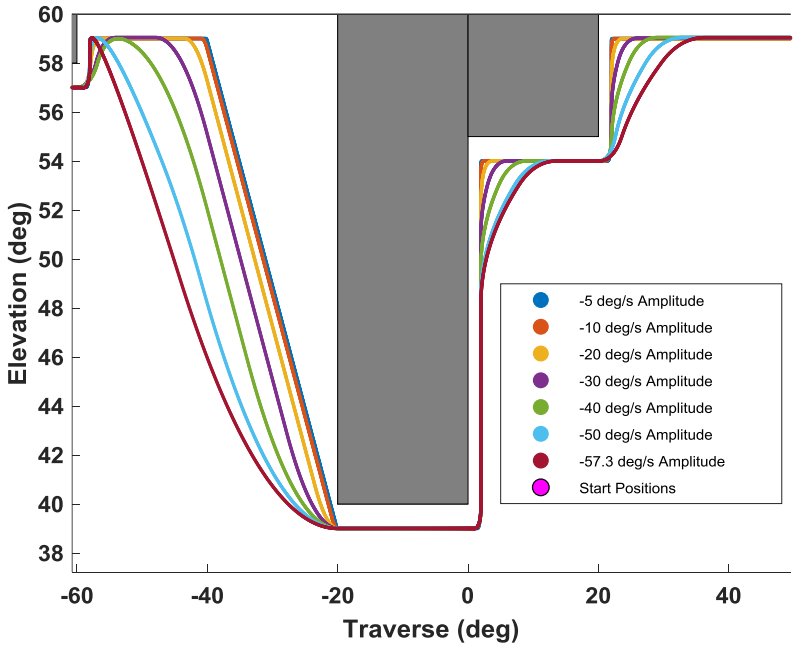

Figure 4.25. The circumference of the upper two adjoining obstacles for negative traverse speed command

To show how the speed commands given according to the obstacles and acceleration limits are shaped by the algorithm, the results with only $-20°/s$ are given in Figure 4.26.

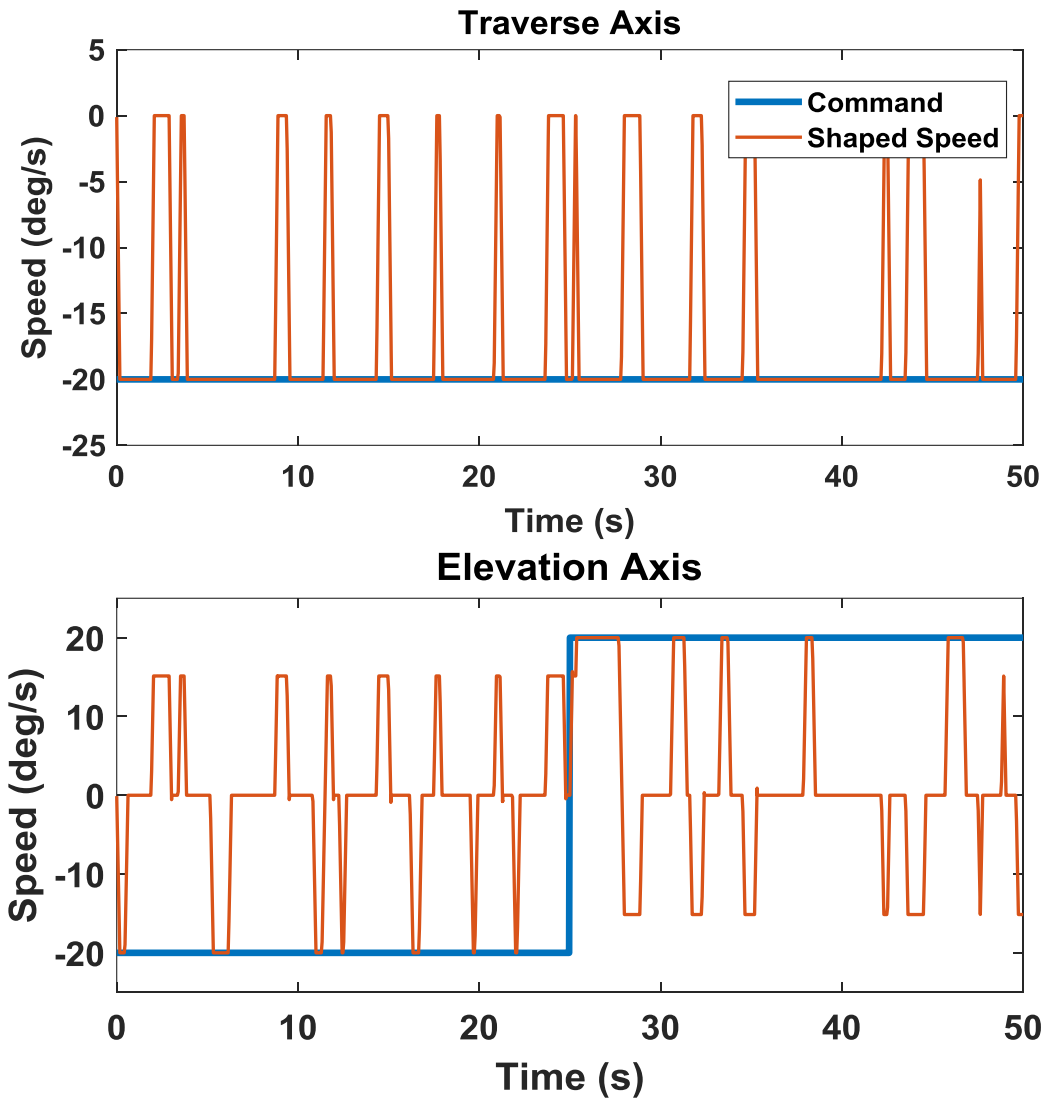

Figure 4.26. Speed commands and shaped speed commands of two axes for -20°/s

4.2.3 Simulations with Noisy Custom Speed Commands

In order to see the performance of the algorithm under the noisy speed commands, the following custom noisy speed commands are given to the traverse and elevation axes and the simulation is run. This speed command is obtained by adding band-limited white noise to the custom speed command created by combining different speeds. The noise power of band-limited white noise (the height of the power spectral density (PSD) of the white noise) is selected as 0.1, the sample time as 0.01 and the seed as 23341, and the output of this noise signal is limited to ±5°/s and added to the speed command. Thus, the robustness of the algorithm can be tested. Because speed commands from the user or controller can be noisy.

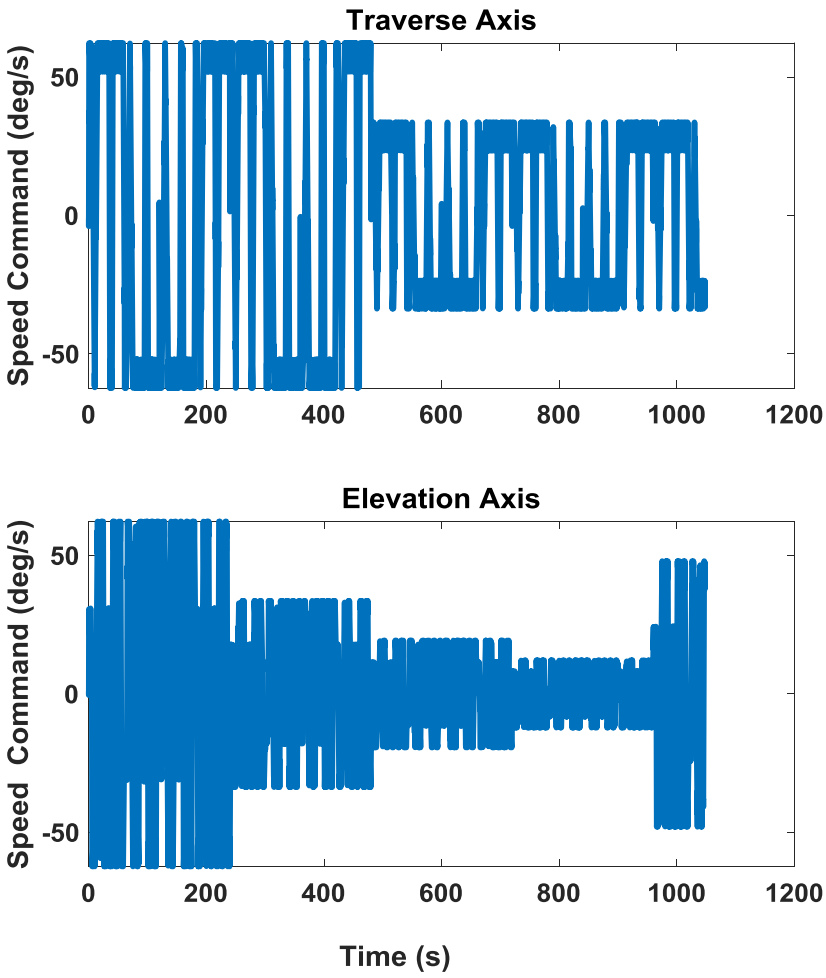

Figure 4.27. Noisy speed commands

The result obtained by giving speed commands in Figure 4.27 is as in Figure 4.28.

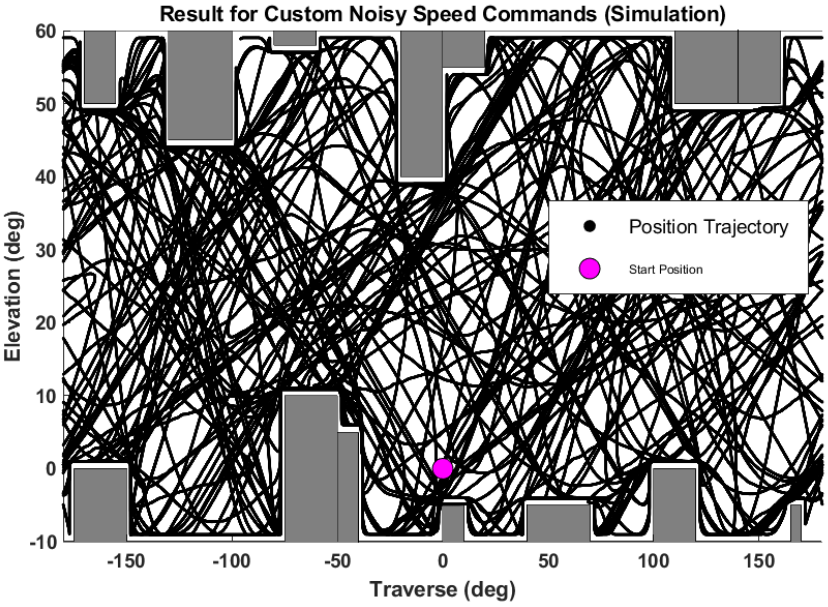

Figure 4.28. Actual position trajectories for custom noisy speed commands

In order to see the circumference of the upper two adherent obstacles in Figure 4.28, a zoomed view is given in Figure 4.29.

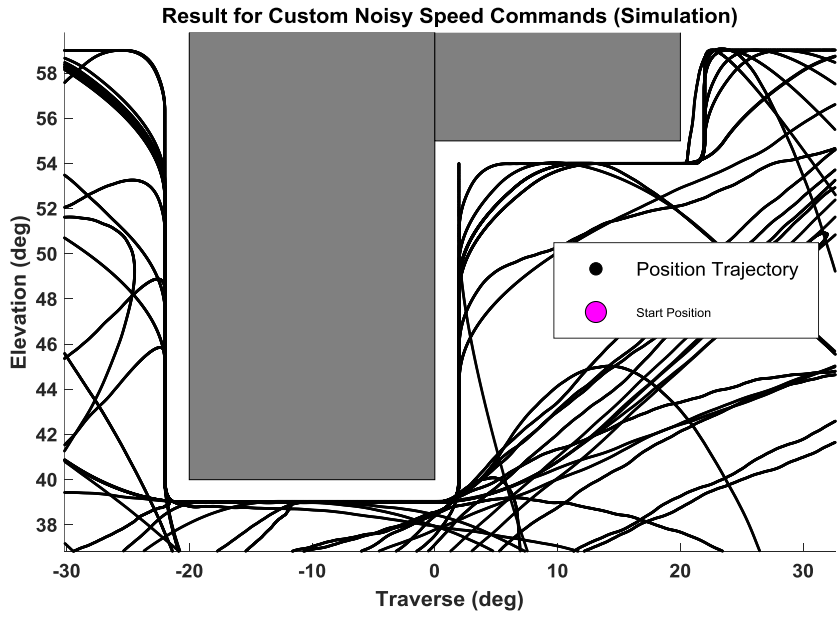

Figure 4.29. The circumference of the upper two adjoining obstacles for custom noisy speed commands.

4.2.4 Simulation of Position Control for Path Planning

It is possible to transform the collision avoidance algorithm into a path planning algorithm by adding a position loop in the form of a cascade as shown in Figure 4.2 on the same algorithm. Position trajectories between 4 different target points from the same starting point without collision are shown in the Figure 4.30. Accordingly, to go to the target, firstly the clockwise (CW) or counterclockwise (CCW) direction is determined and then the target is reached without collision. Here the goal is not the minimum time or path, but a simple and uncomplicated position control.

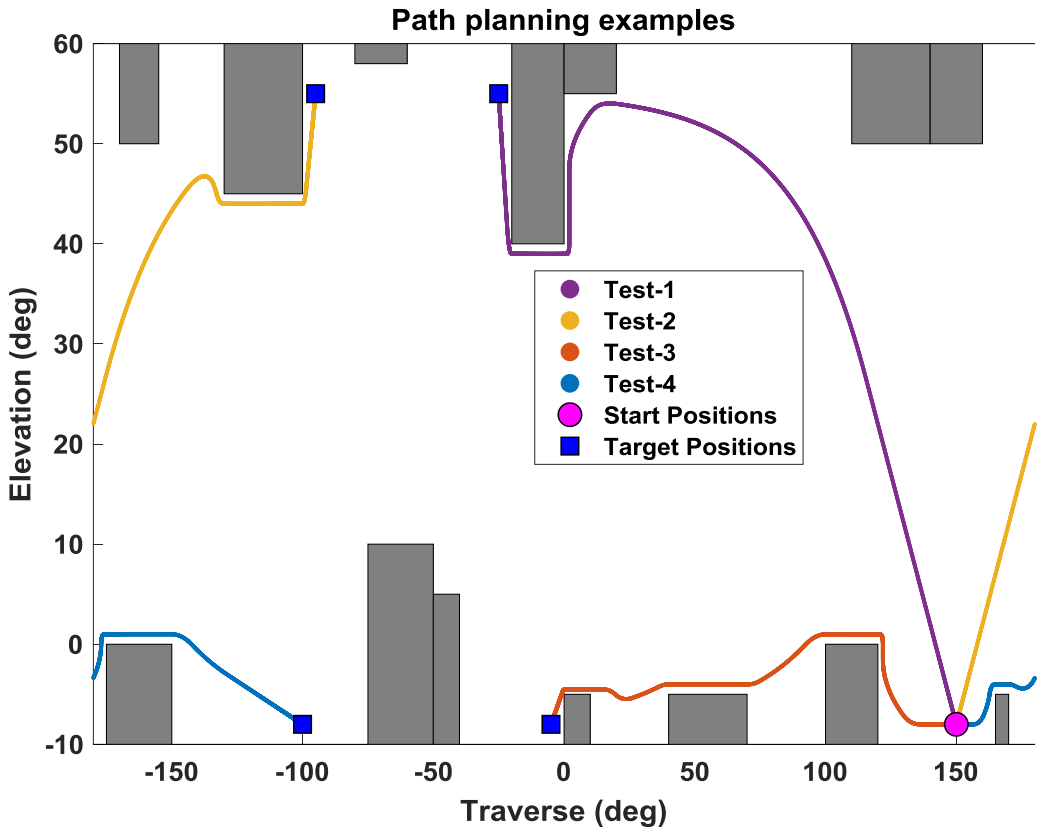

Figure 4.30. Some path planning examples in simulation

4.3 Experiments

A number of experiments are conducted to examine the algorithm's performance and behavior under different conditions in real-time. First, the worst conditions, then the behavior under different constant speed commands and custom noisy signals are examined. The tests are conducted on the stabilized remote-controlled gun turret system given in Figure 4.31. As mentioned, the system consists of two axes and each axis has a servo system. The algorithm is implemented to shape the speed reference only, without interfering with the speed / torque loops. All the obstacles used in the simulation are also defined in the real-time system and tests are carried out as if there are obstacles around the turret.

Figure 4.31. Test bench with 6-DOF Stewart platform and a stabilized remote-controlled gun turret system

4.3.1 Experiments of the Worst Cases

The worst cases are tested in the simulation. Later, the worst cases are created on a real-time system and the response of the system is observed.

4.3.1.1 Narrow width obstacles

It is stated that a single equivalent ahead position would not be sufficient for narrow width obstacles. The width of the obstacle defined in Figure 4.32 is 5°. When the maximum possible speed of the system is $57.3°/s$, the maximum deceleration is $114.6°/s^2$ and the maximum jerk is $2000°/s^3$, the minimum equivalent ahead position number (n_c) that should be used according to Eq. (4.7) is 3. Accordingly, two different tests are carried out and maximum speed commands ($57.3°/s$) are sent in two axes since it is aimed to test the worst case. The initial positions of the turret are chosen differently to converge to different axes. In either case, the turret did not hit the obstacle as expected.

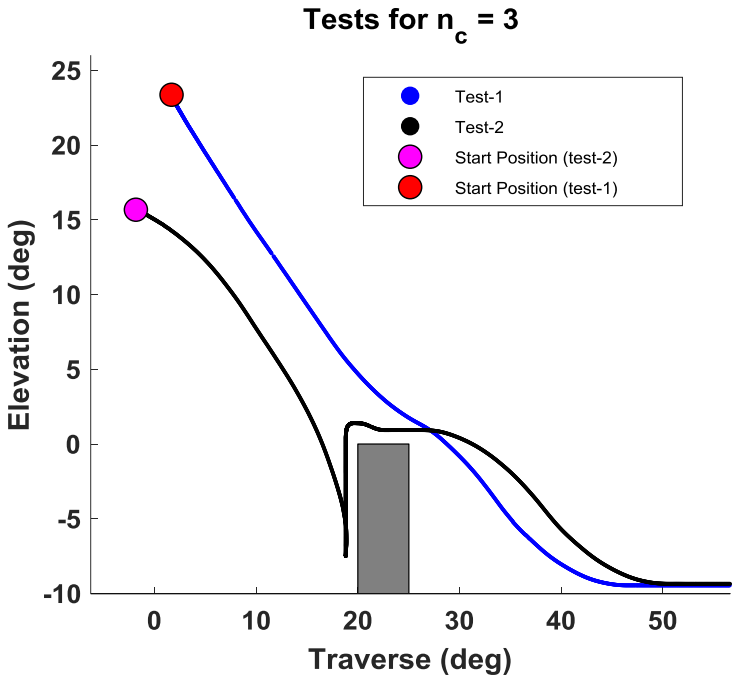

Figure 4.32. Real-time test of narrow width obstacle

4.3.1.2 Adjoining obstacles

It was critical as some action was required on both axes to overcome adjacent obstacles. Figure 4.33 shows the results when double-acting is active and $57.3°/s$ speed command is sent to both axes. Accordingly, by performing double-acting in real-time tests, the system cannot allow any collision in this case.

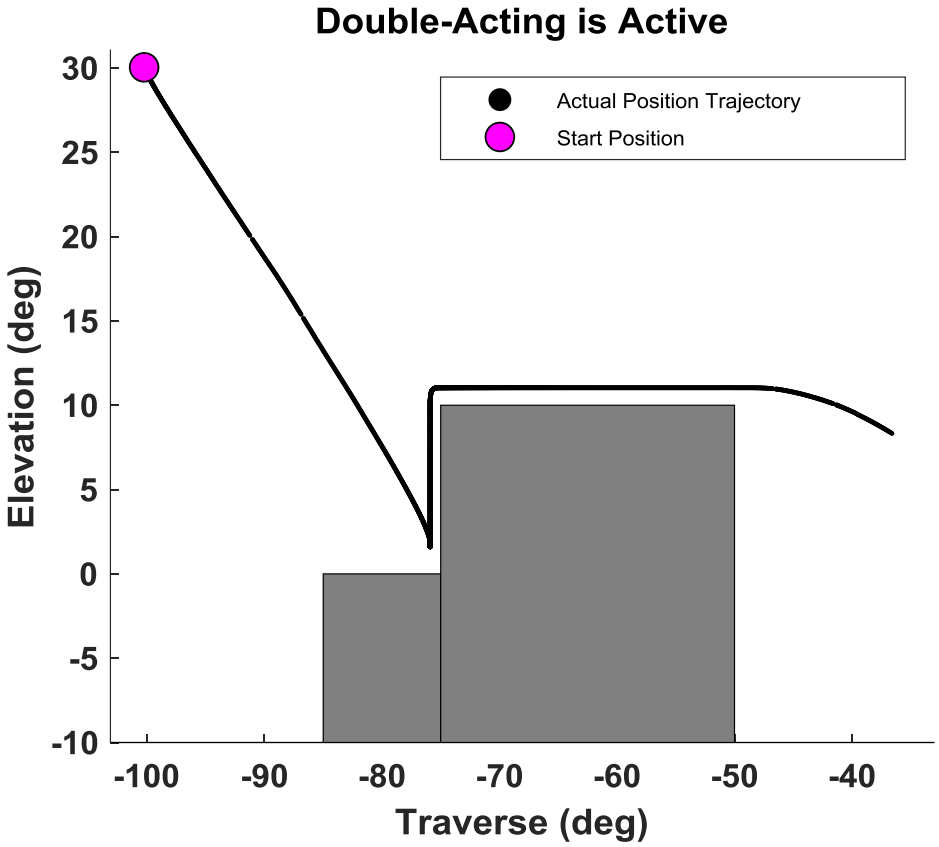

Figure 4.33. Real-time test of adjoining obstacles

4.3.1.3 Low speed motion

In the worst cases section, it is mentioned that low speed movements cause local minimum and precautions are taken for this. While the precautions are active, in the case of sending $1°/s$ speed commands to both axes, the system did not encounter the

local minimum and avoided the obstacle with the proposed solution as seen in Figure 4.34.

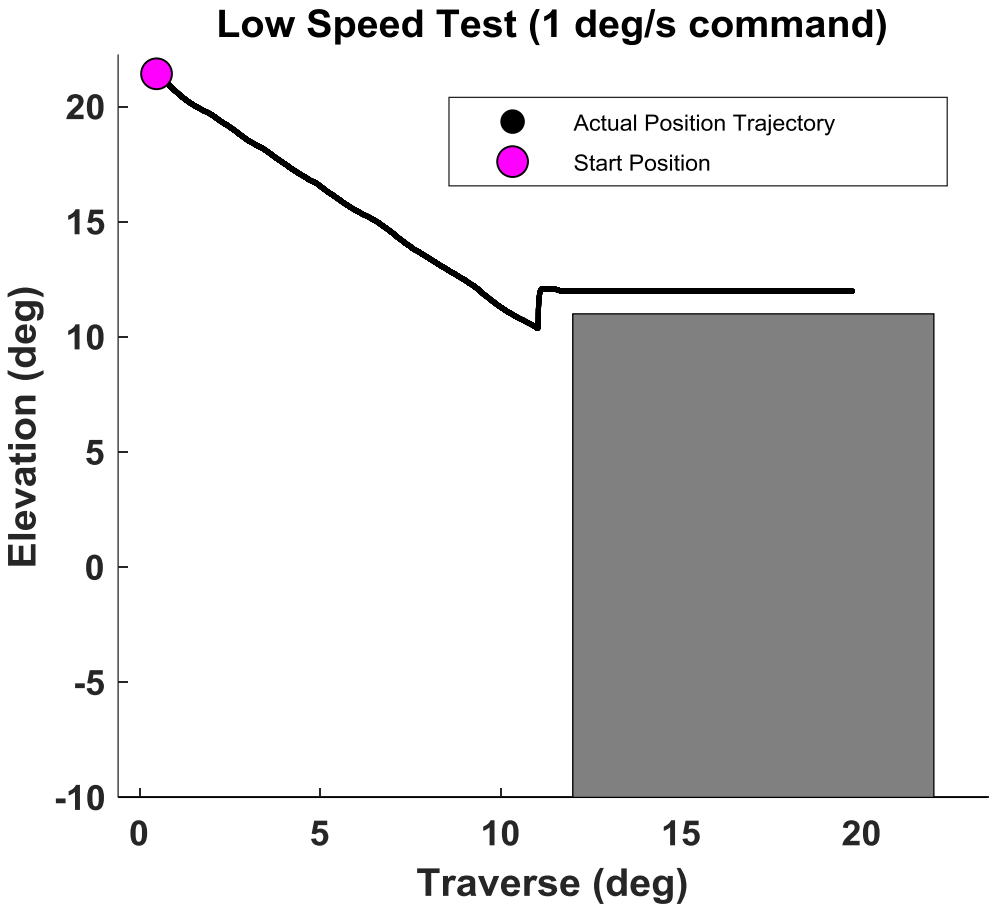

Figure 4.34. Real-time test of low speed motion

4.3.1.4 Emergency case

In the test, which is carried out with the current position of the turret inside the obstacle, the turret rose to a safe area and stopped despite zero speed commands as seen in Figure 4.35.

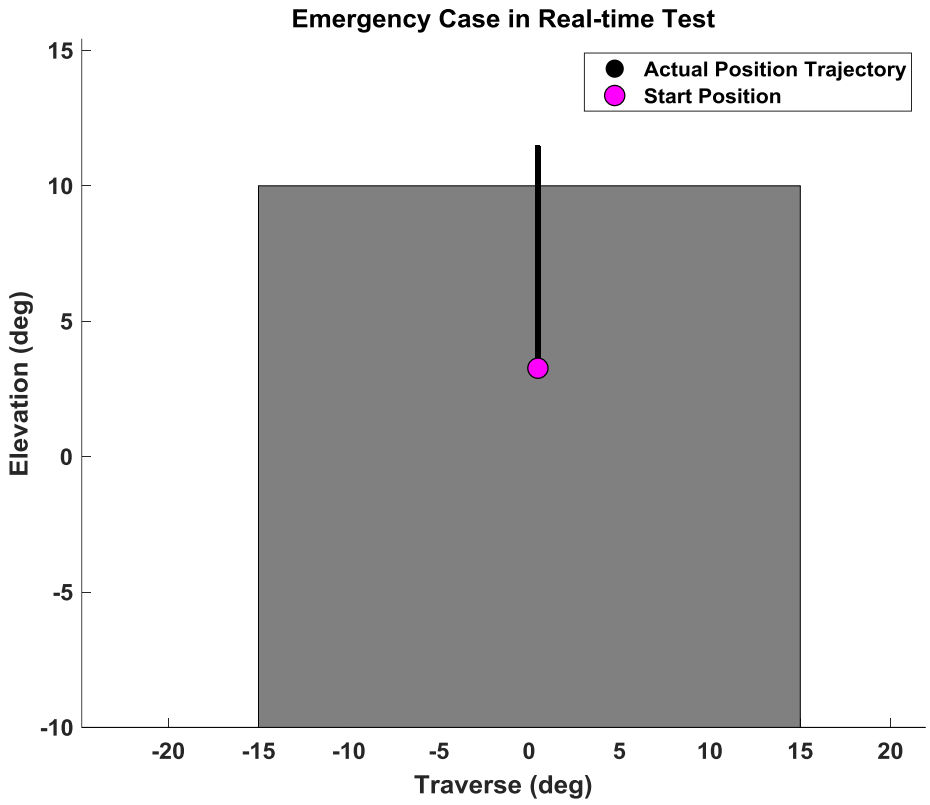

Figure 4.35. Real-time test of emergency case

4.3.2 Experiments with Different Step Speed Commands

In order to see the performance of the system under different speeds in real-time tests, tests are carried out by giving the same speed commands used for the simulations in Figure 4.18. While constant commands are given to the traverse axis, the speed commands for the elevation axis are created in the form of steps in order to avoid upper and lower-hang obstacles. The results obtained by giving the speed commands in Figure 4.18 are as in Figure 4.36. As can be seen, in all of the simulations made with 7 different speeds, the turret does not collide with the obstacles and continues its movement. When positive speed commands are given to the traverse axis, the turret approaches all obstacles from their left.

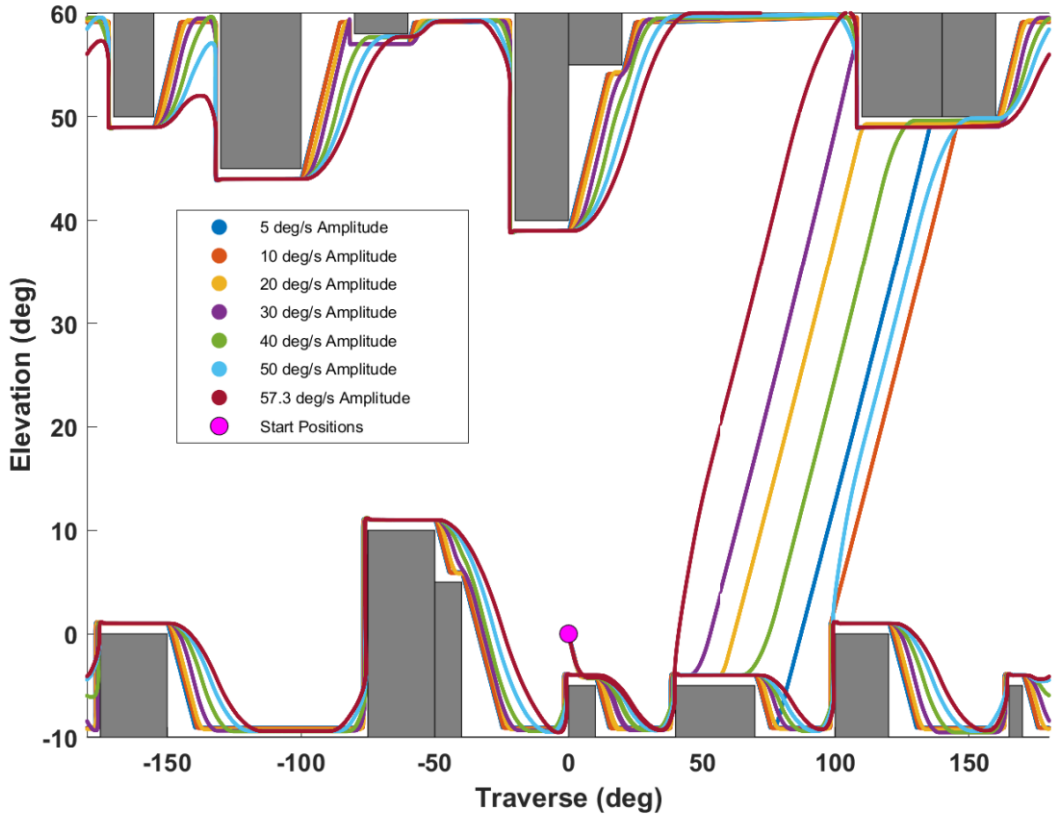

Figure 4.36. Actual position trajectories of turret for 7 different speed commands in real-time tests

In order to see the circumference of the upper two adherent obstacles seen in Figure 4.36, a zoomed view is given in Figure 4.37. As mentioned, while explaining the algorithm, the radius formed at different speeds under the same acceleration is different and this radius increases as the speed increases.

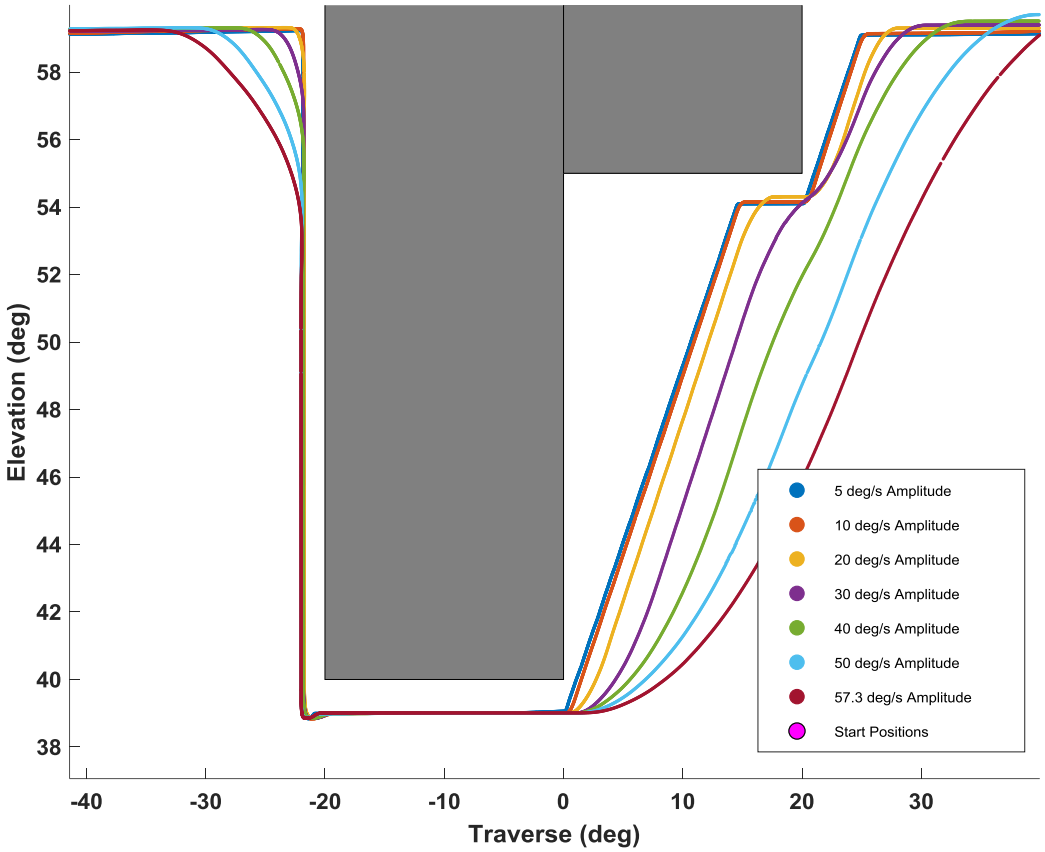

Figure 4.37. The circumference of the upper two adjoining obstacles for positive traverse speed command in real-time tests

To show how the speed commands given according to the obstacles and acceleration limits are shaped by the algorithm, the results with only $20°/s$ are given in Figure 4.38. The shaped speed commands are applied to the system. While real-time speeds are taken from the gyroscope on the system, the positions are read from the absolute encoders in both axes.

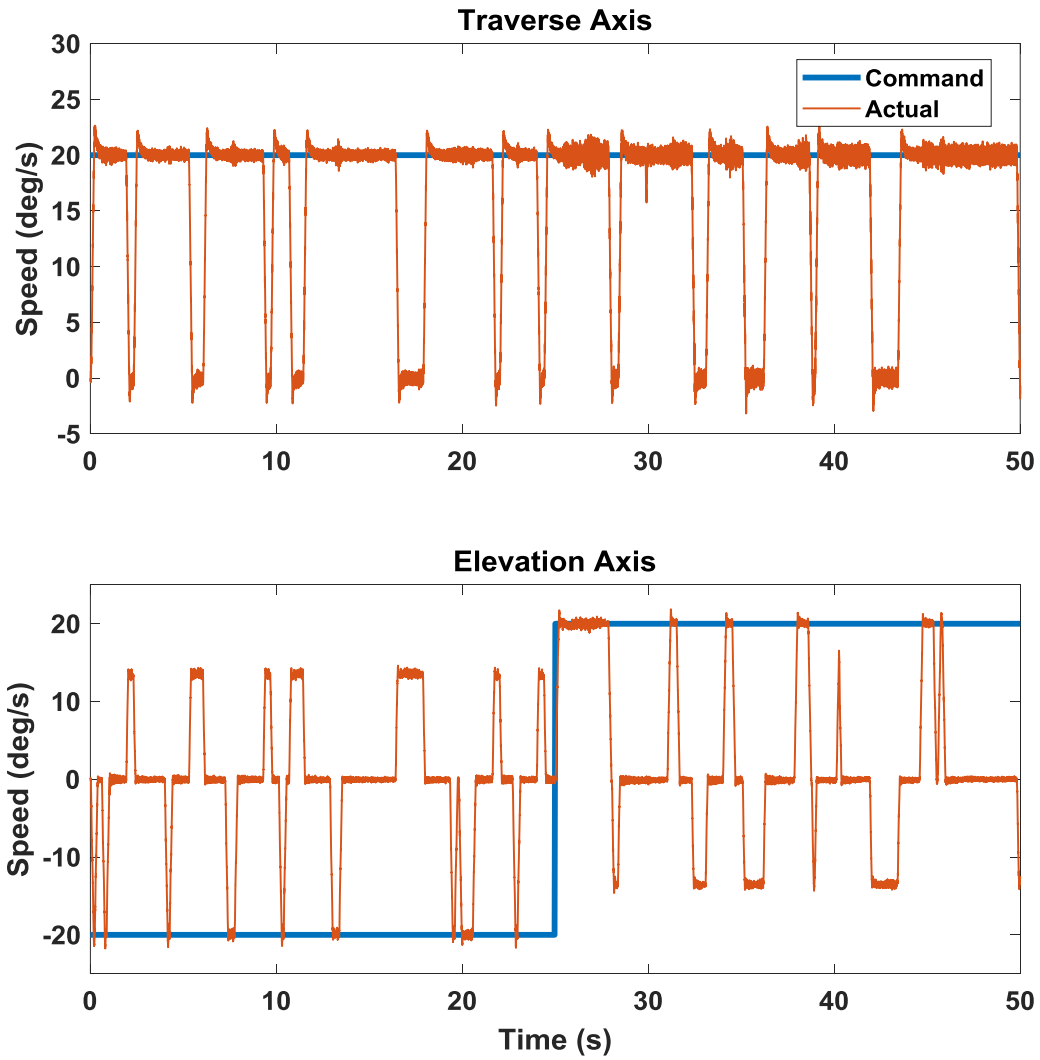

Figure 4.38. Speed commands and actual speed of two axes for 20°/s speed commands in real-time tests

When given the speed commands in Figure 4.18, the turret approaches all obstacles from their left. In order to test the approach of obstacles from their right, the elevation command in Figure 4.18 is kept the same and only the traverse axis commands are given as negative and then the tests are repeated. In this case, the turret did not collide with any obstacles. In order to see the circumference of the upper two adherent obstacles seen in Figure 4.39, a zoomed view is given in Figure 4.40.

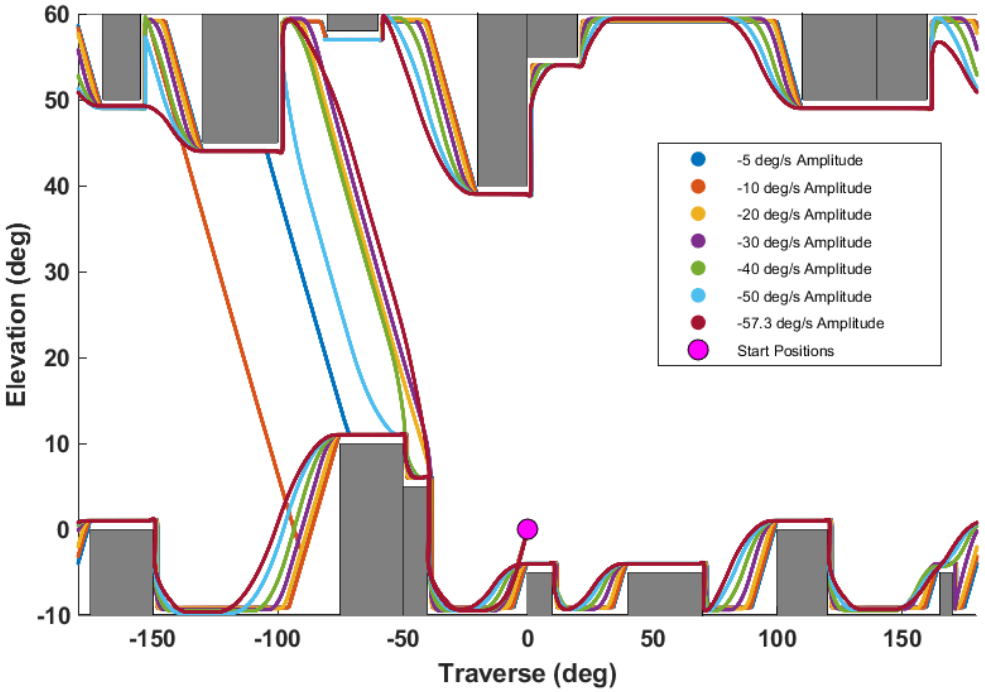

Figure 4.39. Actual position trajectories of turret for 7 different speed commands in real-time tests

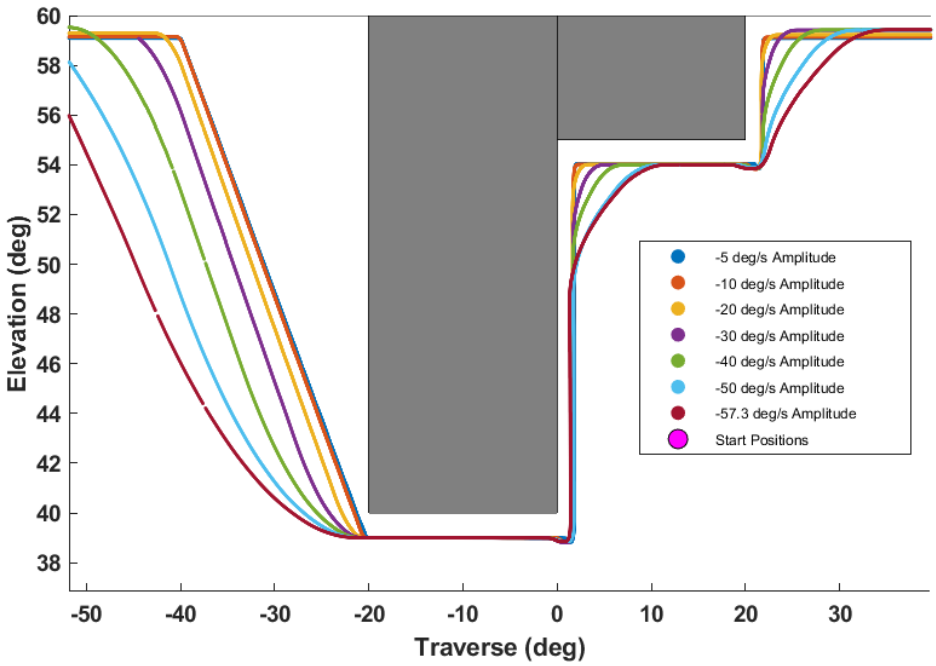

Figure 4.40. The circumference of the upper two adjoining obstacles for negative traverse speed command in real-time tests

To show how the speed commands given according to the obstacles and acceleration limits are shaped by the algorithm, the results with only $-20°/s$ are given in Figure 4.41.

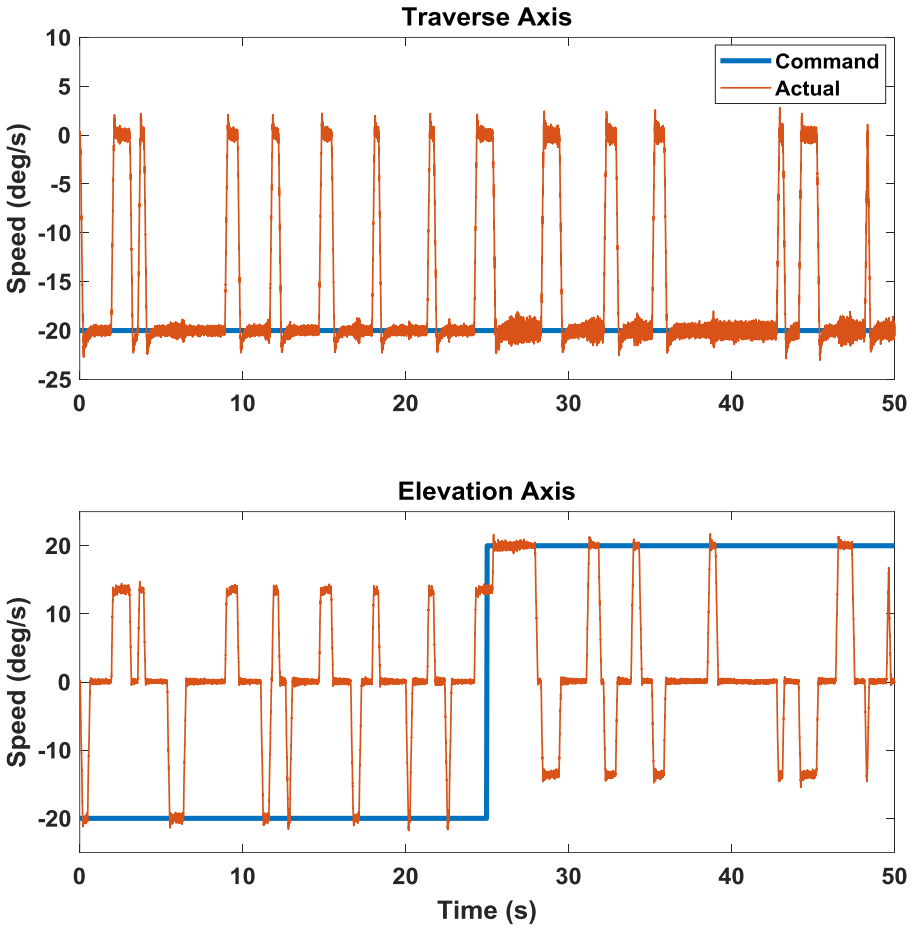

Figure 4.41. Speed commands and actual speed of two axes for -20°/s speed commands in real-time tests

4.3.3 Experiments with Noisy Custom Speed Commands

In order to see the performance of the algorithm under the noisy speed commands in real-time, the custom noisy speed commands in Figure 4.27 are given to the traverse and elevation axes and tests are carried out. The results obtained after the test are as in Figure 4.42. In order to see the circumference of the upper two adherent obstacles seen in Figure 4.42, the zoomed view is given in Figure 4.43.

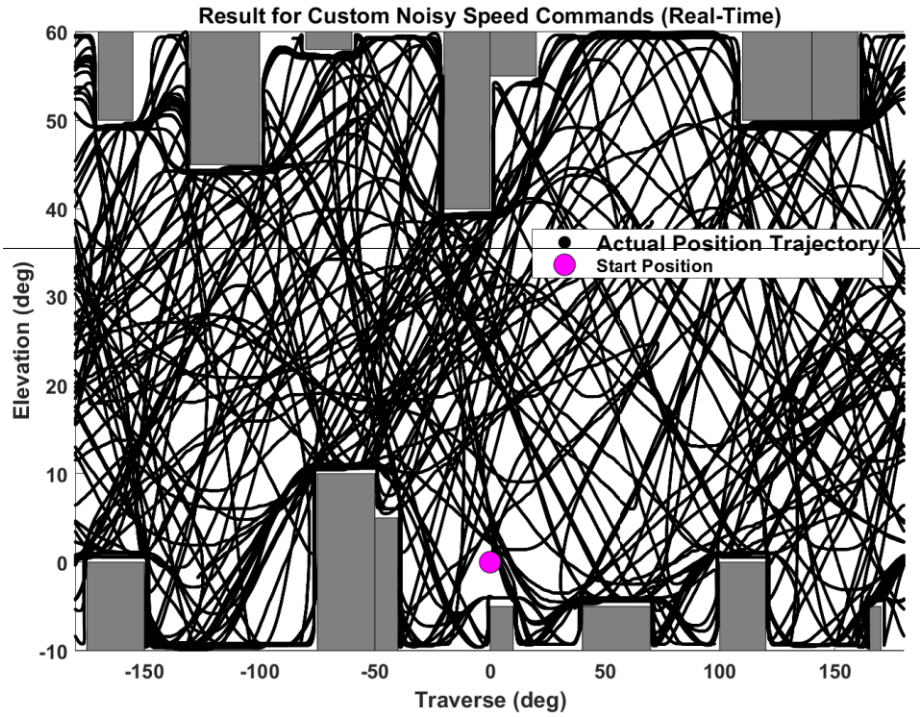

Figure 4.42. Actual position trajectories for custom noisy speed commands in real-time tests

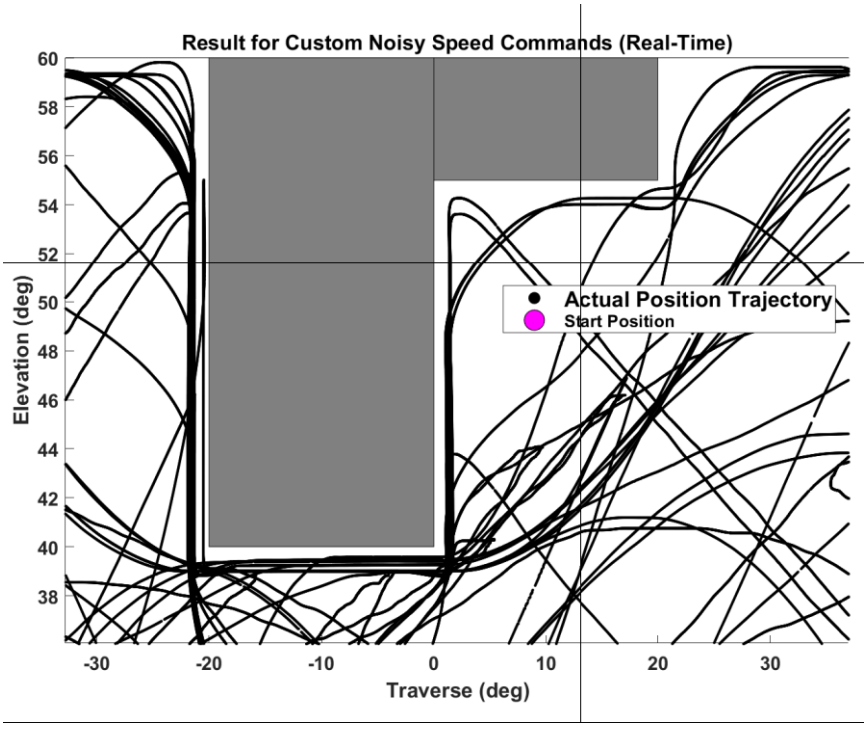

Figure 4.43. The circumference of the upper two adjoining obstacles for custom noisy speed commands in real-time tests

4.3.4 Experiments of Position Control for Path Planning

As explained in the simulation part in Section 3.4, position control is also performed with the same algorithm. Thus, it is possible to make a road plan without collisions between two different locations. The location control verified on the simulation is also verified with the help of experiments. Real time position trajectory without collision between 4 different target points from the same starting point is shown in Figure 4.44. Accordingly, to go to the target, firstly the CW or CCW direction is determined and then the target is reached without collision.

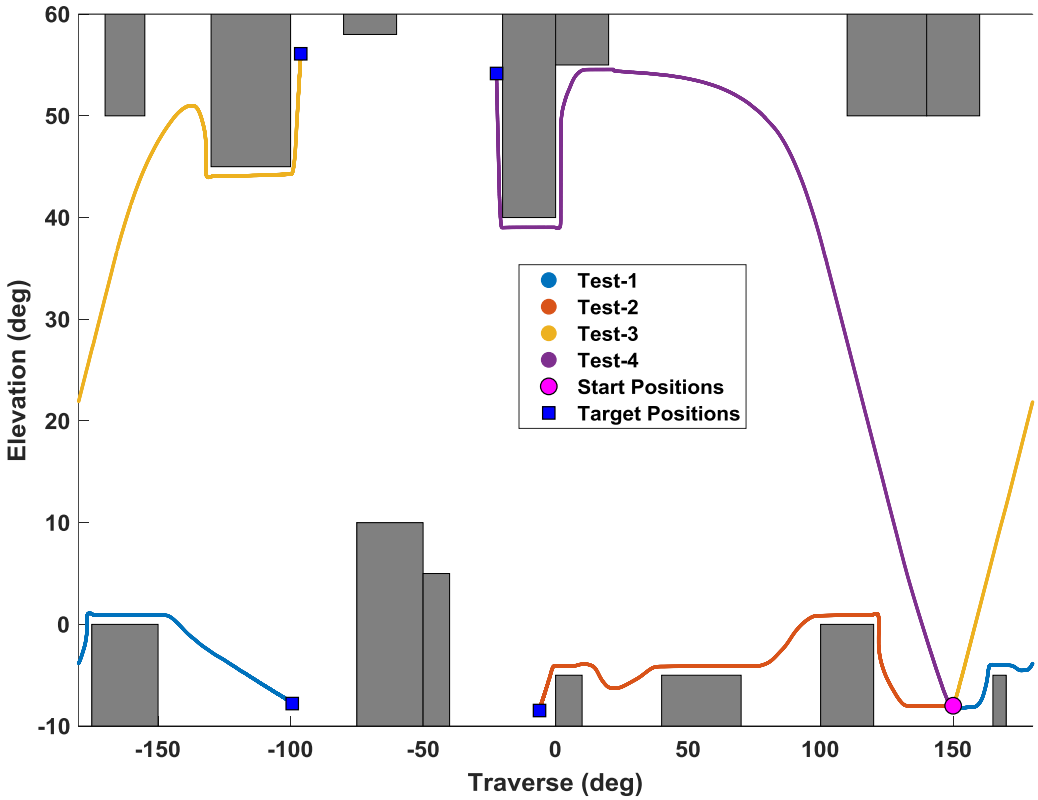

Figure 4.44. Some path planning examples in real-time test

4.4 Collision Avoidance for Multiple Turret System

Collision free manual driving of the single turret is explained in detail in Section 4.1. A collision avoidance algorithm for manual driving has been proposed and validated by simulations and real-time tests. The simulation and real-time test results of the proposed algorithm for manual driving of the single turret are explained in sections 4.2 and 4.3, respectively. We can transform the work done for a single turret into a form that can be applied to multiple turrets in 2 different ways. One of the options is to select the turret that will be driven manually among multiple turrets as the master, and the other turrets that will remain stationary, to choose the slave. Another option is to have all turrets driven manually at the same time.

4.4.1 Choosing Master and Slaves

The double turret in Figure 2.14 can be taken as an example. First of all, the 4-D configuration space of the double turret is obtained as in Figure 2.15 and Figure 2.16 which consist all configurations of double-turret system. By using the existing traverse and elevation positions of the turret selected as slave, the 2-D section of the 4-D configuration space is taken and the instant 2-D configuration space of the master turret is obtained. Turret-2 is selected as the master and if the elevation and traverse angles of the turret-1 are taken as 16 and 120 degrees, respectively, the configuration space of the turret-2 is as in Figure 4.45.

After the 2-D space of the turret-2 is obtained, grid-based obstacles in space must be grouped and transformed into rectangular shaped obstacles. By using the Algorithm-4.5, obstacles in a given configuration space can be grouped and a rectangle covering the grouped obstacles can be obtained.

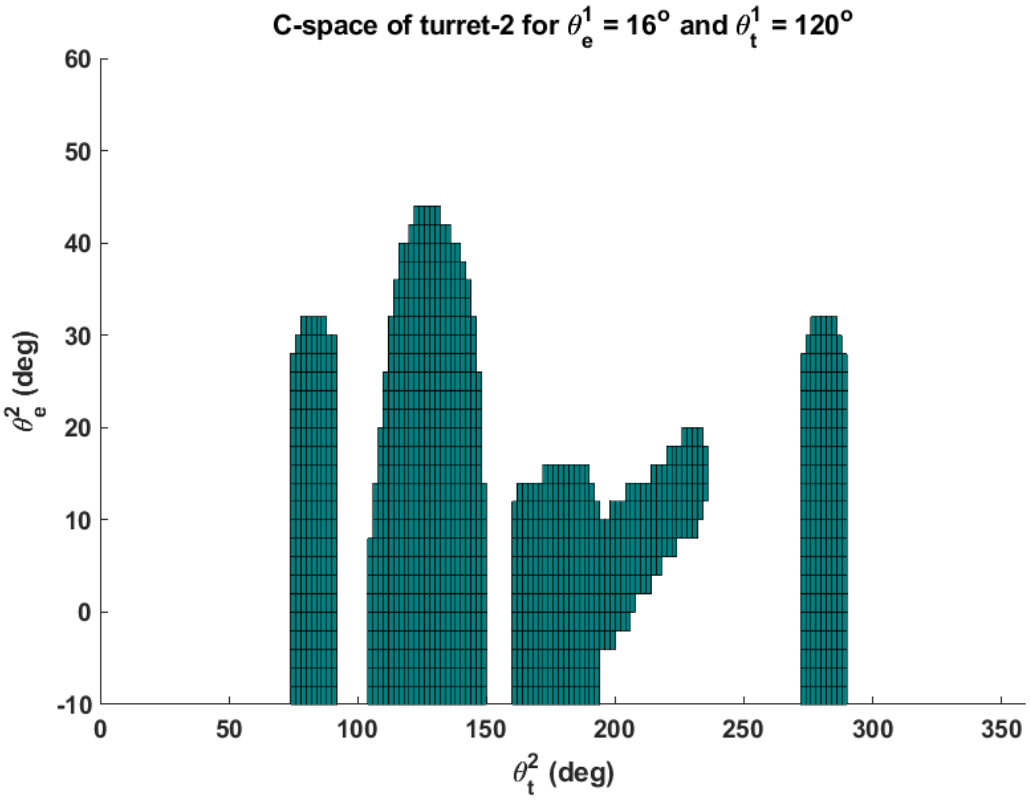

Figure 4.45. C-space of turret-2 while elevation and traverse angles of turret-1 are 16° and 120°, respectively.

The ObstacleGrouping algorithm (Algorithm-4.5) starts by initializing 6 inputs; $MAP2D, n_o, s_d, g_s, UL, LL$ which represent 2-D configuration space, number of obstacles, safe distance given as grid, increase in safe distance, upper limit of elevation axis and lower limit of elevation axis, respectively. Firstly, empty n_o-elements Obs and $status$ arrays are created in Line-1. The while loop is then started and returned until success is achieved. In Line-3, the coordinates of disabled nodes in the given MAP2D configuration space are assigned to p. Next, an inner while loop is started that satisfies the condition p is not empty in Line-4. In line-5, the first element of p array is assigned to k and equal to u. Again, another inner while loop is started that satisfies the condition k is not empty in Line-6. Between Line-7 and Line-8, the first element of k is assigned to f_k and this element is removed from the array k. The neighbors within s_d distance from f_k node are assigned to z in Line-9.

Between Line-10 and Line-14, the elements in z that have the element of p are added to the queue of k and u arrays and the first element of the array k is removed. When there are no more elements to be removed in array k, all the elements in the u array are added as a group to the y set as an obstacle group, and the number of obstacle groups found (o) is increased by one in Line-15. If the total number of obstacle groups is set as n_o and two of them are reserved for the upper and lower limit of the elevation axis (UL, LL) which are defined in Line-22, the allowed additional obstacle number is $(n_o - 2)$. The condition of "if statement" in Line-16 breaks the while loop if the number "o" is greater than $(n_o - 1)$, since the number "o" is already incremented in an upper Line. If there is no remaining element in the p array between Line-17 and Line-18 and the number of obstacle groups found (o) is equal to or greater than $(n_o - 1)$, this loop is successful and the top while loop ends. If it is not successful, the safe distance (s_d) is increased by the grid step (g_s) and the while loops are repeated. Between Line-19 and Line 20, the obstacle sets found are surrounded by a minimum axis-aligned rectangle. Then these rectangles are connected to that axis whichever is closer to the upper or lower limits of the elevation axis. Between Line-23 and Line-24, the status of the obstacles is determined by looking at the areas of the rectangles surrounding them. Finally, full n_o-elements Obs and $status$ arrays are obtained in Line-25.

Algorithm-4.5: ObstacleGrouping

In: $MAP2D, n_o, s_d, g_s, UL, LL$

Out: $Obs(n_o), status(n_o)$

1 $success = 0$, $Obs = zeros(n_o)$, $status = zeros(n_o)$,

2 **while** $success = 0$

3 $p = find(MAP2D == 1)$, $o = 1$

4 **while** p *is not empty*

5 k ← assign first element of p array, $u = k$

6 **while** k *is not empty*

7 f_k ← assign first element of k array

8 delete f_k from k array

9 $z = \boldsymbol{findNeighbor}(f_k, s_d)$

10 **for** $i = 1 \ to \ [(2s_d + 1)^2 - 1] \rightarrow number \ of \ neighbors$

11 **if** i^{th} element of z $(z_i) \in p$

12 z_i element is added to the queue of k and u arrays

13 z_i element is deleted from p array

14 delete first element of k array

15 $y\{o\} = u; \ o = o + 1;$

16 **if** $o \geq (n_o - 1) \rightarrow \boldsymbol{break};$

17 **if** $o \geq (n_o - 1)$ & $(p \ is \ empty) \rightarrow success = 1;$

18 **else** $\rightarrow sd = sd + g_s;$

19 **for** $j = 1 \ to \ \ of \ obstacles \ in \ found \ (y)$

20 $Obs_{temp(j)} = rectangle \ (y\{j\})$

21 $Obs(j)$ =Link the $Obs_{temp(j)}$ to the upper or lower limit (UL, LL)

22 $Obs(n_o - 1) = rectangle \ (UL), Obs(n_o) = rectangle \ (LL)$

23 **for** $r = 1 \ to \ \ n_o \ \rightarrow \# \ of \ obstacles$

24 **if** $area \ of \ rectangle \ of \ Obs(r) \neq 0 \ \rightarrow status(r) = 1$

25 **return** $Obs \ \ and \ status$

As can be seen roughly in Figure 4.45, there are 6 obstacles in total, one for each of the upper and lower limits. If we want to group the obstacles on the map in Figure 4.45 using the ObstacleGrouping algorithm, it is necessary to first determine how many obstacles can be defined at most. If this number is 6 and above, the obstacles are obtained as in Figure 4.46. The obstacle groups that occur when we reduce the number of obstacles allowed one by one are given in order as shown in Figure 4.47, Figure 4.48 and Figure 4.49.

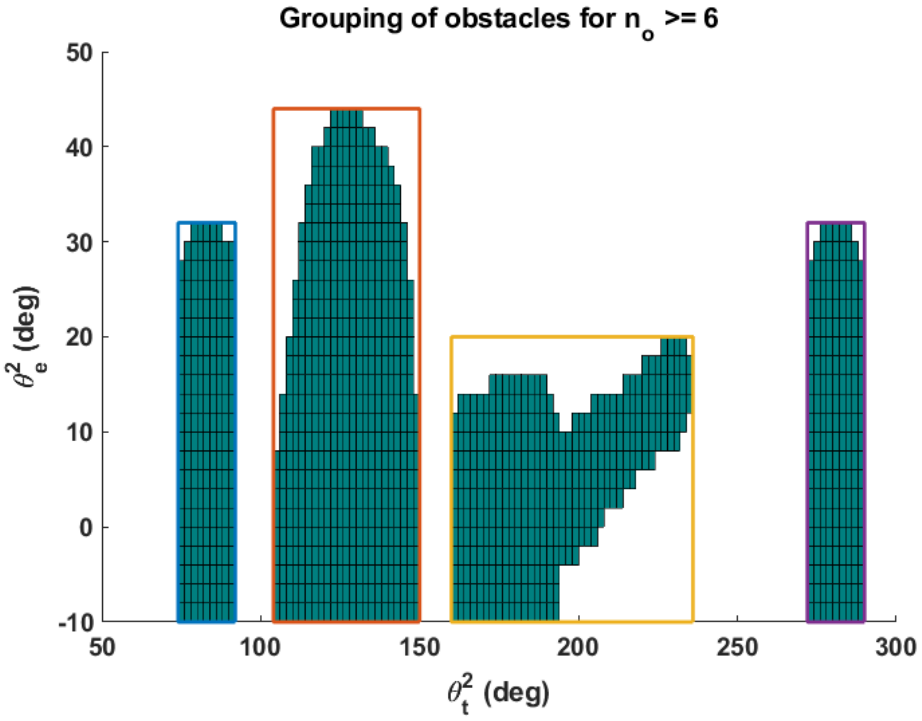

Figure 4.46. Grouping of the obstacles for n_o is equal or greater than 6

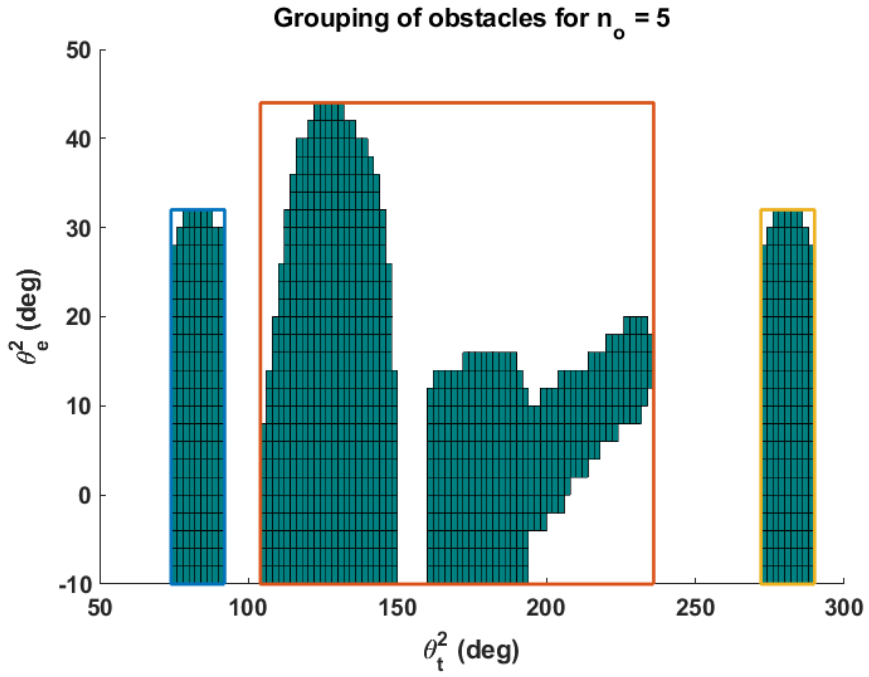

Figure 4.47. Grouping of the obstacles for n_o is equal to 5

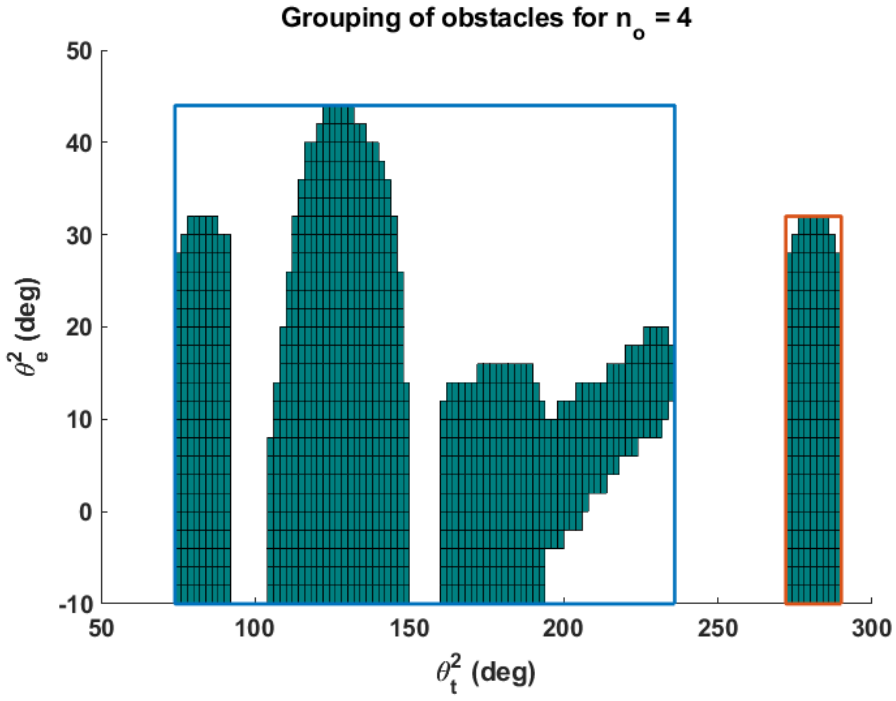

Figure 4.48. Grouping of the obstacles for n_o is equal to 4

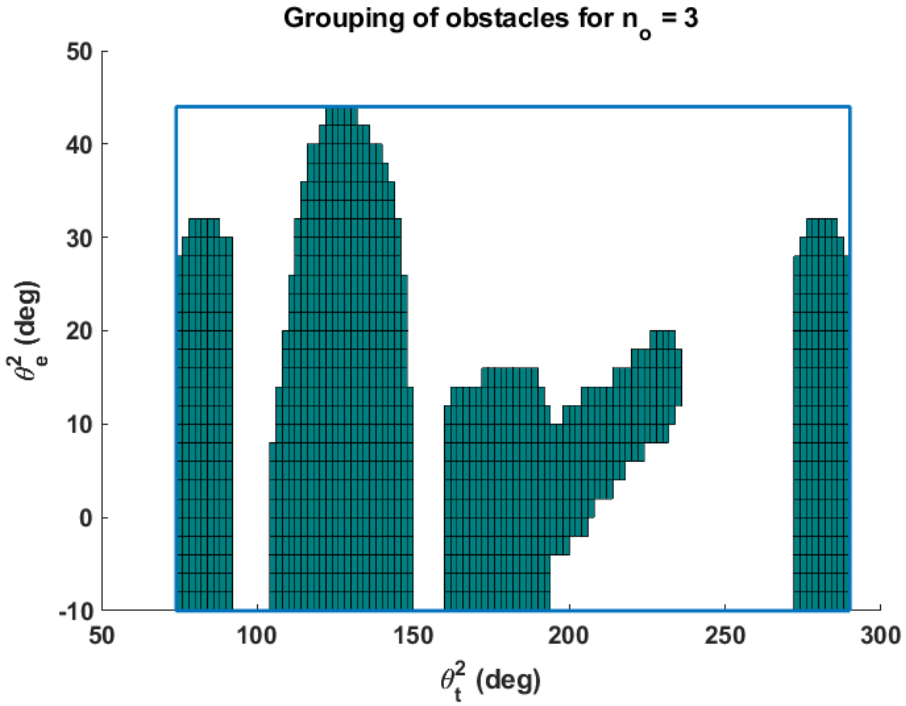

Figure 4.49. Grouping of the obstacles for n_o is equal to 3

To test the master-slave option for the double-turret system, the simulation model of the system was created using MATLAB®-Adams® Control as shown in Figure 4.50. Details of decision block are given in Figure 4.51.

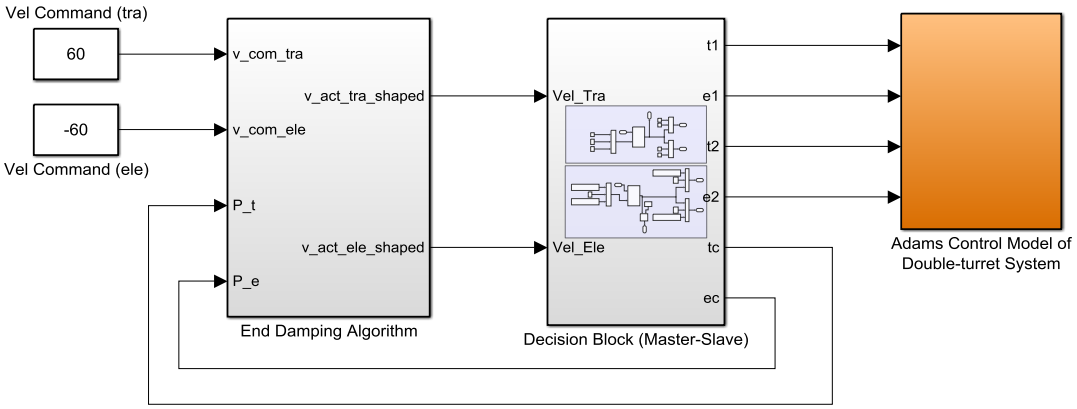

Figure 4.50. MATLAB®-Adams® Control model of double-turret system

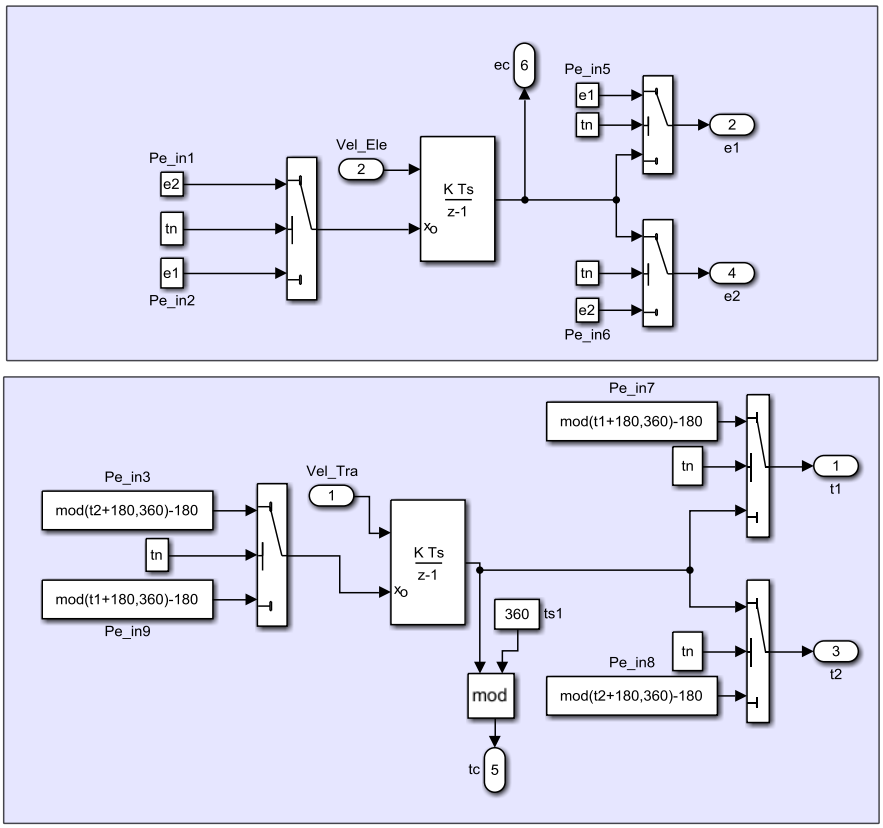

Figure 4.51. Inside of Decision Block (Master-Slave)

The traverse and elevation angles of the Turret-1 were kept 120 and 16 degrees, respectively, and the Turret-2 was chosen and driven as the master. The number of obstacles allowed is 6. In this case, the obstacles will be grouped as in Figure 4.46. The initial angles of the traverse and elevation axes of Turret-2 were selected as 200 and 21 degrees, respectively, and the speed commands of $60°/s$ and -$60°/s$ were given to the traverse and elevation axes, as in Figure 4.50, and the simulation was run for 18 seconds. Accordingly, the movement of Turret-2 in C-space is as follows. As a result, no collision has occurred.

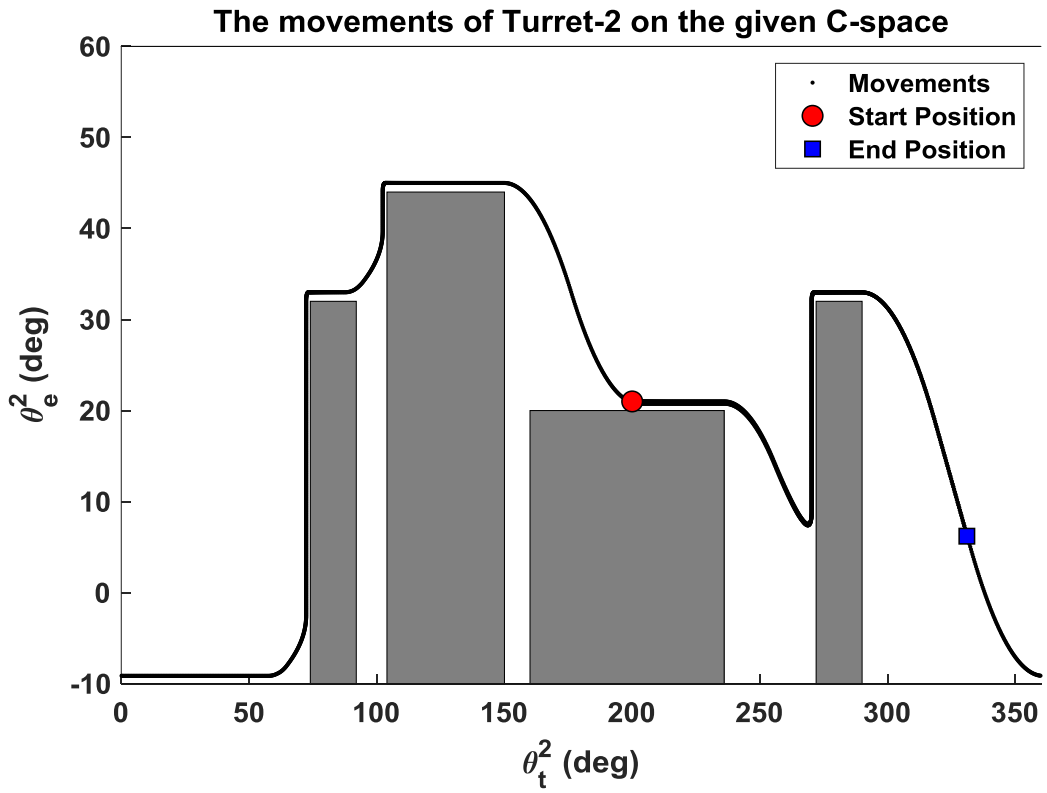

Figure 4.52. The movements of Turret-2 on the given C-space

4.4.2 Instant Multi-driving

In section 4.4.1, manual driving by selecting one of the multi-turret systems as a master is discussed. Master slave changes can be made if desired. Thus, instantaneous high calculations are avoided. However, in systems where calculation

times are not limited and with high processing power, multiple turrets can be driven manually together. In this case, instead of selecting a master slave, it is necessary to obtain the instantaneous configuration space of another using the position of one turret, then group the obstacles in this space. Once these are done, the problem will be as described in section 4.4.1.

4.5 Conclusion

Within the scope of this work, a new collision avoidance algorithm is proposed for a more efficient and safe use of all types of turret systems. With the help of this algorithm, the speed and position requests from the user can be responded according to the specified speed, acceleration and jerk limits. Obstacles with the possibility of collision are avoided within the framework of these limits. Even in situations such as avoiding obstacles, deceleration and acceleration, if the commands new to the user do not cause a collision, the algorithm starts to apply the new commands which provides flexibility to the user. In addition, possible worst scenarios for turret systems are determined one by one, and the efficiency of the algorithm in these worst scenarios and its overcoming is shown by both simulations and real-time tests. Later, a C-space where worst scenarios can occur is created for the performance measurement of the algorithm, and the same space is used in all tests. By giving different speed commands in the specified C-space, the performance of the algorithm at different speeds is observed on the stabilized gun turret system both in simulations and real time tests. For the measurement of the performance under the noisy speed commands, a custom noisy speed command of about 1000 seconds is created and both simulation and real-time tests are performed. As a result of these tests, it is shown that there is no collision. Finally, by controlling the position, the departure from the starting point to the desired target point is achieved without any collision.

The most important feature that distinguishes this algorithm from the others is that both speed and position can be controlled and during transition phase, the target point can be changed instantly. The algorithm can be integrated into turrets of all sizes,

robotic systems with minimal or no change. Since the Algorithm does not intervene in the speed and torque loops in contrast to potential field-based methods, it can be added to ready-to-use systems by manipulating only the speed references. It is obvious that this will provide a great advantage. Since most obstacle avoidance algorithms on the market are potential field based, even the torque loop needs to be intervened, it is not possible to do this in closed box systems. This is the biggest increase of the algorithm that is propagated.

CHAPTER 5

SUMMARY AND CONCLUSION

In the first part of this study, the developed approach with two different methods to obtain high-dimensional configuration space is explained in detail where one of the methods is obtaining by using point clouds and the second one is using only simulation software. These two methods produce a 4-D configuration space for a double-turret device, which can be used to verify the process and compare methods. As a result, the difference between these two methods is around 1%, depending on the density of the point cloud. The disparity between the two forms steadily decreases as the point cloud density increases. By the way, the first method is 8.1 times more effective than the second, which is one of the first method's advantages. The requirement to build a point cloud for each component is seen as a drawback of the first method. For the second method, however, 3-D cad models of parts and appropriate input profiles are all that is needed to complete the study. Using the previously obtained 4-D configuration space, a sample path planning with the A* algorithm was created at the end of the analysis.

In the second part of this study, a 4-D path planning algorithm is proposed which attempts to search a path on three different types of C-Spaces named as rectangular, circular and torus shaped in three converging options which are fast, medium and optimum depending on the application. So that, a collision free motion planning can be carried out for double-turret system operating in a common workspace. After obtaining 4-D C-space of double-turret system by using the method studied in first part, with the help of the proposed algorithm, 4-D path planning problem was realized as 2-D + 2-D by using six sequences and their options. The sequence and options of these sequences are given in an acceptable order thanks to random

simulations, and the proposed algorithm is also tested on different input/output combinations. A sample path planning was created to test the algorithm's output and, as a result, the converging options. The results obtained for three different converging options were simulated on the model of the double-turret system and it was observed that there was no collision between any bodies.

Thus, using this 4-D path planning algorithm, aligned multiple turrets in the same environment or any aligned multiple systems with two degrees of freedom each can work without colliding with each other. With this study, more turrets can be integrated on a single platform for military applications. Also, since manual aiming tasks of multiple-turret system will be autonomous, operation time will be seriously reduced and collision risk will disappear. In addition, the mobility of the system due to its full rotational axes will be preserved by using circular and torus shaped C-Spaces.

In the last part of this study, manual driving of multiple-turret system is studied. A new collision avoidance algorithm named “End Damping Algorithm” is proposed for a manual driving of turrets. The user's speed and position requests can be responded to using this algorithm based on the given speed, acceleration, and jerk limits. Obstacles that have the potential to collide are avoided under these parameters. Furthermore, potential worst-case scenarios for turret systems are identified one by one, and the algorithm's efficiency in resolving these worst-case scenarios is demonstrated by simulations and real-time tests. The algorithm's output is then evaluated using a C-space, which simulates the worst-case scenarios. These experiments have shown that there is no collision. Finally, by monitoring the position, a collision-free transition from the starting point to the desired target point is achieved.

The most important aspect that sets the End Damping Algorithm apart from other collision avoidance algorithms is that both speed and location can be monitored, and the target point can be modified instantly during the transition process. With minor to no changes, the algorithm can be implemented in turrets of all sizes and robotic

systems. Since, unlike possible field-based approaches, the algorithm does not interfere in the speed and torque loops, it can be applied to ready-to-use applications by simply changing the speed references. This will, without a doubt, be a significant benefit. Since most obstacle avoidance algorithms on the market are focused on potential fields, even the torque loop must be intervened, which is impossible in closed box systems. This is the most significant improvement in the algorithm.

The algorithm is also thought to be useful for autonomous vehicles and can be extended to all related 2-DOF systems to satisfy collision avoidance requirements. The algorithm could be turned into a crash avoidance algorithm for autonomous vehicles in the future. In this case, the lower and upper limits of the elevation axis are built as road barriers in the configuration space, and the obstacles are constructed as vehicles; this can be accomplished by making minor changes to the algorithm.

REFERENCES

[1] M. Bahrin, M. Othman, N. H. N. Azli and M. F. Talib, "Industry 4.0: A review on industrial automation and robotic," *Jurnal Teknologi (Sciences and Engineering)*, pp. 137– 143, 2016.

[2] F. Padula and V. Perdereau, "An on-line path planner for industrial manipulators," *International Journal of Advanced Robotic Systems*, January 2013.

[3] S. Yoon, H. Do and J. Kim, "Collaborative mission and route planning of multi-vehicle systems for autonomous search in marine environment," *Int. J. Control Autom. Syst.*, vol. 18, no. 3, pp. 546–555, 2020.

[4] M. Reuter, H. Oberc, M. Wannöffel, D. Kreimeier, J. Klippert, P. Pawlicki and B. Kuhlenkötter, "Learning factories' trainings as an enabler of proactive workers participation regarding industrie 4.0," *Procedia Manufacturing*, pp. 354– 360, 2017.

[5] Y. Liu, C. Yu, J. Sheng and T. Zhang, "Self-collision avoidance trajectory planning and robust control of a dual-arm space robot," *Int. J. Control Autom. Syst.*, vol. 16, pp. 2896–2905, 2018.

[6] SO. Park, M.C. Lee and J. Kim, "Trajectory planning with collision avoidance for redundant robots using jacobian and artificial potential field-based real-time inverse kinematics," *Int. J. Control Autom. Syst.*, vol. 18, pp. 2095–2107, 2020.

[7] The requirements for future military robots supporting mobility relevance and possible future role of robotic/unmanned systems for FINABEL land forces. *European land forces interoperability center FINABEL*, 2013.

[8] "Multiple cradle launcher," *Roketsan Missiles Inc*., www.roketsan.com.tr/wpcontent/uploads/2013/05/IDEX-1.pdf (last accessed: April-2020).

[9] Implementing multi-turret and twin-barrel support with a 3rd soviet heavy line, 2015, Retrieved from http://ritastatusreport.blogspot.com/2015/12/implementing-multi-turret-and-twin_16.html.

[10] J. Zhao, Y. Chao and Y. Yuan, "A Cooperative Obstacle-Avoidance Approach for Two-Manipulator Based on A* Algorithm," *Intelligent Robotics and Applications. ICIRA 2019. Lecture Notes in Computer Science*, vol 11745. Springer, Cham, 2019.

[11] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki and S. Thrun, "Principles of Robot Motion: Theory, Algorithms, and Implementations [Book Review]," *IEEE Robotics & Automation Magazine*, 2005.

[12] J. Pan and D. Manocha, “Efficient Configuration Space Construction and Optimization for Motion Planning,” *Engineering, transactions of CAE*, vol. 1, no. 1, Pages 46–57, 2015.

[13] O. Khatib, “Real-Time Obstacle Avoidance for Manipulators and Mobile Robots,” *in Proceedings - IEEE International Conference on Robotics and Automation*, 1985.

[14] T. Lozano-Pérez and M. A. Wesley, “An algorithm for planning collision-free paths among polyhedral obstacles,” *Commun of the ACM*, vol. 22, no. 10, pp. 560-570, 1979.

[15] T. Lozano-Pérez, “Automatic planning of manipulator transfer movements,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 11, no. 10, pp. 681–98, October 1981.

[16] T. Lozano-Perez, “Spatial planning: A configuration space approach,” *IEEE Transactions on Computers*, vol. 32, no. 2, pp. 108-120, February 1983.

[17] J. Pan and D. Manocha, “Efficient configuration space construction and optimization for motion planning,” *Engineering*, vol. 1, no 1, pp 46–57, 2015.

[18] H. Choset and J. Latombe, “Principles of robot motion: theory, algorithms, and implementations [Book Review],” *IEEE Robotics & Automation Magazine*, vol. 12, 2005.

[19] E. Freund and H. Hoyer, “Real-time path finding in multi-robot systems including obstacle avoidance,” *Int J Robotics*, vol. 7, no. 1, pp. 42–70, 1988.

[20] Y. Fei, D. Fuqiang and Z. Xifang, “Collision-free motion planning of dual-arm reconfigurable robots,” *Robot.Comput. Manuf.*, vol. 20, no. 4, pp. 351–357, 2004.

[21] S. Huang, R. S. H. Teo and K. K. Tan, “Collision avoidance of multi unmanned aerial vehicles: A review,” *Annual Reviews in Control*, vol. 48, pp. 147 -164, 2019.

[22] Z. Shi, J. He, T. Wang, C. Zhou and J. Guo, “Optimal formation control and collision avoidance in environment with multiple rectangle obstacles,” *Journal of the Franklin Institute*, 2018.

[23] N. Sariff and N. Buniyamin, “An overview of autonomous mobile robot path planning algorithms,” *in Proc of the 4th Student Conference on Research and Development*, pp.183-188, 2006.

[24] J. Cui, Y. Zhang, S. Ma, Y. Yi, J. Xin and D. Liu, “Path planning algorithms for power transmission line inspection using unmanned aerial vehicles,” *In Proc. of Control and Decision Conference*, pp.2304–2309, 2017.

[25] C.W. Warren, “Fast path planning method using modified A* method”, *in Proc. of IEEE International Conference on Robotics and Automation*, Atlanta, USA, pp.662–667, 1993.

[26] T. Lozano-Perez, “Spatial planning: A configuration space approach,” *IEEE Transactions on Computers*, C-32(2), pp.108–120, 1983.

[27] J. Borenstein and Y. Koren, “The vector field histogram - fast obstacle avoidance for mobile robots,” *Journal of Robotics and Automation*, vol. 7, no. 3, pp. 278-288, 1991.

[28] O. Khatib, “Real-time obstacles avoidance for manipulators and mobile robots," *in International Conference on Robotics and Automation*, March, pp. 500-505, 1985.

[29] D. Fox, W. Burgard, and S. Thrun, “The dynamic window approach to collision avoidance,” *IEEE Robotics Automation Magazine*, vol. 4.1, pp. 23-33, 1997.

[30] H. Myre, “Collision Avoidance for Autonomous Surface Vehicles Using Velocity Obstacle and Set-Based Guidance,” *Master Thesis, Norwegian University of Science and Technology*, 2016.

[31] P. Fiorini and Z. Shiller, “Motion planning in dynamic environments using velocity obstacles,” *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760-772, 1998.

[32] A. Benzerrouk, L. Adouane and P. Martinet “Dynamic Obstacle Avoidance Strategies using Limit Cycle for the Navigation of Multi-Robot System,” *IEEE/RSJ IROS’12, 4th Workshop on Planning, Perception and Navigation for Intelligent Vehicles*, Vilamoura, Algarve, Portugal, 2012.

[33] Xu, Y. Hu, J. Zhai, L. Li, and P. Guo, ‘‘A novel non-collision trajectory planning algorithm based on velocity potential field for robotic manipulator,’’ *Int. J. Adv. Robot. Syst.,* vol. 15, no. 4, pp. 1–13, Jul. 2018.

[34] J. van den Berg, J. Snape, S. J. Guy and D. Manocha, "Reciprocal collision avoidance with acceleration-velocity obstacles," *IEEE International Conference on Robotics and Automation*, Shanghai, pp. 3475-3482, 2011.

[35] Z. Yan, J. Li, G. Zhang, and Y. Wu, “A real-time reaction obstacle avoidance algorithm for autonomous underwater vehicles in unknown environments,” *Sensors Basel*, vol. 18, p. E438, 2018.

[36] C. Ackermann, J. Bechtloff and R. Isermann, “Collision avoidance with combined braking and steering,” *In: Pfeffer P. (eds) 6th International Munich Chassis Symposium. Proceedings. Springer Vieweg*, Wiesbaden, 2015.

[37] M. Schorn and R. Isermann “Automatic steering and braking for a collision avoiding vehicle,” *In:4th IFAC symposium on mechatronic systems, Heidelberg*, pp.378–383, 12–14 September 2006.

[38] D. F. Llorca, V. Milanes, I. P. Alonso, et al. “Autonomous Pedestrian Collision Avoidance Using a Fuzzy Steering Controller,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 390–401, 2011.

[39] M. Brannstrom, J. Sjoberg, L. Helgesson and M. Christiansson, “A real-time implementation of an intersection collision avoidance system,” *IFAC Proc*., vol. 18, part 1, pp. 9794-9798, 2011.

[40] W. Ben-Messaoud, M. Basset, J. Lauffenburger and R. Orjuela, “Smooth Obstacle Avoidance Path Planning for Autonomous Vehicles,” *IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pp. 1-6, Madrid, 2018.

[41] J. Cho, D. Pae, M. Lim and T. Kang, “A Real-Time Obstacle Avoidance Method for Autonomous Vehicles Using an Obstacle-Dependent Gaussian Potential Field,” *Journal of Advanced Transportation*, vol. 2018, Article ID 5041401, pp. 1-15, 2018.

[42] B. Sultan and M. McDonald, “Assessing the safety benefit of automatic collision avoidance systems (during emergency braking situations),” *In: The 18th International Conference on the Enhanced Safety of Vehicles*, Nagoya, Japan, 2003.

[43] R. Hayashi, J. Isogai, P. Raksincharoensak and M. Nagai, “Autonomous collision avoidance system by combined control of steering and braking using geometrically optimised vehicular trajectory,” *Vehicle System Dynamics: International Journal of Vehicle Mechanics and Mobility*, vol. 50, pp. 151-168, 2012.

[44] X. Li, X. Xu and L. Zuo, “Reinforcement learning based overtaking decision-making for highway autonomous driving,” *2015 Sixth International Conference on Intelligent Control and Information Processing (ICICIP)*, pp. 336-342, Wuhan, 2015.

[45] V. Milanés, D. F. Llorca, J. Villagrá, J. Pérez, C. Fernández, I. Parra, C. González, and M. A. Sotelo, “Intelligent automatic overtaking system using vision for vehicle detection,” *Expert Systems with Applications*, vol. 39, no. 3, pp. 3362–3373, 2012.

[46] J. Qu, Y. Cui and W. Zhu, “Algorithm and Its Implementation of Vehicle Safety Distance Control Based on the Numerical Simulation,” *J. Networks*, vol. 9, pp. 3486-3493, 2014.

[47] Z. Pan, D. Wang, H. Deng and K. Li, “A Virtual Spring Method for the Multi-robot Path Planning and Formation Control,” *Int. J. Control Autom. Syst.*, vol. 17, no. 5, pp. 1272-1282, May 2019.

[48] T. Liski, “3-D collision checking for improving machine operator's spatial awareness,” *(Master Thesis) Aalto University-School of Electrical Engineering*, Finland, 2014.

[49] W. Wu, H. Zhu, X. Zhuang, G. Ma and Y. Cai, "A multi-shell cover algorithm for contact detection in the three-dimensional discontinuous deformation analysis," *Theoretical and Applied Fracture Mechanics*, vol. 72, no. 1, pp. 136–49, 2014.

[50] J. Klein and G. Zachmann, "Point cloud collision detection," *Computer Graphics Forum,* vol. 23, no. 3, pp. 567-576, 2004.

[51] J. Schauer and A. Nüchter, "Collision detection between point clouds using an efficient k-d tree implementation," *Advanced Engineering Informatics*, vol. 29, no. 3, pp. 440–458, 2015.

[52] W. J. Beksi and N. Papanikolopoulos, "A topology -based descriptor for 3D point cloud modeling: Theory and experiments," *Image and Vision Computing*, vol. 88, pp. 84–95, 2019.

[53] J. Han, "An efficient approach to 3D path planning," *Information Sciences*, vol. 478, pp. 318–30, April 2019.

[54] D. Henrich, C. Wurll and H. Worn, "Online path planning with optimal c-space discretization," *Proceedings of the 1998 IEEE/RSJ International Conference on Robots and System*, Victoria, BC, Canada, pp. 1479–84, 1998.

[55] K. H. Kim, S. Sin and W. Lee, "Exploring 3D shortest distance using A* algorithm in unity3d," *TechArt: Journal of Arts and Imaging Science*, vol.2, no. 3, pp. 81-85, August 2015.

[56] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz and S. Thrun, "Anytime Dynamic A*: An Anytime, Replanning Algorithm," *in ICAPS 2005 - Proceedings of the 15th International Conference on Automated Planning and Scheduling*, 2005.

[57] J. J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," *In: Proceedings of IEEE International Conference on Robotics and Automation*, pp. 995–1001, 2000.

[58] L. Kavraki, P. Svestka, J. C. Latombe and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, August 1996.

[59] "Remote controlled weapon systems," Aselsan A.Ş. https://www.aselsan.com.tr/Remote_Controlled_Weapon_Systems_3441.pdf (last accessed: April-2020).

[60] Y. Ting, W. I. Lei, and H. C. Jar, "A path planning algorithm for industrial robots," *Comput. Industr. Engineer*., vol. 42, no. 2-4, pp. 299–308, April 2002.

[61] X. Cui and H. Shi, "A*-based pathfinding in modern computer games," *Int. J. Comput*. Sci. Netw. Secur., vol. 11, pp. 125–130, 2011.

[62] C. Yuan, G. Liu, W. Zhang and X. Pan, “An efficient rrt cache method in dynamic environments for path planning,” *Robot. Auton. Sys*., vol. 131, 103595, September 2020.

[63] A. K. Pamosoaji, M. Piao and K.-S Hong, “PSO-based minimum-time motion planning for multiple vehicles under acceleration and velocity limitations,” Int. *J. Control Autom. Syst*., vol. 17, pp. 2610–2623, 2019.

[64] F. C. Park and K. M. Lynch, “Introduction to robotics: mechanics, planning, and control,” *Cambridge University Press*, United Kingdom, 2017.

[65] F. A. Raheem, and A. A. Hussain, “Applying A* path planning algorithm based on modified c-space analysis,” *Al-Khwarizmi Engineering Journal*, vol. 13, no. 4, pp. 124-136, 2017.

[66] L. Jaillet and J. M. Porta, “Efficient asymptotically-optimal path planning on manifolds,” *Robot. Auton. Syst*., vol. 61, no. 8, pp. 797–807, 2013.

[67] Z. Gao, “On discrete time optimal control: a closed-form solution,” *Proceedings of the 2004 American Control Conference*, Boston, MA, USA, pp. 52-58 vol.1, 2004.

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Yerlikaya, Ümit
Nationality: Turkish (TC)
Date and Place of Birth: 3 December 1990, Ağrı
Marital Status: Married
Phone: +90 543 242 25 11
email: yerlikaya04@gmail.com

EDUCATION

Degree	Institution	Year of Graduation
MS	METU Mechanical Engineering	2016
BS	Hacettepe Mechanical Engineering	2013
High School	Hayrettin Atmaca High School, Ağrı	2007

WORK EXPERIENCE

Year	Place	Enrollment
2017 -present	FNSS Defense Systems Inc.	Lead Servo Control Design Engineer
2013- 2017	Roketsan Missiles Inc.	Electro-Mechanical Design and Control Engineer
2012-2013	Mercedes Benz Turk Truck Comp.	Intern Eng. Student

FOREIGN LANGUAGES

Advanced English, Fluent German

PUBLICATIONS

1. Yerlikaya, U., Balkan, R. T., "Elektromekanik Kontrol Tahrik Sisteminin Matematiksel Modellenmesi ve Kontrolü," *Türkiye Robotbilim Konferansı (ToRK)*, Istanbul, 2016.

2. Yerlikaya, U., Balkan, R. T., "Hareket Kısıtlı Sistemlerde Viskoz ve Coulomb Sürtünme Tanılaması," *Ulusal Makina Teorisi Sempozyumu (UMTS)*, Trabzon, 2017.

3. Yerlikaya, U., Balkan, R. T., "Elektromekanik Kontrol Tahrik Sistemlerinde Coulomb Sürtünme Telafisi Yöntemiyle Bant Genişliğinin Arttırılması," *Ulusal Makina Teorisi Sempozyumu (UMTS)*, Trabzon, 2017.

4. Yerlikaya, U., Balkan, R. T., "Dynamic Modeling and Control of an Electromechanical Control Actuation System," *ASME Dynamic Systems and Control Conference (DSCC)*, Tysons, Virginia, USA, 2017.

5. Yerlikaya, U., Balkan, R. T., "Identification of Viscous and Coulomb Friction in Motion Constrained Systems," *IEEE/ASME Advanced Intelligent Mechatronics (AIM2018)*, Auckland/New Zealand,2018.

6. Yerlikaya, U., Balkan, R. T., "Increasing the Bandwidth Frequency by Coulomb Friction Compensation Method in Electromechanical Control Actuation Systems," *IEEE/ASME Advanced Intelligent Mechatronics (AIM2019)*, Hong Kong, China, 2019.

The publications given below are the result of doctoral studies.

7. Yerlikaya, U., Balkan, R. T., "Obtaining High-dimensional Configuration Space for Robotic Systems Operating in a Common Environment," *ICACSR 2021: International Conference on Autonomous Control Systems and Robotics*, London, UK, May, 2021.

8. Yerlikaya, U., Balkan, R. T., "Collision Free Motion Planning for Double Turret System Operating in a Common Workspace", Int. J. Control Autom. Syst., 2021.

9. Yerlikaya, U., Balkan, R. T., "Collision Free Motion Planning for an Aligned Multiple-turret System Operating in Extreme Environment", Robotica, 2021.

10. Yerlikaya, U., Balkan, R. T., "End control damping algorithm for a stabilized gun turret system for the satisfaction of the collision avoidance requirement", Robot. Auton. Syst, 2021.