REPUBLIC OF TÜRKİYE
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

DYNAMIC PROGRAMMING BASED ROUTE PLANNING AND MANEUVER CONTROL FOR MOBILE ROBOTS

Barış Yasin DEMİR

MASTER OF SCIENCE THESIS
Department of Control and Automation Engineering
Program of Control and Automation Engineering

Supervisor
Assoc. Prof. Dr. Yavuz EREN

July, 2024

REPUBLIC OF TÜRKİYE
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

DYNAMIC PROGRAMMING BASED ROUTE PLANNING AND MANEUVER CONTROL FOR MOBILE ROBOTS

A thesis submitted by Barış Yasin DEMİR in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE** is approved by the committee on 12.07.2024 in Department of Control and Automation Engineering, Program of Control and Automation Engineering.

Assoc. Prof. Dr. Yavuz EREN
Yildiz Technical University
Supervisor

Approved By the Examining Committee

Assoc. Prof. Dr. Yavuz EREN, Supervisor
Yildiz Technical University _____

Asst. Prof. Dr. Bilal EROL, Member
Yildiz Technical University _____

Assoc. Prof. Dr. Uğur DEMİR, Member
Marmara University _____

I hereby declare that I have obtained the required legal permissions during data collection and exploitation procedures, that I have made the in-text citations and cited the references properly, that I haven't falsified and/or fabricated research data and results of the study and that I have abided by the principles of the scientific research and ethics during my Thesis Study under the title of Dynamic Programming Based Route Planning and Maneuver Control for Mobile Robots supervised by my supervisor, Assoc. Prof. Dr. Yavuz EREN. In the case of a discovery of false statement, I am to acknowledge any legal consequence.

Barış Yasin DEMİR

Signature

Dedicated to my family

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor, Assoc. Prof. Dr. Yavuz EREN, who has always supported me with patience and understanding throughout my thesis study and has been generous in sharing his experiences. His vast knowledge, guidance and constant motivation have played a major role in the completion of this study. It would not have been possible for me to reach this stage without his guidance.

I would like to express my endless gratitude to my professors, friends and colleagues who have always been with me financially and morally and have never withheld their support. Your presence and support gave me strength to overcome the difficulties I encountered in this process.

On the occasion of my master's thesis study, I would like to express my gratitude to my dear, precious and respectful mother Şaziye DEMİR and my father Rasim DEMİR, who provided all kinds of support throughout my education and training life. Their sacrifices, patience and love have been my greatest sources of motivation in reaching this point. My parents encouraged me by making me feel that they were always by my side and gave me the courage I needed to reach my goals.

In addition, my dear sisters, your presence, warmth and support have always been a source of morale for me. I would like to extend my sincerest love to you too. I am grateful for being by my side and supporting me throughout my life.

Barış Yasin DEMİR

TABLE OF CONTENTS

LIST OF SYMBOLS vii

LIST OF ABBREVIATIONS viii

LIST OF FIGURES ix

LIST OF TABLES x

ABSTRACT xi

ÖZET xiii

1 INTRODUCTION 1

1.1 Literature Review 1

1.2 Hypothesis 4

1.3 Purpose of the Thesis 4

1.4 Methodology 5

2 PROBLEM DESCRIPTION AND MODELLING OF MOBILE ROBOT 7

2.1 Problem Description 7

2.2 Mathematical Modelling of Mobile Robots 9

2.2.1 Kinematic Model of Mobile Robots 9

3 CONFIGURATION OF PATH PLANNING SCENARIOS 11

3.1 Configuration of Scenario-I 11

3.2 Configuration of Scenario-II 14

3.3 Configuration of Scenario-III 16

3.4 Configuration of Scenario-IV 18

3.5 Configuration of Scenario-V 24

4 SIMULATION STUDIES 25

4.1 Simulation Studies for Scenario-I 26

4.2 Simulation Studies for Scenario-II 28

4.3 Simulation Studies for Scenario-III 29

4.4 Simulation Studies for Scenario-IV 32
4.5 Simulation Studies for Scenario-V 34

5 DISCUSSIONS **41**
5.1 Discussions on Research Outcomes 41
5.2 Future Study Suggestions 41

6 CONCLUSION **43**

REFERENCES **44**

PUBLICATIONS FROM THE THESIS **47**

LIST OF SYMBOLS

$a(t)$	Acceleration
ω	Angular Velocity
K_d	Derivative Gain
d	Distance Between of The Wheels
K_i	Integral Gain
v	Linear Velocity
$\|d\|$	Norm of The Vector
u	Orientation
$\%$OS	Percent Overshoot
K_p	Proportional Gain
r	Radius of Mobile Robot
θ	Rotational Angle
T_s	Settling Time
$\sum$	Summation
t	Time
τ	Time Constant
Δt	Time Step
x	X-Coordinate
y	Y-Coordinate

LIST OF ABBREVIATIONS

APF	Artificial Potential Field
MDP	Markov Decision Processes
MPC	Model Predictive Control
PID	Proportional-Integral-Derivative
RMS	Root Mean Square
RRT	Random Rapidly Tree
VFH	Vector Field Histogram

LIST OF FIGURES

Figure 1.1 Flowchart of the methodology 5
Figure 4.1 Block diagram for system 26
Figure 4.2 Simulation of Scenario-I 26
Figure 4.3 PID response for Scenario-I 27
Figure 4.4 Simulation of Scenario-II 28
Figure 4.5 PID response for Scenario-II 29
Figure 4.6 Simulation of Scenario-III 30
Figure 4.7 PID response for Scenario-III 31
Figure 4.8 Images for Scenario-IV - 1 to 9 32
Figure 4.9 Images for Scenario-IV - 10 to 18 33
Figure 4.10 Images for Scenario-IV - 19 and 20 33
Figure 4.11 PID response for Scenario-IV 34
Figure 4.12 Images for Scenario-V.a - 1 to 9 35
Figure 4.13 Images for Scenario-V.a - 10 to 18 36
Figure 4.14 Images for Scenario-V.b - 1 to 4 37
Figure 4.15 Images for Scenario-V.b - 5 and 6 38
Figure 4.16 PID response for Scenario-V.a 39
Figure 4.17 PID response for Scenario-V.b 39

LIST OF TABLES

Table 4.1 Common parameters for simulations 25
Table 4.2 PID parameters for each scenario 25
Table 4.3 Important parameters for Scenario-III 30
Table 4.4 Parameters for Scenario-V.a 35
Table 4.5 Parameters for Scenario-V.b 37

ABSTRACT

Dynamic Programming Based Route Planning and Maneuver Control for Mobile Robots

Barış Yasin DEMİR

Department of Control and Automation Engineering
Master of Science Thesis

Supervisor: Assoc. Prof. Dr. Yavuz EREN

In this thesis, the performance of the Dijkstra algorithm, one of the optimal path finding algorithms for mobile robots, has been investigated through five simulation scenarios of increasing complexity. Initially, scenarios with static obstacles were introduced, followed by scenarios incorporating dynamic obstacles alongside the static ones. Additionally, challenging scenarios requiring dynamic path adjustments were defined, including cases with one or multiple vehicles travelling in the same environment and leader-follower configurations. The Dijkstra algorithm was restructured with necessary improvements for obstacle avoidance. The effectiveness of the Dijkstra algorithm was evaluated in all scenarios. The results provide significant insights into the performance of the Dijkstra algorithm in the context of autonomous navigation.

In real-time navigation-based applications for mobile robots, controlling the rotational dynamics based on vehicle dynamics is among the important control problems. In this context, the implementation of necessary maneuver movements for the route obtained based on the Dijkstra algorithm with a simply applicable controller is also addressed in this thesis. A Proportional-Integral-Derivative (PID) controller was designed for reference tracking based on the rotational angle of the mobile vehicle. In scenarios involving multiple vehicles, at each change of the rotational angle dependent on the route, the data related to the primary vehicle's rotational angle were also used for the other vehicle. Holistically, the control performance of the mobile robot's reference rotational angles under different map forms and obstacle configurations was tested in a simulation environment.

The obtained results were interpreted for each scenario, and the insights gained regarding the development of autonomous vehicle navigation strategies were expressed.

Keywords: Mobile robots, dijkstra algorithm, route planning, PID controller, reference tracking

ÖZET

Mobil Robotlar için Dinamik Programlama Temelli Rota Planlaması ve Manevra Kontrolü

Barış Yasin DEMİR

Kontrol ve Otomasyon Mühendisliği Anabilim Dalı
Yüksek Lisans Tezi

Danışman: Doç. Dr. Yavuz EREN

Bu tez çalışmasında mobil robotlar için optimum yol bulma algoritmalarından olan Dijkstra algoritmasının performansı, her biri artan karmaşıklığa sahip beş simülasyon senaryosu aracılığıyla araştırılmıştır. Başlangıçta statik engellerin olduğu senaryolar daha sonra da statik engellerin yanına hareketli engellerin olduğu senaryolar tanıtılmıştır. Ayrıca, aynı ortamda seyahat eden bir veya birden fazla araç durumu ve lider-takipçi konfigürasyonları ile dinamik yol ayarlamaları gerektiren zorlayıcı senaryolar tanımlanmıştır. Engellerden kaçınma için gerekli iyileştirmeler ile Dijkstra algoritması yeniden düzenlenmiştir. Tüm senaryolarda Dijkstra algoritmasının etkinliği değerlendirilmiştir. Sonuçlar, otonom navigasyon bağlamında Dijkstra algoritmasının performansı hakkında önemli bilgiler sağlamaktadır.

Mobil robotlar için navigasyon tabanlı gerçek zamanlı uygulamalarda, araç dinamiklerine bağlı olarak dönüş dinamiklerinin kontrol edilmesi önemli kontrol problemleri arasında yer almaktadır. Bu bağlamda Dijkstra algoritmasına bağlı olarak elde edilen rota için gerekli manevra hareketlerinin basitçe uygulanabilen bir kontrolcü ile gerçeklenmesi de tez kapsamında ele alınmıştır. Mobil aracın dönüş açısına dayalı referans takibi için Oransal-İntegral-Türevsel (PID) kontrolcüsü tasarlanmıştır. Birden fazla aracın kullanıldığı senaryolarda, rotaya bağlı her dönüş açısı değişiminde, birincil aracın dönüş açısına ilişkin veriler diğer araç için de kullanılmıştır. Bütüncül olarak bakıldığında, elde edilen rota uyarınca mobil robotun referans dönüş açılarının farklı harita şekilleri ve engel konfigürasyonları altındaki kontrol performansı benzetim ortamında test edilmiştir.

Elde edilen sonuçlar her bir senaryo için yorumlanmıştır ve otonom araç navigasyon stratejilerinin gelişimine yönelik elde edilen fikirler ifade edilmiştir.

Anahtar Kelimeler: Mobil robotlar, dijkstra algoritması, rota planlama, PID kontrolcü, referans izleme

1
INTRODUCTION

1.1 Literature Review

Autonomous mobile robots are intelligent machines that fulfil specific tasks, designed to meet the needs of humanity. Similar to assistants, they can complete tasks according to our instructions. These instructions are given through codes. Codes are a set of commands that tell the robot what to do and how to do it, just as a recipe tells us how to cook a dish. These codes are written using computer programmes and uploaded to the robot. Thanks to the programming tools , robots can easily perform even complex tasks. For example, we can use a robot to clean a house, manufacture in a factory or perform surgery in a hospital [1]. Mobile robots play an important role by increasing productivity in a wide range of fields from manufacturing to agriculture, military to education [2]. In recent years, the COVID-19 pandemic has caused the use of autonomous mobile robots to increase rapidly. With the pandemic, human contact is tried to be minimised to prevent the spread of the virus. This has accelerated the transition from human to machine in many sectors such as healthcare, security and food. Mobile robots have also played an important role in this transition, replacing human labour in many areas [3]. Autonomous mobile robots need to be able to adapt to the environment, communicate with other systems and be resilient to unexpected situations. They must be able to perceive and interpret their environment in real time to avoid obstacles and complete their tasks in complex environments. They must also be able to communicate reliably and without delay with control stations and other robots, and be fault tolerant. Overcoming these challenges will enable autonomous robots to provide significant benefits in manufacturing, logistics, search and rescue and many other areas [4]. Mobile robots have the ability to navigate autonomously in different environments, such as static and dynamic, regular and uncertain. This ability is made possible by the navigation technique used to move the robot from one point to another. The basic principle of navigation is to enable the robot to perceive its environment, determine its position and reach its destination by

avoiding obstacles [5]. A navigation system helps you get from where you are to where you want to be. It figures out the best route and adjusts it along the way if it bumps into any surprises, like a road closure [6]. Robots require a variety of navigation techniques to navigate safely and efficiently in complex environments. These techniques perform critical tasks such as determining the robot's position, detecting obstacles, and planning the optimal path to the target [7]. Path planning is the process of finding the shortest, safest or fastest path from the robot's current position to the destination. During this process, it is necessary to consider the obstacles around the robot. In mobile robotics, path planning is a critical task that enables the robot to reach the goal in the most appropriate way by avoiding obstacles. This planning directly affects the robot's performance, safety and ease of use [8]. Effective path planning strategies can significantly improve the performance and efficiency of mobile robots. Choosing the right navigation technique reduces the robot's overall operating costs by saving time, minimizing wear and tear and optimising capital expenditure [9]. In the field of mobile robots, figuring out how to get from point A to point B is a crucial task. This challenge, known as path planning, involves finding a safe path for the robot (considering its size and shape) within its environment. Unlike a simple dot, the robot needs to not only determine the right direction but also make adjustments along the way to reach its destination. This additional step of figuring out how to move while considering the robot's physical form is called maneuvering planning [10].

Various path planning methods were developed for mobile robots that Dijkstra, A*, D*, Artificial Potential Field (APF), Vector Field Histogram (VFH), Random Rapidly Tree (RRT) algorithms and etc [11]. Dijkstra algorithm is an algorithm used to select the optimal path in a topology roadmap that meets certain conditions. Its basic principle is to find the shortest path by starting from the source point (starting point), moving to the neighbouring point with the shortest distance at each step and applying this process to all points. It is simple and easy to understand, fast and efficient. It always finds the shortest path in graphs without negative edge weights [12]. The A* algorithm is an algorithm used in grid-based path planning which, unlike other graph-based search algorithms, uses heuristic functions to estimate the distance to the destination. In this way, it works faster and more efficiently by evaluating fewer nodes to find the optimal path. The A* algorithm calculates f(n) = g(n) + h(n) for each node. g(n) represents the actual distance from the source node to node n, h(n) represents the estimated distance from node n to the destination node, and f(n) represents the estimated total distance from node n to the destination. The algorithm proceeds by selecting the node with the lowest f(n) value at each step [13]. D* is a graph algorithm known for its speed in finding new

paths when the surroundings shift [14]. This method can be described as a more dynamic and expanded variant of the A* algorithm that operates without relying on a heuristic function [15]. The D* algorithm is composed of three main functions: Process-State, Modify-Cost, and Move-Robot. Process-State calculates the best path costs toward the target, Modify-Cost updates the cost function of connections, and Move-Robot utilizes these functions to guide the robot optimally during movement [16]. Another method which called RRT is a popular algorithm used for path planning of robots and other autonomous systems in complex environments. This method uses a random tree search strategy to find the optimal route between the origin and destination points while avoiding obstacles [17]. In recent years, the RRT algorithm, based on random search principles, has become a popular method for path planning in complex and high-dimensional spaces for robotic manipulators. This algorithm is characterised by its probability-based structure and excellent scalability. In addition, thanks to the use of random sampling and collision detection in the state space, it can effectively plan paths even in environments with complex constraints, without the need for modelling the entire space [18]. The working principle of the RRT algorithm is as follows. The start and target configurations are determined. A tree is initialised with a single node consisting of the initial configuration. A random sample is selected from the configuration space. For the selected random sample, the nearest node in the tree is found. It is checked whether the link between the random instance and the nearest node is collision-free. If the connection is collision-free, the tree is expanded with a new node and this node is connected to the random instance. If the connection is collisional, a new node is created using the routing function and this node is connected to the nearest node [19]. Another effective technique is potential field-based techniques for path planning. Previous studies have shown that potential field-based planning methods are suitable for quickly and efficiently planning routes around stationary obstacles [20]. The APF method is also used for obstacle avoidance on behalf of mobile robots and has a great importance in the literature in this respect. The concept of using APF for robot obstacle avoidance was pioneered by Khatib [21]. This approach models the motion of the robot as a particle constrained by an APF. The APF consists of an attractive field that enables the robot to reach the target and repulsive fields that enable it to avoid obstacles [22]. Obstacle avoidance is critical for successful applications of mobile robot systems. By using different sensors and algorithms, mobile robots are enabled to detect obstacles in their environment and travel safely. With the advances in technology, obstacle avoidance systems will become more advanced and the usage areas of mobile robots will expand [23]. Just as APF, VFH method is also important for obstacle avoidance in autonomous navigation of the mobile robot [24]. The basic VFH algorithm is a method that

helps robots avoid obstacles, but it is not completely goal-oriented. This means that it does not guarantee that the robot always reaches the target [25].

For mobile robots safety, controlling techniques are very important. Research into controlling the mobile robots has been a focal point for many studies. Specifically, the unique movement constraints of these systems have spurred the creation of complex control methods [26]. For example, tracking control is a type of control challenge encountered with non-holonomic systems. Depending on whether the system is represented by a kinematic model or a dynamic model, the tracking control problem is categorized as either a kinematic or dynamic tracking control problem [27]. The one of the most known as a controller for mobile robots is Proportional-Integral-Derivative (PID) controller. To ensure that a mobile robot is asymptotically stable, it is essential to use PID controller [28]. Thanks to the PID controller, the features of the mobile robot such as position and rotation angle can be controlled [29]. The PID algorithm comprises three tuning parameters that have specific roles over time: P (proportional) calculates current errors, I (integral) tracks cumulative past errors, and D (derivative) estimates future errors based on the current rate of change [30].

1.2 Hypothesis

This thesis focuses on the development and evaluation of the Dijkstra algorithm, one of the optimal path finding algorithms for mobile robots. The hypothesis is to test the performance limits of Dijkstra algorithm with PID, the most easily realisable controller, under different map topologies and obstacle configurations. According to this hypothesis, it is aimed that the optimal path finding algorithm developed in this thesis will enable mobile robots to reach their targets safely, quickly and efficiently and that the designed controller will be controllable for the maneuvering of mobile robots in terms of safety in such a real-life application.

1.3 Purpose of the Thesis

The purpose of this thesis is to simulate various scenarios to enable mobile robots to move safely and efficiently in complex environments. The focus of the study is to enable mobile robots to find the shortest and safest paths between specified start and target points, taking into account static and dynamic obstacles. In Scenario-I, the shortest journey of a single mobile robot to a given destination considering static obstacles is simulated. In Scenario-II, in addition to the first scenario, there is a second mobile robot travelling towards the same destination. In order for the two

vehicles to follow the safest path determined by Dijkstra algorithm and to minimise the risk of mutual collision, the rotational angle of the first vehicle needs to be dynamically changed. Considering Scenario-III, the second mobile robot follows the same trajectory as the first mobile robot. Moreover, different methods are applied to ensure that both vehicles reach the target considering the static obstacles. The second vehicle follows the trajectory of the first vehicle while the first vehicle travels to the target according to Dijkstra algorithm. As in the second scenario, the rotational angle of the first vehicle needs to be changed dynamically to minimise the risk of mutual collision. At the other scenarios (variations of Scenario-IV) both static and dynamic obstacles were considered. Mobile robots were simulated to avoid obstacles using Dijkstra algorithm to select the safest and shortest paths to the specified target. Furthermore, to increase the complexity of the scenarios, dynamic control of the rotational angles was provided to reduce the risk of collisions between the vehicles, both between themselves and with dynamic obstacles.

As a result of the simulations performed for each scenario, the rotational angle control of the primary mobile robot at each step is monitored and managed in real time with the PID algorithm. The results of this study aim to contribute to the safe and effective use of mobile robots in real world applications. The findings can provide important information for the development of algorithms and control strategies used in mobile robot navigation.

1.4 Methodology

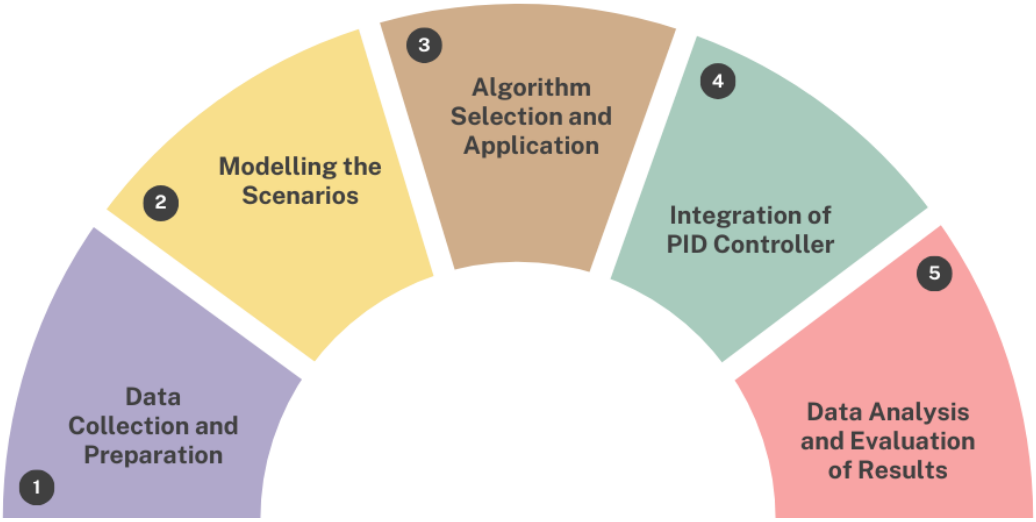


Figure 1.1 Flowchart of the methodology

The methodology of this work is based on several scenarios developed to simulate the safe navigation of mobile robots in complex environments. All simulations were performed in MATLAB® environment. In the following, the main steps and the methodological approach of the thesis are described in detail.

In the step of which named Data Collection and Preparation, the necessary data were collected to determine the mobile robots and obstacle typologies to be used in the study. The physical characteristics of the mobile robot and the characteristics of the static and dynamic obstacles in the environment are defined at this stage.

In the step of which named Modelling the Scenarios, firstly simulation models were designed for each scenario. The coordinates of static and dynamic obstacles were determined and represented on the map. The kinematic models and motion capabilities of the mobile robots has been integrated in the simulation environment to suit the requirements of the scenarios.

In the next step which is Algorithm Selection and Application, the most suitable navigation algorithms were determined for each mobile robot according to the requirements. Dijkstra algorithm is preferred to ensure that mobile robots reach certain targets in the shortest and safest ways.

Then, in the step of Integration of PID Controller, the rotation angles of the mobile robots at each step were obtained through a feedback system and given as input to the PID controller. In this way, it was tried to determine whether the mobile robots could follow the specified trajectories and adapt to the environmental conditions.

Finally, in the step of Data Analysis and Evaluation of Results, the simulation results of the mobile robots were recorded and analysed. Observations on the path of the mobile robots on the map, rotational movements and environmental conditions were evaluated within the scope of the methodology. Statistical analyses were performed on the simulation results to evaluate the performance of the mobile robots and the effectiveness of the navigation strategies. A detailed evaluation of the differences between the scenarios and the effects of algorithm choices is carried out.

These methodological steps explain the general structure of the thesis and the phases of the study. The results obtained aim to provide important information for the development of safe and effective navigation strategies for mobile robots.

2
PROBLEM DESCRIPTION AND MODELLING OF MOBILE ROBOT

2.1 Problem Description

The main problem addressed in our study is to enable mobile robots to safely reach specific destinations in various scenarios. While reaching this target, robots need to consider obstacles and interact with other robots to avoid collisions. In addition, the emergence of mobile obstacles requires robots to adapt to dynamic environmental conditions and make instantaneous decisions.

To address the mentioned problem, methods such as Dijkstra algorithm and PID controller are used to calculate the optimal paths of mobile robots and track these paths in real time. This algorithm helps the mobile robots to find the shortest and safest path through obstacles, while the PID controller is used for the mobile robots to follow the target path by considering the rotational angles values. The rotational angle information obtained in our study was used as basic data to evaluate the usability of the PID controller for reference tracking.

In this study, the total of 5 scenarios were considered. These scenarios show how mobile robots behave in different environmental conditions and which strategies are most effective, as well as simulate situations where mobile robots are alone or interacting with other robots.

In this manner, Scenario-I focuses on the target reaching problem of a single mobile robot in an environment full of static obstacles. The robot calculates the shortest path using the Dijkstra algorithm and aims to reach the target safely by following this path. The Dijkstra algorithm determines the optimal path that prevents the robot from colliding with obstacles.

Scenario-II increases the complexity of the first scenario by adding two mobile robots. Both robots have different starting positions but the target point is the same.

In this scenario, both robots calculate the shortest path using the Dijkstra algorithm and try to avoid both static obstacles and each other while following this route.

In the Scenario-III, the leader-follower approach is discussed. While the main vehicle, that is, the leader vehicle, takes the shortest path using the Dijkstra algorithm, the follower vehicle follows the trajectory of the leader vehicle. Similar safety precautions have been taken into account in this scenario as in the Scenario-II.

Scenario-IV considers the target reaching problem of a single mobile robot in an environment containing dynamic obstacles in addition to static obstacles. The robot calculates the shortest path using Dijkstra algorithm and avoids static and dynamic obstacles while following this path. In case of encountering dynamic obstacles, the robot performs emergency maneuvres to avoid collision and continues to complete the path. This scenario requires the robot to sense its environment and make quick decisions to avoid collisions.

Scenario-V has two simulations. In these simulations, the problem of two mobile robots reaching the target in an environment full of dynamic obstacles is addressed. Both robots calculate the shortest path using the Dijkstra algorithm and avoid static obstacles, each other and dynamic obstacles while following this path. In the first of the simulations, a simulation was made with collision control between vehicles and dynamic obstacles, while in the second, a simulation was made with collision control between vehicles.

At the end of each scenario, the rotational angle of the main mobile robot at each step until it reaches the target is tried to be controlled by means of a PID controller. The kinematic equations of the common mobile robot used in the scenarios are given in 2.2.1.

As a result, this study investigates navigation strategies for mobile robots across a series of increasingly complex scenarios. By analyzing these scenarios, the research aims to contribute to the development of safe, efficient, and adaptive navigation strategies for mobile robots. This systematic approach, moving from static environments to dynamic ones and from single robots to multiple robots, provides valuable insights for real-world applications. The PID controller implementation further strengthens the practical applicability of the research findings. Ultimately, this thesis paves the way for mobile robots to navigate complex environments with greater safety and efficiency. The developed strategies have the potential to be implemented in a wide range of fields, from industrial automation to autonomous vehicles.

2.2 Mathematical Modelling of Mobile Robots

The modeling of mobile robots involves rigorous mathematical representations that describe their behavior, dynamics, and interaction with the environment. Accurate models are crucial for designing control strategies, simulating performance, and ensuring robust operation in real-world scenarios. Considering the various modeling approaches, kinematic and dynamic modeling types are of primary importance. Kinematic models describe the motion of robots without considering the forces that cause the motion. These models are based on geometric relationships and are particularly useful for path planning and trajectory generation. Kinematic modeling typically involves defining the robot's configuration, such as position and orientation, and how it changes over time. Besides, dynamic models incorporate the forces and torques acting on the robot, providing a more comprehensive representation of its motion. These models are essential for designing control systems that account for the robot's inertia, friction, and other physical properties. Dynamic modeling requires understanding the robot's mass distribution, actuator capabilities, and interaction with the environment. Hence, in this study, kinematic modeling techniques have been utilized as we focus on solving the path planning problem of mobile robots.

2.2.1 Kinematic Model of Mobile Robots

In this study, two-wheeled mobile robots with one free wheel for balance and two drive wheels on the same axis are investigated. This type of vehicles are widely used in various applications requiring precise movement and control. Due to their maneuverability, they are often employed in studies and applications involving path planning, obstacle avoidance, and mapping.

The variables used in kinematic modelling for differentiating two-wheeled mobile robots are positional and angle parameterization (x, y, θ), radius of mobile robot (r), distance between of the wheels (d), linear velocity (v) and angular velocity (ω) [31].

The equations with these parameters are given below [32].

$$\omega = \dot{\theta} = \frac{v}{r} \tag{2.1}$$

This equation (2.1) represents the angular velocity, which depends on the linear velocity and the radius of the mobile robot.

$$v_{left} = (r + \frac{d}{2}) \cdot \omega \tag{2.2}$$

$$v_{right} = (r - \frac{d}{2}) \cdot \omega \tag{2.3}$$

$$V = \frac{v_{left} + v_{right}}{2} \tag{2.4}$$

$$\omega = \frac{v_{left} - v_{right}}{d} \tag{2.5}$$

Based on equations 2.2, 2.3 and 2.4, a new equation for angular velocity is obtained as in equation 2.5.

$$\dot{x} = v \cdot cos\theta \tag{2.6}$$

$$\dot{y} = v \cdot sin\theta \tag{2.7}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{2.8}$$

Based on equations 2.1, 2.6 and 2.7, the state-space form in equation 2.8 is obtained.

3
CONFIGURATION OF PATH PLANNING SCENARIOS

In the previous section, definitions of the scenarios and the mobile robot modelling were presented. In this section, the fundamental mathematical configurations that will enable the execution of the scenarios in the simulation environment will be provided. Basic calculations and algorithmic solutions required for the necessary actions according to the scenario definitions have been described. Algorithms related to the requirements such as obstacle placements needed in the scenarios, mobile robot position calculations, restructured Dijkstra algorithms, inter-vehicle collision control, and vehicle-obstacle distance calculations have been explained. All processes necessary for the detailed execution of the scenarios in the simulation environment have been step-by-step clarified.

3.1 Configuration of Scenario-I

Scenario-I addresses the target-reaching problem of a single mobile robot navigating an environment filled with static obstacles. Suppose that our map is of size M×N, i.e. it contains a total of M rows and N columns. The position of an obstacle is expressed in (x,y) coordinates, where x is the column number and y is the row number. Since the obstacles will be placed randomly on the map, the values of x and y will be chosen randomly. For this situation, the operations performed in this scenario are described by the pseudo code which called Algorithm 1.

Algorithm 1 Placing the Obstacles

Step 1. Inputs: *M, N, K*

Step 2. Function `placeObstacles(`*M, N, K*`)`**:**

```
Initialize empty list obstacles;
for i = 1 to K do
    Generate a random integer x between 1 and N − 1;
    Generate a random integer y between 2 and M;
    Append tuple (x, y) to obstacles;
end
return obstacles;
```

The value of x is chosen as a random integer between 1 and N-1. This is a column number such that it is within the boundaries of the map. The value y is chosen as a random integer between 2 and M. This is used to select a location at the top of the map, outside the first row. The total number of obstacles, determined as K, refers to the number of obstacles to be placed on the map. Total number of K obstacles are generated as

$$\sum_{i=1}^{K} (x_i, y_i) \tag{3.1}$$

where i represents the index number of obstacle.

In this configuration, x_i and y_i values are selected randomly for each obstacle separately. The value of x_i is chosen from a uniform distribution between 1 and N-1 and the value of y_i is chosen from a uniform distribution between 2 and M. In this way, the obstacles are randomly placed in the map.

Basic motion equations and trigonometric properties have been utilized to calculate the movement dynamics of the mobile robot as it progresses towards the target point. The dynamic calculations carried out within the scope of Scenario-I are provided in Algorithm 2.

Algorithm 2 Calculating the New Position of Mobile Robot

Step 1. Inputs: $x_{\text{c}}, y_{\text{c}}, x_{\text{f}}, y_{\text{f}}, v, \Delta t$

Step 2. Function `calculateNewPosition(`$x_c, y_c, x_f, y_f, v, \Delta t$`)`**:**

$\theta := tan^{-1}(y_{\text{f}} - y_{\text{c}}, x_{\text{f}} - x_{\text{c}})$

$x_{\text{new}} := x_{\text{c}} + v \cdot \cos(\theta) \cdot \Delta t$

$y_{\text{new}} := y_{\text{c}} + v \cdot \sin(\theta) \cdot \Delta t$

return (x_{new}, y_{new})

Trigonometry is utilized to calculate the direction (i.e. angle) from the robot's current position to the target point. This is done by the function tan^{-1}. Current location of mobile robot is (x_c, y_c) and location of target is (x_f, y_f). The new

position of the robot is calculated with constant velocity determined with respect to the current position and the direction towards the target. This is realised using the Euler method or a similar simple differential equations approach. Where v is constant velocity of the mobile robot, θ is heading angle towards the target (in radians) and Δt is time step (time elapsed during the simulation step).

In this study, the Dijkstra algorithm, which finds the shortest distance between the start and target nodes in a unweighted graph model, has been utilized. The algorithm determining the reference route for the performance of Scenario-I, based on the Dijkstra approach, has been developed as Algorithm 3.

Algorithm 3 Dijkstra Algorithm for the Scenario-I

Step 1. Input: G (graph), s (starting node)
Step 2. Function `Dijkstra(`*G, s*`)`**:**
    $d[v] := \infty$ for $v \in G$.vertices
    $d[s] := 0$
    $visited := \emptyset$
    while |*visited*| $<$ |*G.vertices*| **do**
        $u := \min(\{v \in G.\text{vertices} : v \notin \text{visited}\})$
        visited := visited $\cup \{u\}$
        for $v \in G.neighbors(u)$ **do**
            $d[v] := \min(d[v], d[u] + 1)$
        end
    end
    return d
Function `getNeighbors(`$node, mapSize$`)`**:**
    neighbors = [ ]
    if $node(1) > 1$ **then**
        neighbors.append([node(1)-1, node(2)])
    end
    if $node(1) < mapSize(1)$ **then**
        neighbors.append([node(1)+1, node(2)])
    end
    if $node(2) > 1$ **then**
        neighbors.append([node(1), node(2)-1])
    end
    if $node(2) < mapSize(2)$ **then**
        neighbors.append([node(1), node(2)+1])
    end
    return $neighbors$

In Algorithm 3, G=(V,E) denotes unweighted graph, where V represents set of vertices and E represent set of edges. s is the initial node. d(v) is the shortest path length from the start node to node v. visited[v] is boolean value indicating whether node v has been visited. The process of the algorithm is as follows step by step. Initialize d(s)=0 for the starting node and d(v)=∞ infinite for all other nodes.

To cover all nodes until the visited set is empty, select the node u that is not included in the visited set and has the smallest d(v) value, mark u as visited, update d(v) for all nodes neighbouring to u:d(v)=min(d(v),d(u)+weight(u,v)). Weight(u,v) is the weight of the edge between u and v. It has been assumed 1 in Algorithm 3. The algorithm traverses the nodes such that the value of d(v) is the smallest at each step. The shortest path distances from the start to each node are found. At the end of the algorithm, the values of d(v) gives the shortest paths.

The 'GetNeighbors' function referenced in Algorithm 3 is utilized to determine the neighbours of a node.In the first step of the function, inputs are taken for a particular node and the dimensions of the map. In the second step, an empty list (neighbors) is created to hold neighbor nodes. The function checks the four main neighbors of the node (left, right, top, bottom). If the node's x-coordinate is greater than 1, the node to its left is added to the neighbors list. If the node's x coordinate is smaller than the map width, the node to its right is added to the neighbors list. If the node's y coordinate is greater than 1, the node above it is added to the neighbors list. If the node's y coordinate is less than the map height, the node below it is added to the neighbors list. The function ends by returning the list containing the specified neighbor nodes (neighbours).

This function is used to determine the valid neighbors of a given node and ensures that the map does not exceed its boundaries. Neighboring nodes are returned for use in later stages of the algorithm.

3.2 Configuration of Scenario-II

Scenario-II involves a more complex version of the configuration in Scenario-I. In this scenario, there are two mobile vehicles. Similar to the previous scenario, it includes a map with static obstacles, where the vehicles start from different initial conditions and attempt to reach the same point.

The obstacle assignment process in this scenario is similar to the procedure in Scenario-I. This process is logically given in the pseudo code called Algorithm 1. The next step for this scenario is simulating the movement of each mobile robots. The algorithm of this step is as Algorithm 4. The starting positions of the vehicles are defined as (x, y) coordinates with the variables 'vehiclePosition1' and 'vehiclePosition2' which are given in Algorithm 4. The movement of each vehicle is calculated using constant velocity (v) and direction (θ). The new position is calculated by the following equations for the x and y coordinates.

$$x_{new} = x_{old} + \ vcos(\theta)dt \tag{3.2}$$

$$y_{new} = y_{old} + vsin(\theta)dt \tag{3.3}$$

where dt represents the time interval (in seconds), θ is the angle in radians, and 'new' and 'old' subscripts refer current and previous values of the mentioned variable, respectively.

Algorithm 4 Calculating New Position of Two Mobile Robots

Step 1. Input: vehiclePos1, vehiclePos2, targetPos, v, dt, map

Step 2. Function `calculatePosition(`*vehiclePos1, vehiclePos2, targetPos, v, dt, map*`)`**:**

- **while** *shortestPath is not empty* **do**
  - $\theta_1 := \tan^{-1}$(targetPos - vehiclePos1)
  - $\theta_2 := \tan^{-1}$ (targetPos - vehiclePos2)
  - $(x_{\text{new1}}, y_{\text{new1}})$:= calculateNewPos(vehiclePos1, v, θ_1, dt)
  - $(x_{\text{new2}}, y_{\text{new2}})$:= calculateNewPos(vehiclePos2, v, θ_2, dt)
  - vehiclePos1 := $(x_{\text{new1}}, y_{\text{new1}})$
  - vehiclePos2 := $(x_{\text{new2}}, y_{\text{new2}})$
  - **if** *distance(vehiclePos1, vehiclePos2)* $\leq$ *robotRadius* **then**
    - Collision Control for Vehicles `// Algorithm 5 is running`
  - **end**
- **end**
- **return**

Step 3. Function `calculateNewPos(`*vehiclePos, v, θ, dt*`)`**:**

- $x_{\text{new}} := x_{\text{old}} + v \cdot \cos(\theta) \cdot dt$
- $y_{\text{new}} := y_{\text{old}} + v \cdot \sin(\theta) \cdot dt$
- **return** $(x_{\text{new}}, y_{\text{new}})$

Operational steps of the simulated moving dynamics of the vehicles are given step by step in Algorithm 4. Hence, the main loop runs until it reaches each point in the 'shortestPath' array. At each step, the direction to the target is determined for both vehicles. The new position is calculated using velocity v and direction. In this manner, collision check is performed. If the distance between the vehicles is smaller than 'robotRadius', the direction of the first vehicle is changed. And then, new positions are updated.

While simulating the movement of vehicles, it is also important to check for collisions in order to plan a safe path. Hence, Algorithm 5 is utilized to prevent the collision happened between the vehicles. This issue is carried out by calculating the distance function between the corresponding coordinates of the vehicles which is given as follows.

$$distance = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{3.4}$$

Algorithm 5 Collision Control for Vehicles

Step 1. Input: vehiclePosition1, vehiclePosition2, robotRadius
Step 2. Function `CollisionControl`(*vehiclePosition1, vehiclePosition2, robotRadius*) **:**
  if *calculateDistance (vehiclePosition1, vehiclePosition2)* $\leq$ *robotRadius* **then**
    $\theta_1 := \theta_1 + 30$
  end
Step 3. Function `calculateDistance`(*vehiclePosition1, vehiclePosition2*) **:**
  return $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

where x_1, y_1, x_2 and y_2 represent the x and y coordinates of each vehicle respectively. If the distance between vehicles is smaller than 'robotRadius', a collision is considered to occur. Thus, the rotational angle of first vehicle changes by 30°.

Finally, considering Scenario-II, Djikstra algorithm was used as the shortest path algorithm for both vehicles. Algorithm 3 was benefited for this.

3.3 Configuration of Scenario-III

Scenario-III deals with the problem which a single mobile robot tries to reach the destination point in an environment containing both of the static and dynamic obstacles. The obstacle assignment process in this scenario is the same as in Algorithm 1. In this scenario, once the map and obstacles have been set up, the next step is to determine the location of the vehicles and the target. Then, the regular procedure is again processed which is steering the vehicles to move from their starting position to the target point. The mathematical operations based on the movement of the vehicles are described in Algorithm 6.

Algorithm 6 Calculating the New Position of Leader and Follower Vehicles

Step 1. Input: p, targetPoint, v, dt
$d_x :=$ targetPoint.x $- p.x$
$d_y :=$ targetPoint.y $- p.y$
norm_d $:= \sqrt{d_x^2 + d_y^2}$
u_x $:= \frac{d_x}{\text{norm_d}}$
u_y $:= \frac{d_y}{\text{norm_d}}$
$p_new_x := p.x + v \cdot$ u_x $\cdot dt$
$p_new_y := p.y + v \cdot$ u_y $\cdot dt$
return (p_new_x, p_new_y)

To represent the positions of two mobile robots, $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ are defined. Here (x_1, y_1) and (x_2, y_2) denote the positions of the first and second vehicle respectively. Let's define the target point as $p_{target} = (x_{target}, y_{target})$ for each vehicle. Guidance between two vehicles is usually performed by calculating the direction vectors towards the target. The distance between the two vehicles and the orientation towards the target are used in these calculations. The distance vector between two vehicles is defined as d=p1-p2. The orientation vector is calculated as u=$\|d\|$. Here $\|d\|$ is defined as the norm of the vector d.

The interaction between vehicles is modelled using the distance and orientation vectors between two vehicles. For example, steering changes can be made when one vehicle approaches or is closer than a certain distance to another. The distance and direction between vehicles can be calculated using the distance vector d and the orientation vector u. For example, if the vehicles are less than a certain distance (collision risk radius), one vehicle can change its direction towards the other. Within the scope of this scenario, collision control was performed benefiting Algorithm 5.

Vector and matrix operations can be used to mathematically express the interaction between two vehicles. For example, the new positions of the vehicles for through the destination point can be updated as follows.

$$p_1(t + \Delta t) = p_1(t) + vu\Delta t \tag{3.5}$$

$$p_2(t + \Delta t) = p_2(t) + vu\,'\Delta t \tag{3.6}$$

where v represents the constant velocity, and u' represent the directions of movement of the first and second vehicle, respectively. Δt is the time step.

The position of each vehicle is defined as $p(t) = (x(t), y(t))$ where $x(t)$ and $y(t)$ are the x and y coordinates of the vehicle at time t, respectively. Its velocity is defined as $v(t) = (vx(t), vy(t))$ $vx(t)$ and $vy(t)$ represent the velocity components in the x and y directions of the vehicle at time t, respectively.

The motion of the vehicle is expressed by differential equations describing its position and velocity. If the vehicle is travelling at a constant speed, its velocity can be considered as a constant value. Hence, position update equation is given as in 3.7.

$$p(t + \Delta t) = p(t) + \int_t^{t+\Delta t} v(\tau)d\tau. \tag{3.7}$$

If the velocity is constant, this equation becomes simpler as follows.

$$p(t + \Delta t) = p(t) + v(t)\Delta t. \tag{3.8}$$

If acceleration $a(t)$ is measured, then the equations of motion can be derived by acceleration:

$$v(t + \Delta t) = v(t) + a(t)\Delta t \tag{3.9}$$

$$p(t + \Delta t) = p(t) + v(t)\Delta t + \frac{1}{2}a(t)(\Delta t)^2 \tag{3.10}$$

Let us consider the case of a vehicle travelling at constant speed such as in 3.11.

$$v(t) = v\widehat{u}. \tag{3.11}$$

Here v is the constant velocity and $\widehat{u}$ is the unit vector indicating the direction of motion. In this case the position update equation can be modified as in 3.12.

$$p(t + \Delta t) = p(t) + v\widehat{u}\Delta t. \tag{3.12}$$

Finally, considering Scenario-III, Djikstra algorithm was used as the shortest path algorithm for only leader vehicle. Algorithm 3 was benefited for Dijkstra algorithm in the context of this.

3.4 Configuration of Scenario-IV

In this scenario, unlike the previous 3 scenarios, there are also dynamic obstacles. The single mobile robot tries to go to the target on a map where dynamic obstacles are positioned as well as static obstacles. The operations in the scenario are expressed using pseudo codes.

In the main code of this scenario, various inputs are first received. These are width and height of the map, starting point, target point, current position of the vehicle, radius of the mobile robot, rotational angle, velocity, number of obstacles, dynamic obstacle size and dynamic obstacle velocity.

In the second step, the initialization of the map and obstacles is performed. The map is defined as a matrix of zeros of given width and height dimensions. Then, random static obstacles are placed on the map according to the specified

number of obstacles. In the assignment process for random static obstacles, it was benefited from Algorithm 1. After this process, dynamic obstacles are generated and positioned on this map. It has been performed in Algorithm 7.

In the third step, the shortest path from the starting to the target point is calculated using Dijkstra algorithm, taking into account the dynamic obstacles. This step aims to find the safest and shortest path by taking into account the movement of dynamic obstacles. Then, the movement of the vehicle while following this calculated path is simulated. This simulation records the paths travelled by the vehicle from the starting point to the target point and its interactions with dynamic obstacles.

Algorithm 7 Generating Dynamic Obstacles

Step 1. Inputs: DynamicObstacles, map
mapSize := size(map)
numDynamicObstacles := size(DynamicObstacles, 1)
for $i = 1$ ***to*** $numDynamicObstacles$ **do**
    x := DynamicObstacles(i, 1)
    y := DynamicObstacles(i, 2)
    while $map(y, x) == 1$ **do**
        x := random(1, mapSize(2))
        y := random(1, mapSize(1))
    end
    DynamicObstacles(i, 1) := x
    DynamicObstacles(i, 2) := y
end

The dynamic obstacles locations given in Algorithm 7 are randomly repositioned until they are placed in a non-valid location on the map. This process is repeated for the number of dynamic obstacles specified in the map. Finally, valid dynamic obstacles positions are returned.

Algorithm 8 Calculating the New Position of Mobile Robot for the Scenario-IV

Step 1.Input: map, mapWidth, mapHeight, startPoint, targetPoint, vehiclePosition, DynamicObstacles, robotRadius, rotationalAngle, v, shortestPath, DynamicObstacleSpeed
currentPoint = startPoint
path = [currentPoint]
movementLog = []
for $i = 1\ to\ length(shortestPath)$ **do**
  targetPoint = shortestPath[i]
  while $norm(targetPoint - currentPoint) > robotRadius$ **do**
    DynamicObstacles = updateDynamicObstacles(DynamicObstacles, DynamicObstacleSpeed, mapWidth, mapHeight, map, vehiclePosition, robotRadius)
    direction = tan^{-1}(targetPoint(2) - currentPoint(2), targetPoint(1) - currentPoint(1))
    [avoidObstacle, newDirection] = avoidObstacles(currentPoint, direction, DynamicObstacles, robotRadius, rotationalAngle)
    if $not\ avoidObstacle$ **then**
      currentPoint = moveVehicle(currentPoint, direction, v)
    end
    else
      currentPoint = moveVehicle(currentPoint, newDirection, v)
      direction = newDirection
    end
    path.append(currentPoint)
    movementLog.append([currentPoint, degrees(direction)])
  end
end
return $movementLog$;

In the first step of Algorithm 8, various inputs are taken such as the map, width and height of the map, starting point, target point, current position of the vehicle, dynamic obstacles, radius of the robot, rotational angle, velocity, shortest path and dynamic obstacle velocity. In the second step of the algorithm, the current point is determined as the starting point and the starting point is added by creating a path and movement log.

The algorithm proceeds step by step along the shortest path. For each target point, a cycle is started as long as the distance from the current point to the target point is greater than the radius of the robot. During this cycle, the positions of dynamic obstacles are updated and ensure that they remain within the boundaries of the map. The direction from the current point to the target point is calculated and then checked whether the robot needs to avoid dynamic obstacles. If the robot does not need to avoid obstacles, the robot moves in its current direction. However, if it needs to avoid obstacles, it moves in the newly calculated direction and this new

direction is updated. In both cases, the robot's new location is added to the path and movement log.

Algorithm 9 Avoidance of Dynamic Obstacles

Step 1.Input: currentPoint, direction, DynamicObstacles, robotRadius, rotationalAngle
avoidObstacle = false
newDirection = direction
for $eachDynamicObstacle$ ***in*** $DynamicObstacles$ **do**
    distanceToObstacle = norm(DynamicObstacle - currentPoint)
    if $distanceToObstacle \leq robotRadius$ **then**
        avoidObstacle = true
        newDirection = direction + deg2rad(rotationalAngle)
        break
    end
end
return $avoidObstacle, newDirection$

Algorithm 9 aims to enable a mobile robot to avoid dynamic obstacles. It detects dynamic obstacles based on the robot's current position and direction of movement and calculates a new direction to avoid obstacles. In the first step of the algorithm, various inputs are taken, such as the robot's current position, movement direction, list of dynamic obstacles, robot's radius, and rotational angle. In the second step, the flag (avoidObstacle) indicating the avoidance status and the new direction are defined. Initially the avoid flag is set to false and the new heading is set to be the same as the current heading.

The algorithm checks each dynamic obstacle. For each of these obstacles, the distance between the current position and the obstacle is calculated. If this distance is equal to or less than the robot's radius, it is determined that the robot must avoid an obstacle. If there is an obstacle avoidance situation, the avoidance flag is set to true and the new direction is updated by adding the rotational angle to the current direction. In this case, the algorithm stops checking other obstacles, breaking the loop.

Algorithm 10 Dijkstra Algorithm for the Scenario-IV

Input: $map, startPoint, targetPoint, DynamicObstacles, robotRadius$

```
visited = false
distances = Inf
distances(startPoint(2), startPoint(1)) = 0
while true do
    minDistance = Inf
    minNode = [ ]
    for y = 1 to mapSize(2) do
        for x = 1 to mapSize(1) do
            if not visited(y, x) and distances(y, x) < minDistance then
                minDistance = distances(y, x)
                minNode = [x, y]
            end
        end
    end
    if minNode is empty or minNode == targetPoint then
        break
    end
    visited(minNode(2), minNode(1)) = true
    neighbors = getNeighbors(minNode, mapSize)
    for each neighbor in neighbors do
        if not visited(neighbor(2), neighbor(1)) and map(neighbor(2), neighbor(1)) != 1 and not isDynamicObstacleHere(neighbor, DynamicObstacles, robotRadius) then
            newDistance = distances(minNode(2), minNode(1)) + 1
            if newDistance < distances(neighbor(2), neighbor(1)) then
                distances(neighbor(2), neighbor(1)) = newDistance
            end
        end
    end
end
path = [targetPoint]
currentNode = targetPoint
while currentNode! = startPoint do
    neighbors = getNeighbors(currentNode, mapSize)
    minDistance = Inf
    minNode = [ ]
    for each neighbor in neighbors do
        if distances(neighbor(2), neighbor(1)) < minDistance then
            minDistance = distances(neighbor(2), neighbor(1))
            minNode = neighbor
        end
    end
    if minNode is empty then
        return [ ]
    end
    path = [minNode; path]
    currentNode = minNode
end
return path
```

Algorithm 10 aims to find the shortest path from the starting point to the target point on a map with dynamic obstacles, using the Dijkstra algorithm for Scenario-IV. In the first step of the algorithm, various inputs are taken, such as the map, starting point, target point, dynamic obstacles and the radius of the robot. In the second step of the algorithm, matrices are created to keep the visited nodes and the distance of each node from the starting point. The distance of the starting point is set to zero and the distances for all other points are defined as infinite. The algorithm constantly searches for the unvisited node (minNode) with the shortest distance. By checking each point on the map, the node with the smallest distance is determined. The selected closest node (minNode) is marked as visited. Then, by benefiting from Algorithm 3, the neighbors of this node are determined and for each neighbor, unvisited, non-obstructed and dynamic obstruction-free are checked. A new distance is calculated for eligible neighbors and updated if this distance is smaller than the current distance. The algorithm exits the loop when the target point is reached or all nodes have been visited. It then backtracks the shortest path starting from the destination point and adds each node to the path. This step continues until the starting point is reached.

The algorithm ultimately ends up returning a path containing the shortest path from the starting point to the target point. If no path to the destination is found, an empty list is returned. This path allows the robot to follow the safest and shortest path, taking into account dynamic obstacles.

Algorithm 11 Finding the Dynamic Obstacles in the Map

Function `isDynamicObstacleHere`$(neighbor, DynamicObstacles, robotRadius)$**:**
  for $eachobstacle$ ***in*** $DynamicObstacles$ **do**
    if $distance(neighbor, obstacle) < robotRadius$ **then**
      return $true$
    end
  end
  return $false$

Algorithm 11, checks whether dynamic obstacles exist within a certain radius (robotRadius) of a node (neighbor) and returns a boolean value based on this information. In the first step of the algorithm, the inputs are the neighbor node, the list of dynamic obstacles, and the radius of the robot. The algorithm checks each dynamic obstacle. For each of these obstacles, the distance between the neighboring node and the dynamic obstacle is calculated. If the distance between the neighboring node and the dynamic obstacle is less than the radius of the robot, it is determined that there is a dynamic obstacle in this node and the function returns true. If no obstacles are found after checking all dynamic obstacles, the function

returns false.

This algorithm is used to detect whether there are dynamic obstacles near a particular node. This information is used by other algorithms to enable the robot to avoid dynamic obstacles.

3.5 Configuration of Scenario-V

In this scenario, unlike the Scenario-IV, there are two mobile robots. The mobile robots calculate the shortest path using Dijkstra algorithm and develop instantaneous strategies to avoid collisions with dynamic obstacles and each other while following this path.

This scenario is logically the same as the previous scenario. The only difference is the presence of the second mobile robot. For this scenario, it was benefited from Algorithm 5, Algorithm 9, Algorithm 10 and Algorithm 11 in terms of avoidance the collision between vehicles, avoidance the collision with dynamic obstacles and in terms of Dijkstra algorithm.

4
SIMULATION STUDIES

In this section, the outputs of the simulation studies and the important parameters used in the simulations are detailed and explained with graphs. The kinematic equations for the mobile robot vehicle model used in the simulations are given in Chapter 2 in 2.2.1. The common parameters used in all simulations are given in the Table 4.1.

Table 4.1 Common parameters for simulations

Parameters	Value
v	0.1 m/s
r	0.5 m
d	0.8 m

At the end of each simulation, the obtained rotation angle information has been taken as reference and given as input to a feedback system created using a PID controller to determine whether the maneuver of the vehicle can be controlled by the PID controller in a real application. The main criterion in the design of PID parameters is the maximum allowable overshoot and settling time value. In each scenario, the upper limits for T_s and $\%$OS values are T_s = 0.234 seconds and $\%$OS = $\%$1.58. Parameters were found using 'MATLAB PID Tuner' by considering these constraints. Common PID parameters were used in each scenario. The values of these parameters are shown in Table 4.2. Furthermore, the feedback system generated at the end of each scenario is shown in Figure 4.1.

Table 4.2 PID parameters for each scenario

Parameters	Value
K_p	80
K_i	400
K_d	85

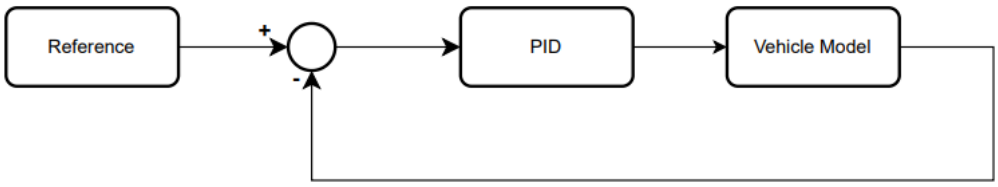


Figure 4.1 Block diagram for system

4.1 Simulation Studies for Scenario-I

In this scenario, simulation studies of the target reaching problem of a single mobile robot in an environment full of static obstacles are presented. In the studies, Dijkstra algorithm is used to ensure that the robot finds the shortest path and reaches the target without colliding with obstacles. The algorithms used in the scenario are given in pseudo codes which called Algorithm 1, Algorithm 2 and Algorithm 3 in Chapter 3.

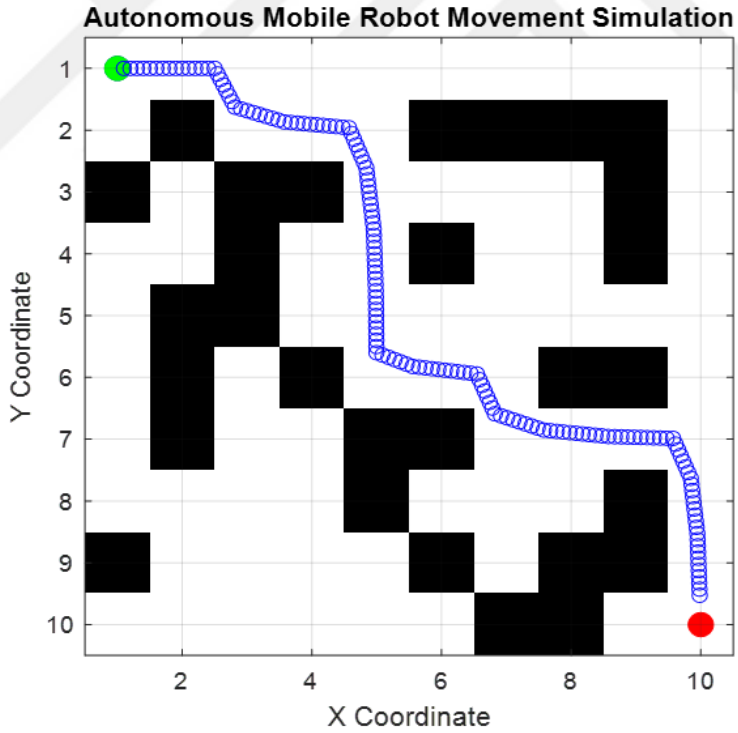


Figure 4.2 Simulation of Scenario-I

Figure 4.2 shows the simulation study of the Scenario I visually. The green circle is the starting point, the red circle is the target point and the blue circles with dashed lines show the locations that the mobile robot travelled until reaching the target.

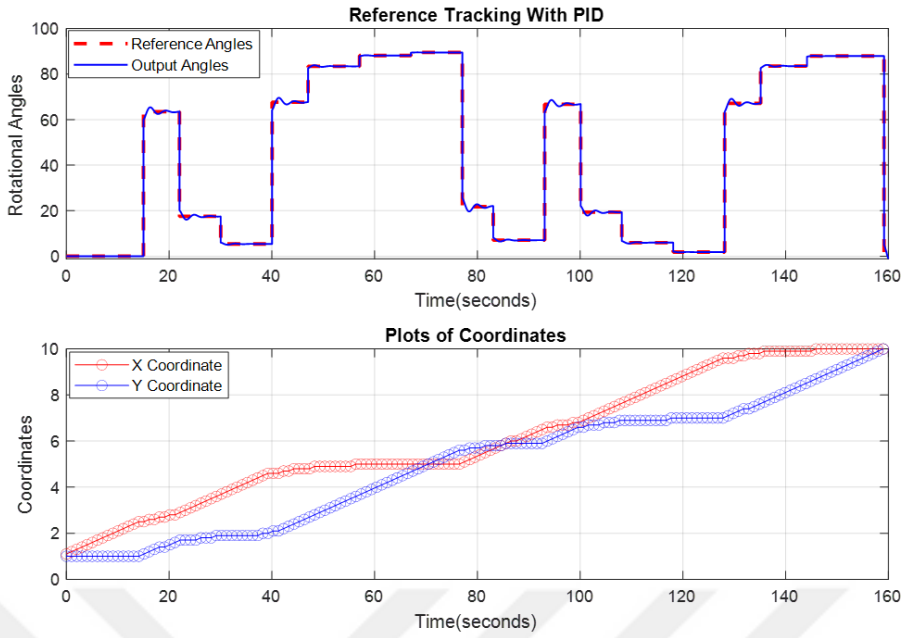


Figure 4.3 PID response for Scenario-I

Figure 4.3 shows the response of the PID controller based on the rotation angle information obtained from the simulation of the Scenario I. In order to see the reference tracking capability of the PID controller, the reference values are shown with dashed red lines and the response of the PID controller is shown with blue lines. The figure below the PID response shows the instantaneous rotation angle information of the vehicle at which time and at which coordinate. Red round lines represent X coordinate information while blue round lines represent Y coordinate information. The simulation was carried out using certain map topology and obstacle configurations. The results show that the Dijkstra algorithm works effectively, the robot reaches the target safely and the reference tracking capability of the PID controller is sufficient in this scenario.

An important metric for evaluating the performance and accuracy of the PID control system is the Root Mean Square (RMS) error value. The RMS error value is calculated as the square root of the mean square of the error signals over time and better represents the overall performance of the system, taking into account the effect of error fluctuations. Ideally, the smaller the RMS error value, the better the system tracks the reference signal and the lower the amount of error. This indicates that the system produces the desired output more accurately and achieves the control objectives more easily. However, the acceptable RMS error value may vary depending on the type of system, the requirements of the application, and the complexity of the system. For example, in a precision control application, such as

medical device control, the acceptable RMS error value is very small, while in less precise applications, such as industrial process control, a larger RMS error value may be acceptable. In this context, the RMS value of the Scenario-I was found as 0.56926.

4.2 Simulation Studies for Scenario-II

In this scenario, simulation studies of the target reaching problem of two mobile robots in an environment full of static obstacles are presented. In the first of the two simulations, the Dijkstra algorithm is used for both robots to find the shortest path and reach the target without colliding the obstacles. In the second simulation, the leader vehicle or main vehicle which has the other name is intended to reach the target according to the Dijkstra algorithm and the follower vehicle is intended to reach the target by following the path of the main vehicle. The algorithms used in this scenario are given in Chapter 3 with pseudo codes which called Algorithm 4 and Algorithm 5.

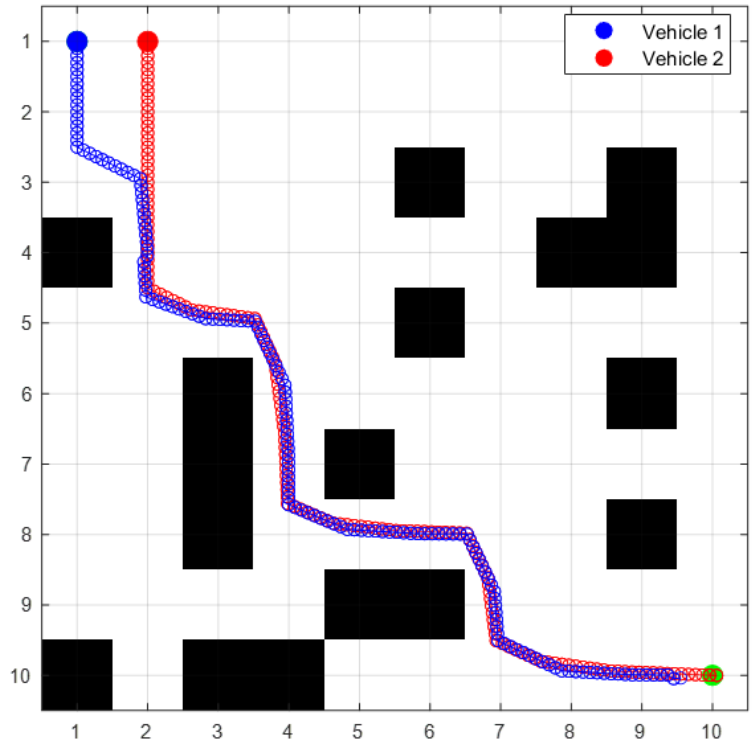


Figure 4.4 Simulation of Scenario-II

Figure 4.4 shows the simulation study of Scenario-II visually. The blue circle represents the starting point of the main vehicle (Vehicle 1), the red circle represents the starting point of the other vehicle (Vehicle 2), the green circle represents the

target point, the dashed blue lines represent the path of the main vehicle until it reaches the target and the dashed red lines represent the path of the other vehicle until it reaches the target. According to the path of the main vehicle, the collision control was carried out twice and the main vehicle maneuvred by 30° at that times.

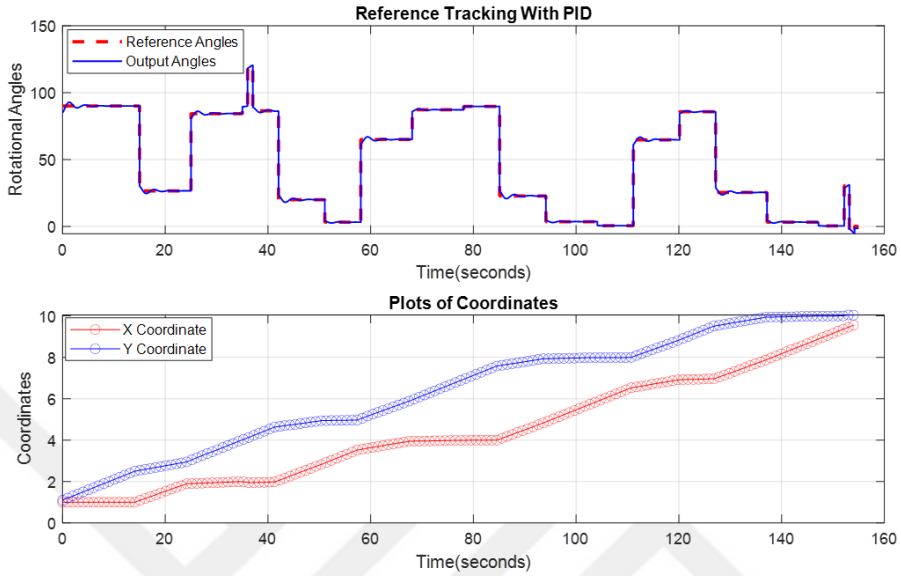


Figure 4.5 PID response for Scenario-II

Figure 4.5 shows the response of the PID controller based only on the main vehicle rotation angle information obtained from the simulation of Scenario-II. To see the reference tracking capability of the PID controller, the reference values are shown with dashed red lines and the response of the PID controller is shown with blue lines.

The simulation was carried out using certain map topology and obstacle configurations. The results show that Dijkstra algorithm works effectively for both vehicles, the vehicles reach safely to the target based on collision control, and the reference tracking capability of the PID controller based on the rotational angles of the main vehicle is sufficient for the desired criteria in this simulation. Finally, the RMS value of the Scenario-II was found as 0.69736.

4.3 Simulation Studies for Scenario-III

The other simulation in this study which called Scenario-III, it has been presented simulation studies that investigate the leader-follower approach to the target reaching problem of two mobile robots in an environment full of static obstacles. In the studies, the first robot pre-computes the shortest path using Dijkstra algorithm

and follows this path. The second robot reaches the target by following the first robot. The algorithms used in this scenario are given in pseudo codes in Chapter 3.

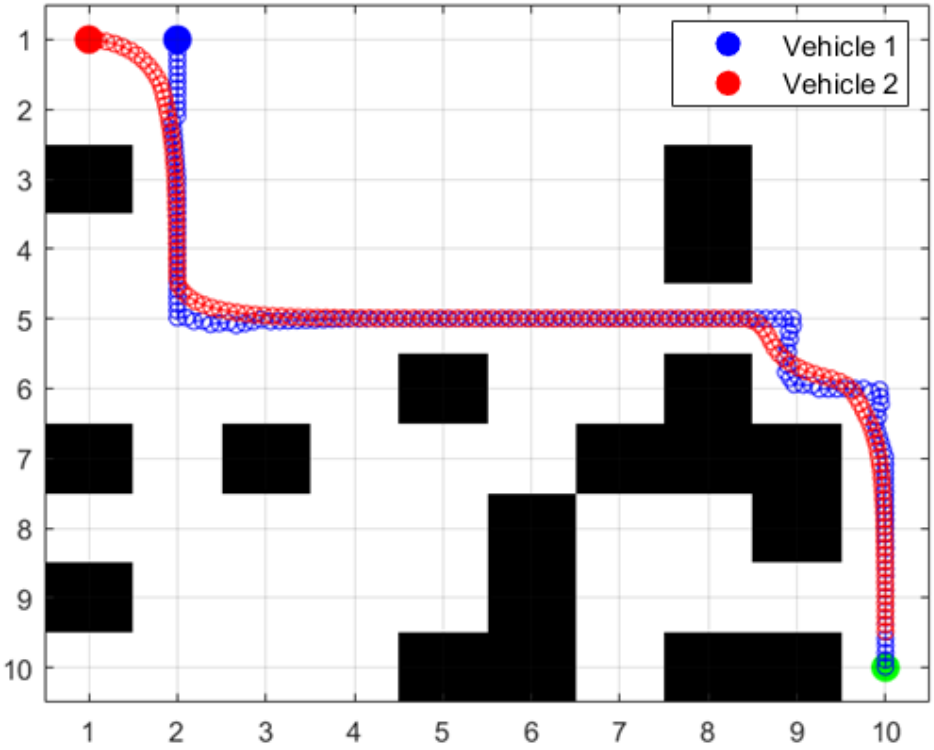


Figure 4.6 Simulation of Scenario-III

Figure 4.6 shows the simulation study of Scenario-III visually. The red circle represents the starting point of the follower vehicle (Vehicle 2), the blue circle represents the starting point of the leader vehicle (Vehicle 1), the green circle represents the target point, the dashed red lines represent the path of the follower vehicle until it reaches the target and the dashed blue lines represent the path of the leader vehicle until it reaches the target. The parameters related to this scenario are given in the Table 4.3.

Table 4.3 Important parameters for Scenario-III

v1	0.1 m/s
v2	0.1 m/s
r	0.5 m
θ_l	30°

In order for the follower vehicle to reach the target simultaneously while following the leader vehicle and to draw a good route, this simulation was carried out by

testing these parameters. The parameter v1 represents the velocity of the leader vehicle, v2 represents the velocity of the follower vehicle, r represents the radius of the vehicles and θ_l represents the maneuver of leader vehicle when the collision control between vehicles happened. The radius value is equal for both vehicles. Then, it is aimed to perform reference tracking by means of PID controller by taking the rotation angle informations of the leader vehicle as a reference from the data obtained as a result of the simulation. The related visualisation is shown in Figure 4.7.

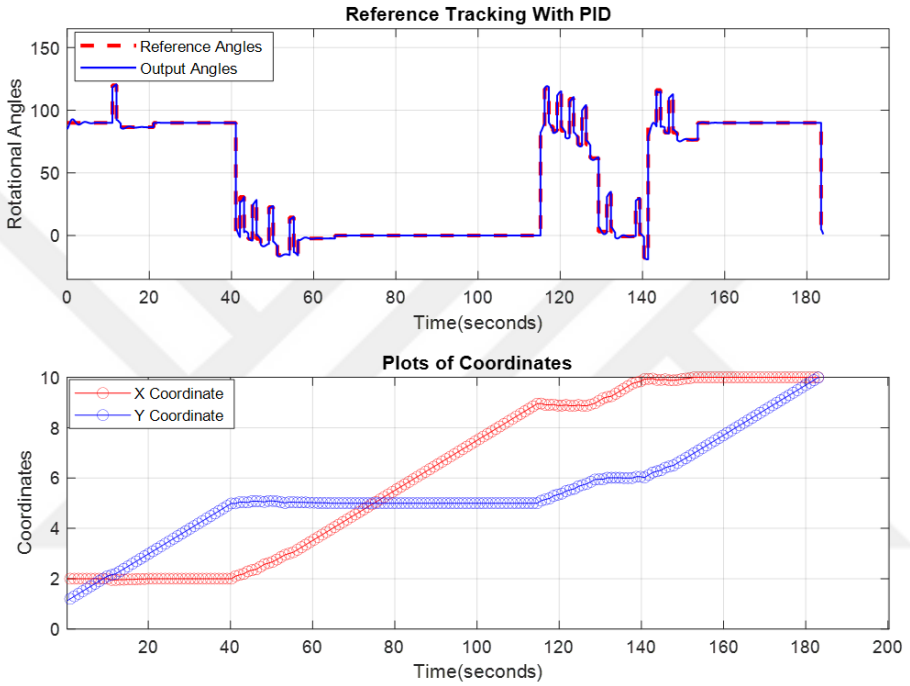


Figure 4.7 PID response for Scenario-III

Figure 4.7 shows the response of the PID controller based only on the leader vehicle rotational angle information obtained from the Scenario II-b simulation. To see the reference tracking ability of the PID controller, the reference values are shown as dashed red lines, and the response of the PID controller is shown as blue lines. Although the reference tracking ability of the controller is sufficient, there is some chattering in the PID response due to angle changes based on collision control that occur at some moments of the simulation. Parts with chattering can be called disturbance. Finally, the RMS value of the Scenario-III was found as 0.90077.

4.4 Simulation Studies for Scenario-IV

In this scenario, dynamic obstacles are also placed in addition to the first scenario. The mobile robot travel from starting point to the target point according to the Dijkstra algorithm on a map with both static and dynamic obstacles. The algorithms used in this scenario are given with pseudo codes which called Algorithm 7, Algorithm 8, Algorithm 9, Algorithm 10, and Algorithm 11 in Chapter 3. The figures related to simulation studies within the scope of this scenario are given below. The figures are taken depending on the change of the vehicle's rotational angle value and the maneuver it makes when encountering dynamic obstacles.

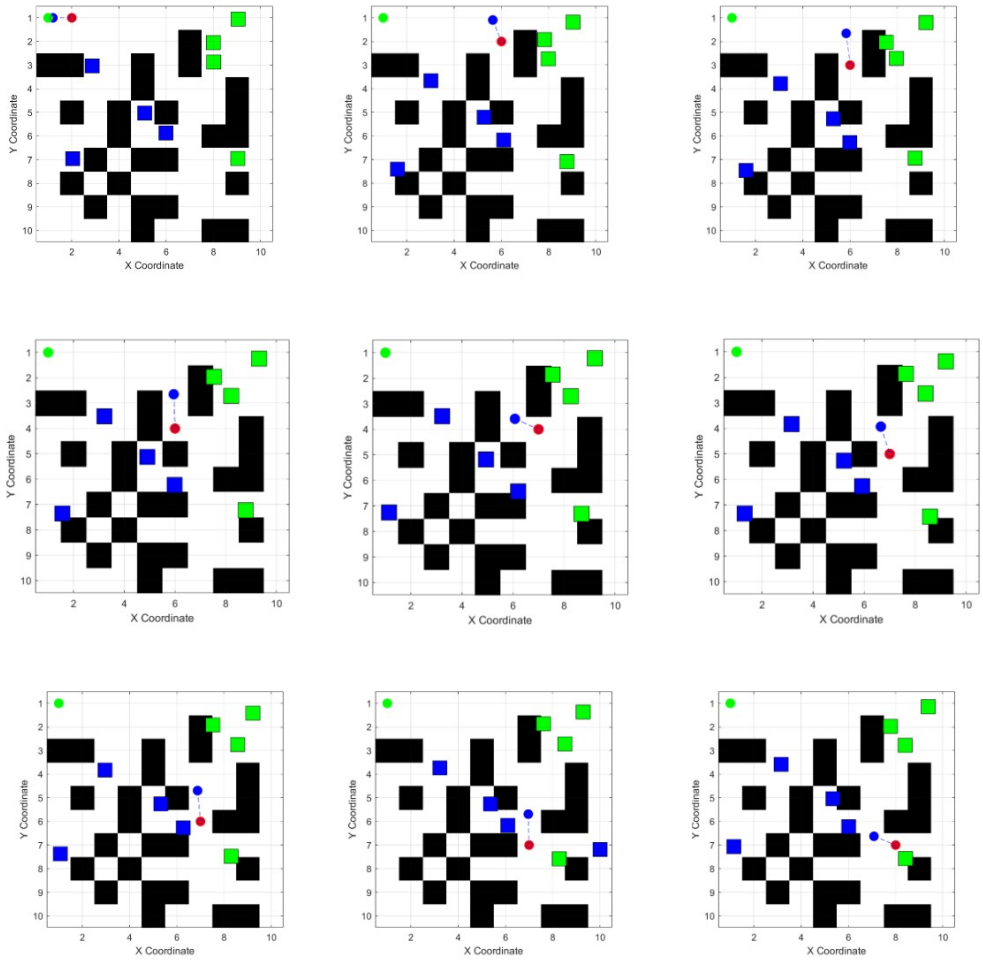


Figure 4.8 Images for Scenario-IV - 1 to 9

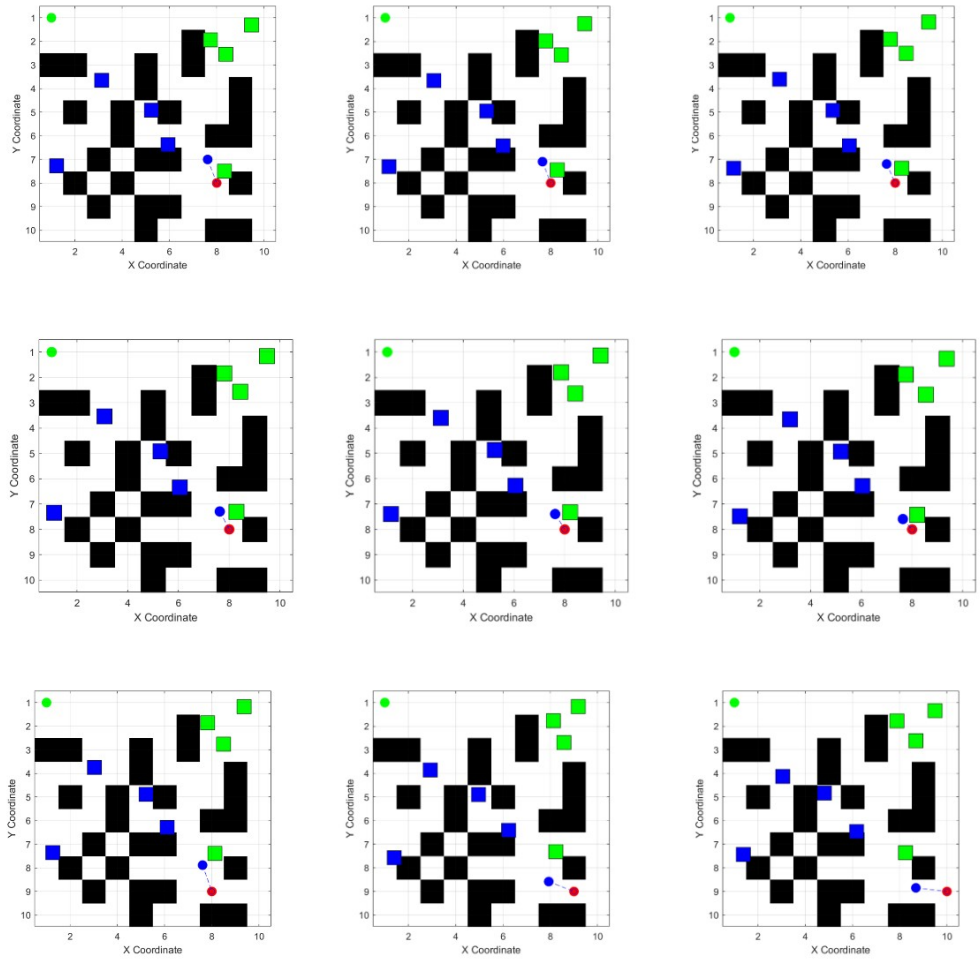


Figure 4.9 Images for Scenario-IV - 10 to 18

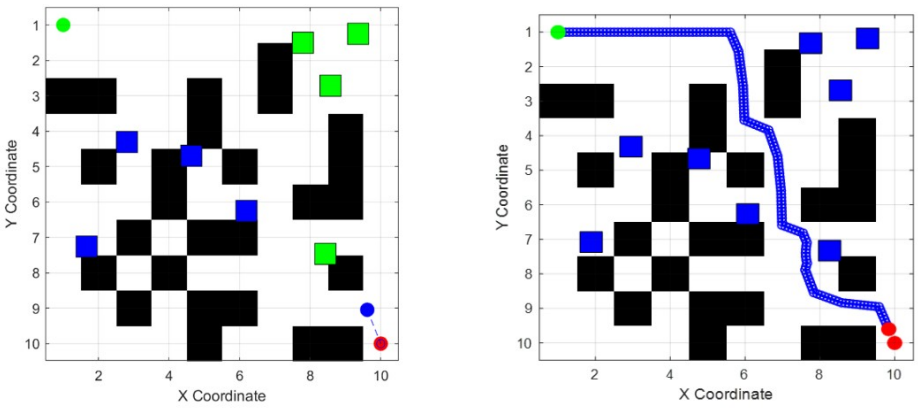


Figure 4.10 Images for Scenario-IV - 19 and 20

These images were taken from the moments when the rotation angle of the mobile robot changed and the collision check was performed, including the beginning

and end of the simulation. From the eleventh to the fifteenth image, it can be seen that the vehicle encountered a dynamic obstacle and collision control was performed accordingly. In other images, the current map was recorded based on the moments when the rotation angle value changed. In the last image, the simulation is completed and the path followed by the mobile robot throughout the simulation is drawn. The reason the images are recorded here is to show that dynamic obstacles are moving. The map is recorded instantly in each image and the instantaneous locations of dynamic obstacles are observed. The recorded images show that the dynamic obstacles are moving because the position of the dynamic obstacles changes.

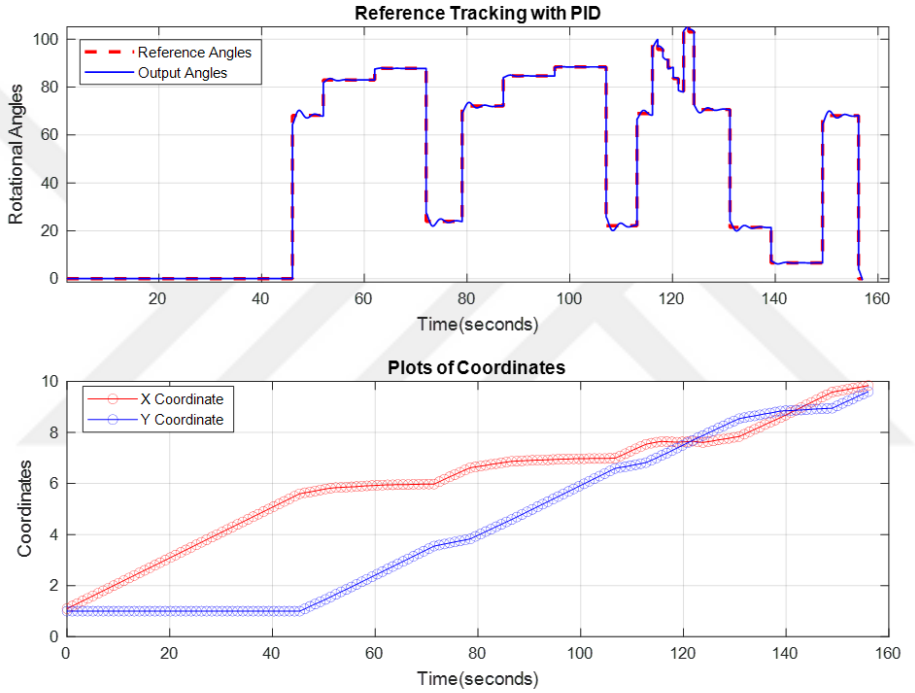


Figure 4.11 PID response for Scenario-IV

Figure 4.11 shows that reference tracking is done by the PID controller, taking the rotation angle informations obtained from the Scenario-IV as reference. Accordingly, it was seen that the PID controller's reference tracking capability was sufficient in this scenario. Finally, the RMS value of the Scenario-IV was found as 0.58098.

4.5 Simulation Studies for Scenario-V

In this scenario, simulation studies were carried out by assuming that there are two mobile robots and dynamic and static obstacles on a map. In this manner, two

simulations were performed. The first simulation is a simulation with collision control between vehicles and dynamic obstacles. The second simulation is a simulation with collision control between vehicles. For both mobile robots, Dijkstra algorithm was used to reach the target. Related parameters, figures and results of the simulations are given below.

Table 4.4 Parameters for Scenario-V.a

v1	0.1 m/s
v2	0.1 m/s
r	0.5 m
θ_m	30°

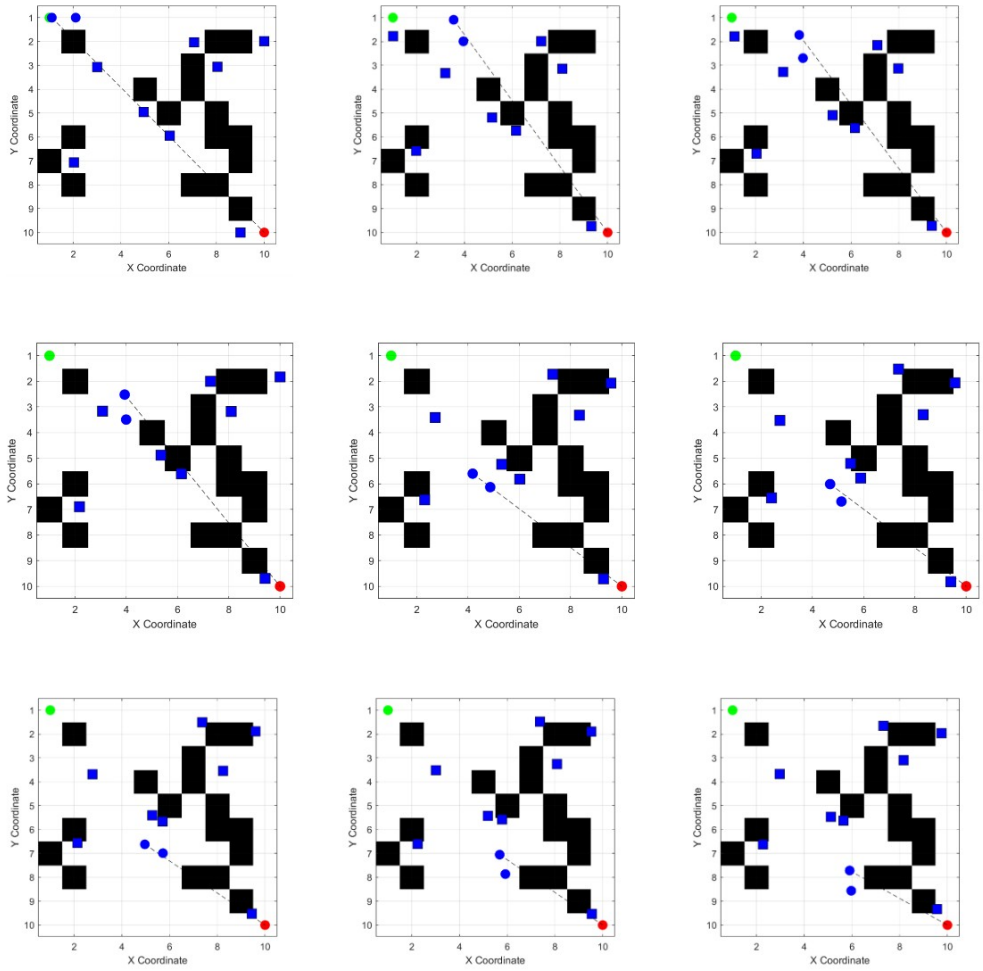


Figure 4.12 Images for Scenario-V.a - 1 to 9

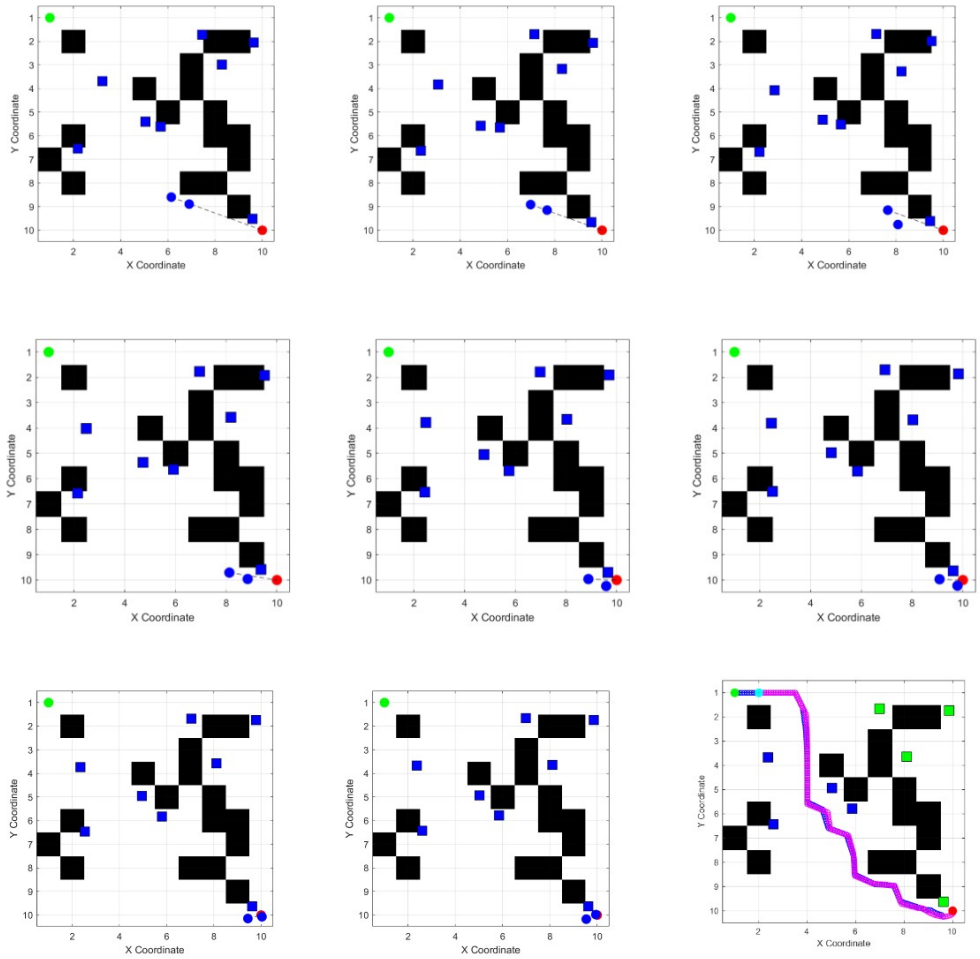


Figure 4.13 Images for Scenario-V.a - 10 to 18

Based on Figures 4.12 and 4.13, the vehicle at point [1, 1] is the main vehicle, i.e. vehicle 1, while the vehicle at point [2, 1] is vehicle 2. Considering the change of the rotational angle values, the above images were obtained and it was aimed to detect the movement of these obstacles by showing that the position of the dynamic obstacles changed in these images. From the thirteenth image to the fifteenth image, the manoeuvres of the vehicles based on the collision control are shown when they get close to the dynamic obstacle for a certain distance. This maneuvre is 30° as indicated in Table 4.4. The last image shows the paths followed by the vehicles throughout the simulation.

Table 4.5 Parameters for Scenario-V.b

v1	0.2 m/s
v2	0.1 m/s
r	0.5 m
θ_m	30°

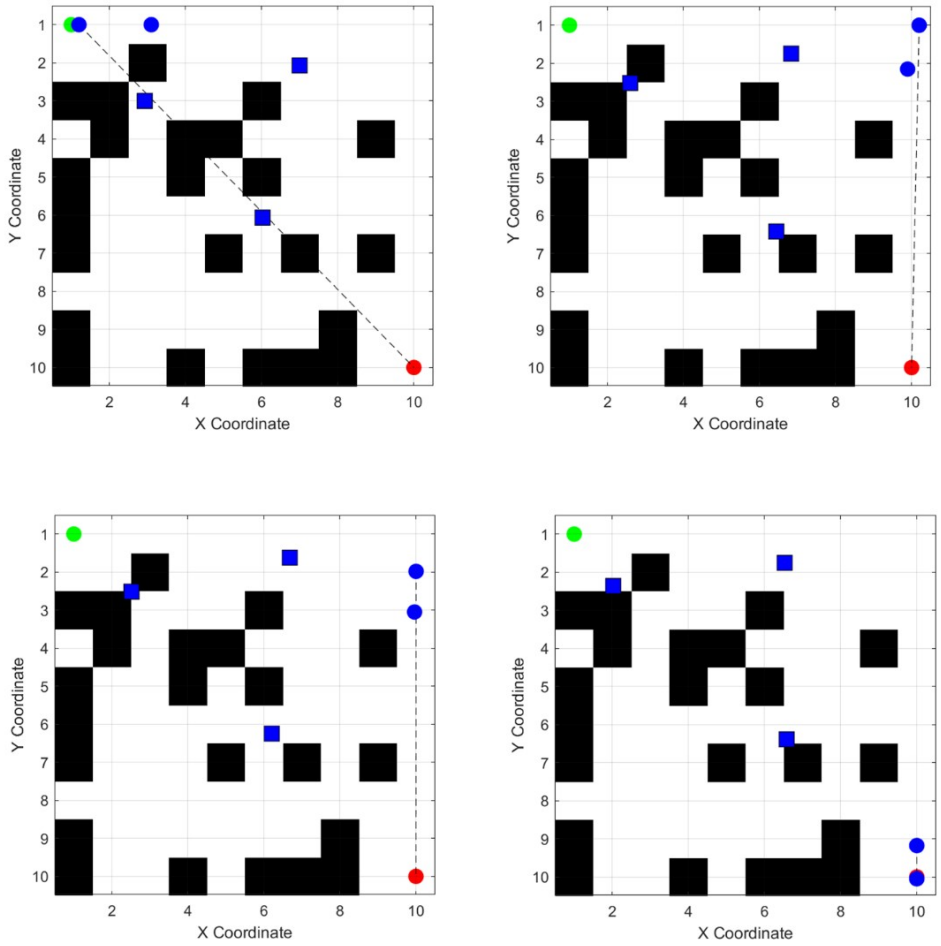


Figure 4.14 Images for Scenario-V.b - 1 to 4

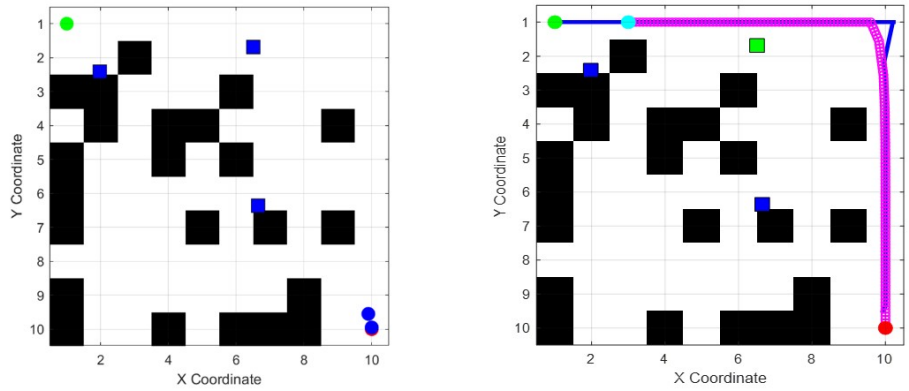


Figure 4.15 Images for Scenario-V.b - 5 and 6

Based on Figures 4.14 and 4.15, the vehicle at point [1, 1] is the main vehicle, i.e. vehicle 1, while the vehicle at point [3, 1] is vehicle 2. Considering the change of the rotational angle values, the above images were obtained and it was aimed to detect the movement of these obstacles by showing that the position of the dynamic obstacles changed in these images. From the fourth image to the fifth image, the manoeuvre of the vehicle 1 based on the collision control are shown when they get close to the vehicle 2 for a certain distance. This maneuvre is 30° as indicated in Table 4.5. Besides, vehicle 1 velocity is indicated as 0.2 m/s unlike Scenario V.a. Because, when the simulation was performed in the equal velocity, collision control between vehicles was not detect. The last image shows the paths followed by the vehicles throughout the simulation.

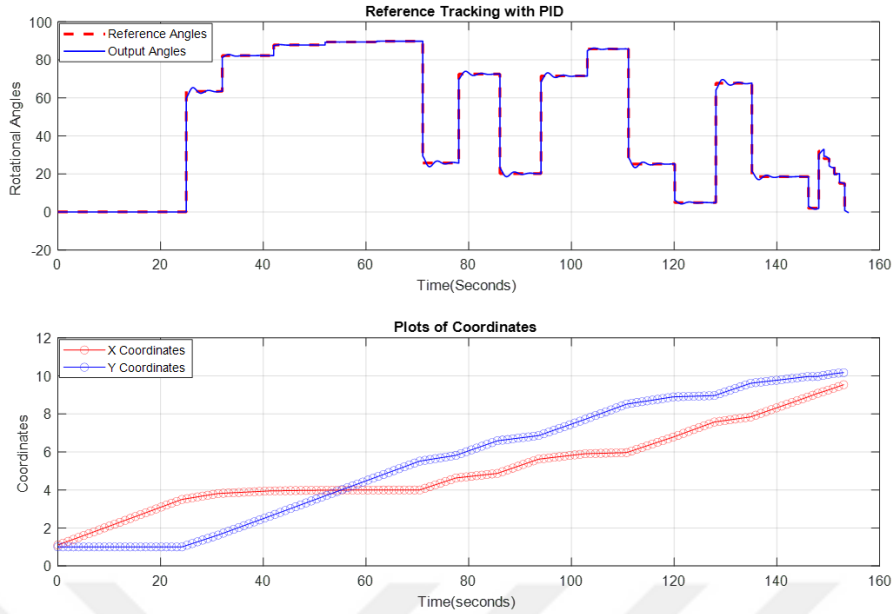


Figure 4.16 PID response for Scenario-V.a

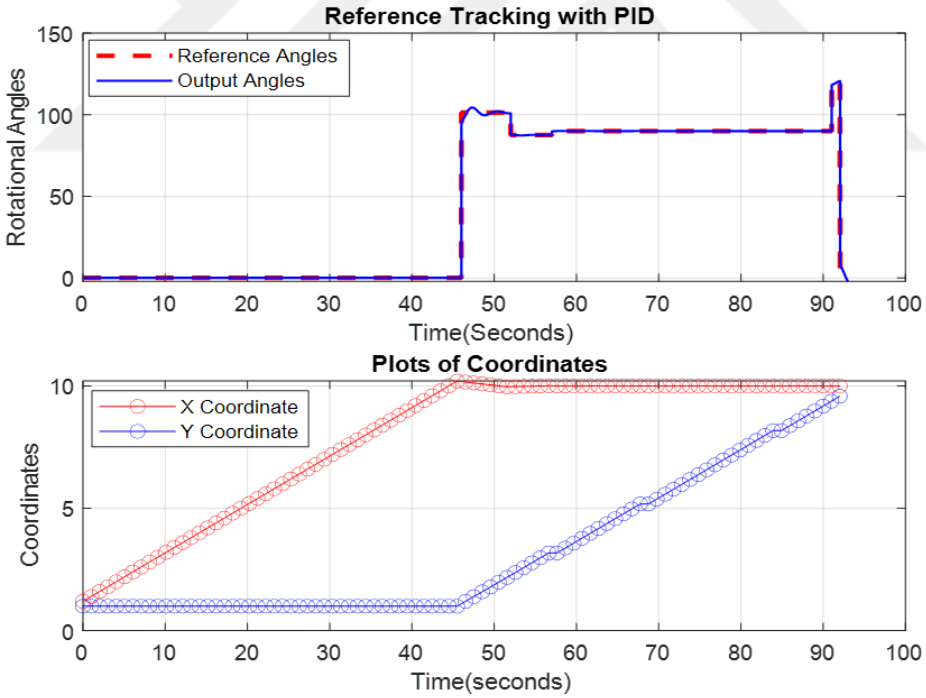


Figure 4.17 PID response for Scenario-V.b

Figure 4.16 and 4.17 show that reference tracking is done by the PID controller, taking the main mobile robot's rotation angle informations obtained from the Scenario-V.a and Scenario-V.b as reference. Accordingly, it was seen that the PID controller's reference tracking capability was sufficient in this scenario. Finally, the

RMS value of the Scenario-V.a was found as 0.56938 and Scenario-V.b was found as 0.61426.

5
DISCUSSIONS

5.1 Discussions on Research Outcomes

In this study, the optimal path finding problem for mobile robots is considered and Dijkstra algorithm is preferred. Dijkstra algorithm is one of the oldest and most well-known path finding algorithms. Therefore, it has been extensively studied in the literature and applied in different scenarios. This has led to a good understanding of the working principles of the algorithm and to the foreknowledge of potential problems. The literature review has shown that Dijkstra algorithm performs robustly and reliably on small-scale maps. This is an important factor considering the scale and complexity of the simulation scenarios used in the study. Dijkstra algorithm has the ability to calculate accurate routes unambiguously. It also has a simpler structure compared to some other path finding algorithms. In this way, it is easier to code and integrate into the simulation environment. Dijkstra algorithm is a time-efficient algorithm that can quickly produce solutions on small and medium-sized maps. This is critical for real-time applications.

Of course, other path finding algorithms such as A*, D*, VFH, APF and RRT could have been used in this study. However, the above-mentioned properties of Dijkstra algorithm make it the most suitable choice for algorithm selection in this study.

Another issue is the choice of controller. As a controller, Model Predictive Control (MPC) could have been preferred instead of PID. However, within the scope of the hypothesis of this thesis, it is emphasised whether PID, which is known as the easiest controller to design, will be sufficient in the studies carried out within the scope of this thesis. Therefore, PID is used as the controller.

5.2 Future Study Suggestions

Comparing the performance of different path finding algorithms, development of more advanced control strategies for navigation in complex environments,

designing coordination and communication protocols for multi-robot systems and testing Dijkstra algorithm and PID controller in real world experiments. These studies will help the development of autonomous navigation systems and the use of mobile robots in a wider range of applications.

One of the most important tasks in autonomous navigation is to find the shortest and safest path from the starting point to the target point. For this purpose, various path finding algorithms such as Dijkstra algorithm, A* algorithm and D* algorithm have been developed. It is an important research topic to compare the performance of these algorithms in different environments and with different robot kinematics and to determine the most appropriate algorithm.

Real-world environments present various challenges, ranging from static obstacles to dynamic ones, variable lighting conditions, and rough terrain. More advanced control strategies are needed for safe and efficient navigation in these complex environments. Techniques such as fuzzy logic, artificial neural networks and reinforcement learning can be used to improve the decision making and control capabilities of robots in such environments.

Autonomous navigation systems that enable multiple robots to work in coordination can be useful in many areas. For example, multiple robots can be used in search and rescue operations, agricultural fields or logistics warehouses. In such systems, there is a need for coordination and communication protocols that prevent collisions between robots and enable them to complete tasks effectively.

The Dijkstra algorithm is used to find the shortest path, but this may not always be the best path. For example, the shortest path may be rough or dangerous. By integrating Markov Decision Processes (MDP) with Dijkstra algorithm, a decision mechanism can be added in path planning. MDP can help the robot to choose the optimal path considering the uncertainties and risks in its environment.

One of the most common controllers used in autonomous navigation is the PID controller. However, more advanced controllers are also available. Comparing the performance of different controllers such as fuzzy logic and MPC and determining which controller is better in which scenarios is an important research topic.

6
CONCLUSION

In this thesis, the performance of Dijkstra algorithm as an optimal path finding algorithm for mobile robots is investigated using five simulation scenarios of increasing complexity. Experiments have shown that Dijkstra algorithm can operate efficiently even in complex environments consisting of static and dynamic obstacles. The algorithm was also able to support coordinated movement of multiple vehicles and leader-follower configurations.

Furthermore, a PID controller was designed and tested to follow the routes generated by Dijkstra algorithm. The controller enabled the mobile robots to efficiently track the reference rotational angles under different map shapes and obstacle configurations.

The results indicate that the Dijkstra algorithm is well-suited for pathfinding in autonomous navigation applications. When combined with a PID controller, it allows mobile robots to navigate safely and efficiently through complex environments.

REFERENCES

[1] T. Palaniswamy, A. Alofi, F. S. Alghamdi, N. M. Aljuaid, R. Alahmadi, "Automated mobile robots–a survey on controls," *Communications on Applied Electronics*, vol. 6, no. 9, pp. 22–26, 2017.

[2] A. V. Savkin, A. S. Matveev, M. Hoy, C. Wang, *Safe robot navigation among moving and steady obstacles*. Butterworth-Heinemann, 2015.

[3] A. Loganathan, N. S. Ahmad, "A systematic review on recent advances in autonomous mobile robot navigation," *Engineering Science and Technology, an International Journal*, vol. 40, p. 101 343, 2023.

[4] N. Ding, C. Peng, M. Lin, C. Wu, "A comprehensive review on automatic mobile robots: Applications, perception, communication and control," *Journal of Circuits, Systems and Computers*, vol. 31, no. 08, p. 2 250 153, 2022.

[5] A. N. A. Rafai, N. Adzhar, N. I. Jaini, "A review on path planning and obstacle avoidance algorithms for autonomous mobile robots," *Journal of Robotics*, vol. 2022, no. 1, p. 2 538 220, 2022.

[6] J. Crowley, "Navigation for an intelligent mobile robot," *IEEE Journal on Robotics and Automation*, vol. 1, no. 1, pp. 31–41, 1985.

[7] S. Deshpande, A. K. Kashyap, B. K. Patle, "A review on path planning ai techniques for mobile robots," *Robotic Systems and Applications*, vol. 3, no. 1, pp. 27–46, 2023.

[8] S. K. Sahoo, B. B. Choudhury, "A review of methodologies for path planning and optimization of mobile robots," *Journal of process management and new technologies*, vol. 11, no. 1-2, pp. 122–140, 2023.

[9] H. Qin, S. Shao, T. Wang, X. Yu, Y. Jiang, Z. Cao, "Review of autonomous path planning algorithms for mobile robots," *Drones*, vol. 7, no. 3, p. 211, 2023.

[10] N. Buniyamin, W. W. Ngah, N. Sariff, Z. Mohamad, *et al.*, "A simple local path planning algorithm for autonomous mobile robots," *International journal of systems applications, Engineering & development*, vol. 5, no. 2, pp. 151–159, 2011.

[11] J. R. Sanchez-Ibanez, C. J. Pérez-del-Pulgar, A. García-Cerezo, "Path planning for autonomous mobile robots: A review," *Sensors*, vol. 21, no. 23, p. 7898, 2021.

[12] M. Zhou, N. Gao, "Research on optimal path based on dijkstra algorithms," in *3rd International Conference on Mechatronics Engineering and Information Technology (ICMEIT 2019)*, Atlantis Press, 2019, pp. 884–892.

[13] J. Pak, J. Kim, Y. Park, H. I. Son, "Field evaluation of path-planning algorithms for autonomous mobile robot in smart farms," *IEEE Access*, vol. 10, pp. 60 253–60 266, 2022.

[14] M. Dakulović, I. Petrović, "Two-way d algorithm for path planning and replanning," *Robotics and autonomous systems*, vol. 59, no. 5, pp. 329–342, 2011.

[15] D. Ferguson, A. Stentz, "The field d* algorithm for improved path planning and replanning in uniform and non-uniform cost environments," *Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-05-19*, 2005.

[16] A. Stentz *et al.*, "The focussed dˆ* algorithm for real-time replanning," in *IJCAI*, vol. 95, 1995, pp. 1652–1659.

[17] S. Abdallaoui, E.-H. Aglzim, A. Chaibet, A. Kribèche, "Thorough review analysis of safe control of autonomous vehicles: Path planning and navigation techniques," *Energies*, vol. 15, no. 4, p. 1358, 2022.

[18] H. Zhang, Y. Wang, J. Zheng, J. Yu, "Path planning of industrial robot based on improved rrt algorithm in complex environments," *IEEE Access*, vol. 6, pp. 53 296–53 306, 2018.

[19] I. Noreen, A. Khan, Z. Habib, "A comparison of rrt, rrt* and rrt*-smart path planning algorithms," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 16, no. 10, p. 20, 2016.

[20] C. W. Warren, "Multiple robot path coordination using artificial potential fields," in *Proceedings., IEEE International Conference on Robotics and Automation*, IEEE, 1990, pp. 500–505.

[21] M. G. Park, J. H. Jeon, M. C. Lee, "Obstacle avoidance for mobile robots using artificial potential field approach with simulated annealing," in *ISIE 2001. 2001 IEEE International Symposium on Industrial Electronics Proceedings (Cat. No. 01TH8570)*, IEEE, vol. 3, 2001, pp. 1530–1535.

[22] F. Bounini, D. Gingras, H. Pollart, D. Gruyer, "Modified artificial potential field method for online path planning applications," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2017, pp. 180–185.

[23] J. Borenstein, Y. Koren, *et al.*, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE transactions on robotics and automation*, vol. 7, no. 3, pp. 278–288, 1991.

[24] W. J. Yim, J. B. Park, "Analysis of mobile robot navigation using vector field histogram according to the number of sectors, the robot speed and the width of the path," in *2014 14th International Conference on Control, Automation and Systems (ICCAS 2014)*, IEEE, 2014, pp. 1037–1040.

[25] B. I. Kazem, A. H. Hamad, M. M. Mozael, "Modified vector field histogram with a neural network learning model for mobile robot path planning and obstacle avoidance.," *Int. J. Adv. Comp. Techn.*, vol. 2, no. 5, pp. 166–173, 2010.

[26] P. Morin, C. Samson, "Motion control of wheeled mobile robots.," *Springer handbook of robotics*, vol. 1, pp. 799–826, 2008.

[27] W. Dong, K.-D. Kuhnert, "Robust adaptive control of nonholonomic mobile robot with parameter and nonparameter uncertainties," *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 261–266, 2005.

[28] R. R. Carmona, H. G. Sung, Y. S. Kim, H. A. Vazquez, "Stable pid control for mobile robots," in *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, IEEE, 2018, pp. 1891–1896.

[29] V. Mudeng, B. Hassanah, Y. T. K. Priyanto, O. Saputra, "Design and simulation of two-wheeled balancing mobile robot with pid controller," *International Journal of Sustainable Transportation Technology*, vol. 3, no. 1, pp. 12–19, 2020.

[30] A. Barsan, "Position control of a mobile robot through pid controller," *Acta Universitatis Cibiniensis. Technical Series*, vol. 71, no. 1, pp. 14–20, 2019.

[31] B. J. Saleh, A. T. Q. al-Aqbi, A. Y. F. Saedi, "A novel biogeography inspired trajectory-following controller for national instrument robot," in *International Conference on New Trends in Information and Communications Technology Applications*, Springer, 2018, pp. 171–189.

[32] N. H. Abbas *et al.*, "Design of a kinematic neural controller for mobile robots based on enhanced hybrid firefly-artificial bee colony algorithm," *Al-Khwarizmi Engineering Journal*, vol. 12, no. 1, pp. 45–60, 2016.

PUBLICATIONS FROM THE THESIS

Conference Papers

1. B. Y. Demir And Y. Eren, "MULTIPLE SCENARIO ANALYSIS IN PATH PLANNING OF AUTONOMOUS MOBILE ROBOTS: COMPARISON OF DIJKSTRA AND CONCURRENT DIJKSTRA ALGORITHMS," 11. Uluslararası Mühendislik ve Teknoloji Yönetimi Kongresi , İstanbul, Turkey, pp.71-78, 2024