Agoraphilic Navigation Algorithm in Dynamic Environment

Submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

by

Hasitha Sanjeewa Hewawasam

November, 2021

DECLARATION

I hereby declare that the thesis entitled "Agoraphilic Navigation Algorithm in dynamic environment" submitted by me, for the award of the degree of *Doctor of Philosophy* to Federation University is a record of bonafide work carried out by me under the supervision of Prof. M. Y. Ibrahim, Dr. G. Kahandawa, and Dr. T. Choudhury.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Date: 28/03/2022 **Haitha Hewawasam**

ABSTRACT

This thesis presents a novel Agoraphilic (free space attraction [FSA])-based navigation algorithm. This new algorithm is capable of undertaking local path planning for robot navigation in static and dynamic environments with the presence of a moving goal. The proposed algorithm eliminates the common weaknesses of the existing navigation approaches when operating in unknown dynamic environments while using the modified Agoraphilic concept.

The Agoraphilic Navigation Algorithm in Dynamic Environment (ANADE) presented in this thesis does not look for obstacles (problems) to avoid; rather, it looks for free space (solutions) to follow. Therefore, this algorithm is also a human-like optimistic navigation algorithm. The proposed algorithm creates a set of Free Space Forces (FSFs) based on the current and future growing free space around the robot. These Free Space Forces are focused towards the current and future locations of a moving goal and finally generate a single attractive force. This attractive force pulls the robot through current free space towards the future growing free space leading to the goal. The new free space concept allows the ANADE to overcome many common problems of navigation algorithms. Several versions of the ANADE have been developed throughout this research to overcome the main limitation of the original Agoraphilic algorithm and address the common weaknesses of the existing navigation approaches. The ANADE I uses an object tracking method to identify the states (locations) of moving objects accurately. The ANADE II uses a dynamic obstacle prediction methodology to identify the robot's future environments. In the ANADE III, a novel controller based on fuzzy logic was developed and combined with the new FSA concept to provide optimal navigational solutions at a low computational cost. In the ANADE III, the effectiveness of the ANADE II was further improved by incorporating the velocity vectors of the moving objects into decision-making. In the ANADE IV, a self-tuning system was successfully applied to the ANADE III to take advantage of the performances of free space attraction-based navigation algorithms.

The proposed final version of the algorithm (ANADE V) comprises nine main modules. These modules are repeatedly used to create the robot's driving force, which pulls

the robot towards the goal (moving or static). An obstacle tracking module is used to identify the time-varying free spaces by tracking the moving objects. Further, a tracking system is also used to track the moving goal. The capacity of the ANADE was strengthened further by obstacle and goal path prediction modules. Future location prediction allowed the algorithm to make decisions by considering future environments around the robot. This is further supported by a self-tuning, machine learning–based controller designed to efficiently account for the inherent high uncertainties in the robot's operational environment at a reduced computational cost.

Experimental and simulation-based tests were conducted under dynamic environments to validate the algorithm. Further, the ANADE was benchmarked against other recently developed navigation algorithms. Those tests were focused on the behaviour of the algorithm under challenging environments with moving and static obstacles and goals. Further, the test results demonstrate that the ANADE is successful in navigating robots under unknown, dynamically cluttered environments.

Keywords: *Agoraphilic*, *navigation*, *mobile robots*, *moving goal*, *dynamic environments*.

ACKNOWLEDGEMENT

This research work would not have been possible without the Australian Government Research Training Program (RTP) Fee-Offset Scholarship through Federation University of Australia. I am especially indebted to all my research supervisors, Prof. Yousef Ibrahim, Dr. Gayan Kahandawa and Dr. Tanveer Choudhury. Their immense knowledge and plentiful experience have encouraged me, providing with guidance and council I need to succeed in the PhD program.

Thank you, Prof. Yousef, for providing me extensive personal and professional guidance and teaching me a great deal about both scientific research and life in general.

My heartfelt thanks go to Dr. Gayan who has been supportive of my career goals and for his guidance, encouragement, and useful critiques of this research.

Further I am grateful to all of those with whom I have had the pleasure to work during this research.

Special thanks are due to my family. I would like to thank my wonderful parents, whose love and guidance are with me in whatever I pursue. They are the ultimate role models. Also, I wish to thank my loving and supportive wife, Hasini and my one and only sibling, Reshan, who had their constant faith in me.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF TERMS AND ABBREVIATIONS

ACO	Ant Colony Optimisation
AI	Artificial Intelligence
ANADE	Agoraphilic Navigation Algorithm in Dynamic Environment
ANN	Artificial Neural Network
APF	Artificial Potential Field
BFO	Bacterial Foraging Optimisation
BPF	Bacterial Potential Field
BPTT	Back Propagation Through Time
CGM	Current Global Map
DBT	Detection-Based Tracking
DFT	Detection-Free Tracking
DOPP	Dynamic Obstacle Position Prediction
DOT	Dynamic Obstacle Tracking
EKF	Extended Kalman Filter
FGM	Future Global Maps
FL	Fuzzy Logic
FLC	Fuzzy Logic Controller
FSA	Free Space Attraction
FSF	Free Space Forces
FSH	Free Space Histogram
FWDOA	Fuzzy-Wind Driven Optimization Algorithm
GA	Genetic Algorithm
GM	Global Map
GNRON	Goal Non-Reachable with Obstacles Nearby problem
IAPFM	Improved Artificial Potential Field Method
IDFC	Instantaneous Driving Force Component
KF	Kalman Filter
MBCGA	Matrix-Binary Codes-Based Genetic Algorithm

MBS	Motion Behaviour Selection
MCL	Monte Carlo Localisation
MGA	Matrix-binary codes based Genetic Algorithm
ML	Machine Learning
MO	Moving Obstacle
OAFLC	Obstacle Avoidance Fuzzy Logic Controller
PDF	Probability Density Function
PF	Particle Filter
PSO	Particle Swarm Optimisation
RL	Reinforcement Learning
RMC	Rvobot's Motion Commands
SDP	Sensory Data Processing
SI	Safety Index
SO	Static Obstacle
SS	Sector Status
SSGFLC	Sector States Generation Fuzzy Logic Controller
TFLC	Tracking Fuzzy Logic Controller
TSP	Travelling Salesman Problem
WDO	Wind Driven Optimisation

LIST OF PUBLICATIONS

1. H. S. Hewawasam, Y. Ibrahim, G. Kahandawa, and T. Choudhury, "Agoraphilic Navigation Algorithm under Dynamic Environment," IEEE/ASME Transactions on Mechatronics, pp. 1-1, 2021, doi: 10.1109/TMECH.2021.3085943.
2. H. S. Hewawasam, M. Y. Ibrahim, G. Kahandawa, and T. A. Choudhury, "Agoraphilic navigation algorithm in dynamic environment with obstacles motion tracking and prediction," Robotica, pp. 1-19, 2021, doi: 10.1017/S0263574721000588.
3. H. S. Hewawasam, M. Y. Ibrahim, G. Kahandawa, and T. Choudhury, "Comparative study on object tracking algorithms for mobile robot navigation inGPS-denied environment," in Proc. IEEE Int.Conf. Ind. Technol., 740 2019, pp. 19–26.
4. H. S. Hewawasam, M. Y. Ibrahim, G. Kahandawa, and T. A. Choudhury, "Development and bench-marking of Agoraphilic navigation algorithm in dynamic environment," in Proc. IEEE 28th Int. Symp. Ind. Electron., 2019, pp. 1156–1161.
5. H. S. Hewawasam, M. Y. Ibrahim, G. Kahandawa, and T. Choudhury, "Agoraphilic navigation algorithmin dynamic environment with and without prediction ofmoving objects location," in Proc. 45th Annu. Conf. IEEE Ind. Electron. Soc., 2019, vol. 1, pp. 5179–5185.
6. H. S. Hewawasam, M. Y. Ibrahim, G. Kahandawa, and T. A. Choudhury, "Evaluating the performances of the Agoraphilic navigation algorithm under dead-lock situations," in Proc. IEEE 29th Int. Symp. Ind. Electron., 736 2020, pp. 536–542.
7. H. S. Hewawasam, Y. Ibrahim, G. Kahandawa, and T. Choudhury, "The Agoraphilic Navigation Algorithm under Dynamic Environment with a Moving Goal," in Proc. IEEE 30th Int. Symp. Ind. Electron., 2021.
8. H. S. Hewawasam, Y. Ibrahim, and G. Kahandawa, "Past, Present and Future

of Path-Planning Algorithms for Mobile Robot Navigation in Dynamic Environments," IEEE Open Journal of the Industrial Electronics Society (submitted-under review).

9. H. S. Hewawasam, Y. Ibrahim, and G. Kahandawa, "Self-Tuning Agoraphilic Algorithm for Robots Navigation in Dynamic Environmentsl," Journal of Intelligent and Robotic Systems, Elsevier (submitted).

10. H. S. Hewawasam, Y. Ibrahim, and G. Kahandawa, "Machine Learning-based Agoraphilic Navigation Algorithm for use in Dynamic Environments with a Moving Goal," IEEE/ASME Transactions on Mechatronics (submitted-under review).

11. H. S. Hewawasam, Y. Ibrahim, and G. Kahandawa, "New Fuzzy Logic-Based Agoraphilic Navigation Algorithm in Dynamic Environments," Mechatronics, Elsevier (submitted).

1

Chapter 1: Introduction

1.1 Background and Significance

Robotics plays a major role in the contemporary world. Compared to fixed robots, mobility promises to be the next frontier in robotics. Mobile robots promise additional capacity to end users in new advanced applications. These applications include mining [1], space [2], surveillance [3], military applications[4] , hospital work [5], agriculture[6] and many others [7]. Navigation is an important element of mobile robots' functionality. Unlike driverless cars' operating conditions, in most circumstances, the mobile robot's operational environment is not specifically engineered for the robot. Therefore, from a navigational perspective, the mobile robot must be able to operate in an unpredictable environment. Further, this environment will, in most cases, be dynamic. This environmental complexity requires the robot to make navigational decisions in real time. Therefore, an efficient, autonomous navigation system is characterised by an ability to navigate robots in unknown dynamic environments using real-time decision-making algorithms.

The literature describes a number of fundamental navigational methods, including the roadmap approach, cell decomposition, mathematical programming and the Artificial Potential Field (APF) method. The roadmap approach is also known as the Restric-

tion, Skeleton and Highway approach. In this approach, the free C-space (i.e., the set of feasible motions) is retracted, reduced to, or mapped onto a network of one-dimensional lines. In the roadmap approach, the solution for the navigation problem converts to a graph searching problem by limiting the solution to a network. The Silhouette method [8], Subgoal Network method [9], Voronoi diagram approach [10] and visibility graph approach [11] are some of the commonly used roadmaps.

In cell decomposition methods, the free C-space is decomposed into a set of simple cells and the adjacency relationship among the cells are computed. In this approach, cells that belong to the starting point and the end point are connected through a sequence of connected cells. The concept of cell decomposition was used in [12].

The mathematical programming approach represents the requirement of obstacle avoidance with a set of inequalities on the configuration parameters. Mathematical programming is formulated as a mathematical optimisation problem that finds a curve between the start and goal configurations, minimising a certain scalar quantity.

In APF-based algorithms, the robot is modelled as a particle in the space, while the goal and obstacles are modelled as charged particles. The goal creates an attractive force on the robot, and obstacles create repulsive forces. The robot is pushed and pulled by these attractive and repulsive forces until it reaches the goal [13]. The basic concepts of these methods are used in many novel navigation algorithms.

Among these methods, the APF-based method has been used widely because of its simplicity and adaptability. However, there are some well-documented problems inherent in APF methods [14] such as,

1. Trap situations or Dead-locks (local minima)

2. No passage between closely spaced obstacles

3. Oscillations in the presence of obstacles

4. Oscillations in narrow passages

5. Goal Non-Reachable with Obstacles Nearby problem (GNRON)

6. In dynamic environments traditional approaches fail to achieve the navigation task.

Researchers have tried to overcome this problem in many ways. Harmonic function-based methods are one of the methods used for generating local minima free potential functions. Kim et al. [15] developed a method based on a panel method to represent obstacles with different shapes to derive potential fields. In this approach, harmonic functions are used to eliminate local minima problems. Some other approaches that use harmonic functions to generate local minima free potential fields are discussed in [16–18]. Another method to overcome deadlock situations is introducing a virtual obstacle concept. Iraji et al. [19] proposed a methodology that uses virtual obstacles in deadlock situations to create an extra repulsive force at the trapping point to push the robot away from the local minima. In this approach, certain criteria have been introduced to identify situations in which virtual obstacles should be used. Park et al. [20] have also proposed a similar methodology using virtual obstacle methodology to avoid traps. Li et al. [21] also attempted to address the local minima issue by redefining the potential function by introducing virtual targets. Another novel approach to avoid deadlock situations is discussed in [22]. Authors have proposed an obstacle avoidance method based on APF and gravity chain.

As mentioned in [23], APF-based algorithms suffer from the inability to reach the goal when there is a big object close to it. In this situation, the robot receives a repulsive

force from the big obstacle that is higher than the attractive force created by the goal. Therefore, the robot fails to reach its destination. Ge and Cui [23] proposed a methodology to overcome the GNRON problem by using several tuning parameters to shape the potential function.

Ran et al. [24, 25] have also tried to address some of the problems inherited with the traditional APF algorithm. The method used in [24] has reduced the oscillations in the presence of obstacles by using an advanced calculation methodology. Newton's method was used in this approach for calculating the gradient of the potential field, and the gradient calculation was used to smooth the trajectory. However, this method drains computational resources [26].

Sfeir [26] implemented a framework to address the GNRON problem and reduce the oscillations in the presence of obstacles. In this approach, the original potential fields of the obstacles are modified using a tuning parameter. Further, this method is capable of navigating robots in unknown environments while reducing some problems of APF algorithms. However, there is no proper way to find the tuning parameter. This is the main disadvantage of this framework.

Montiel et al. [27] an algorithm called Bacterial Potential Field (BPF). This uses the APF method with a Bacterial Evolutionary Algorithm to obtain a better path planning algorithm and to reduce the drawbacks of the traditional APF method. The proposed technique is capable of navigating robots in environments with static obstacles (SOs) and moving obstacles (MOs). Further, the BPF algorithm uses the start, goal and obstacle positions as features to obtain a sequence of objective points that the mobile robot must attain. A sequence of objective points is transformed into a path using the gradient information influenced by attractive and repulsive forces.

Siming et al. [28] also developed an APF-based algorithm that is suitable to use in dynamic environments. This algorithm employs a methodology to avoid the GNRON problem. The algorithm uses a virtual target to modify the repulsive fields at local extreme points. A special subroutine is used when the robot is at a local extreme point. In this special subroutine, if the robot, obstacle and goal are in a straight line, a virtual target is introduced. In [29], the problem of navigating robots in dynamic environments is addressed. The proposed solution is based on developing a Fuzzy Logic (FL) APF filling an autonomous robot's workspace by combining the attractive and repulsive forces acting on the robot from two FL expert subsystems. This APF is conducted by initially sensing the surrounding dynamic environment by the robot and localisation.

Kovacs et al. [30] presented a concept for the path planning of mobile robots in household environments. The APF algorithm has been extended by motion characteristics of household animals. This algorithm also attempts to extend the APF algorithm to unknown dynamic environments. In this algorithm, the repulsive potential function created by obstacles is not monotonic. Therefore, this method allows the obstacles to attract the robot in certain cases depending on the distance and parameter range. However, this algorithm assumes that the position and velocity of moving objects can be accurately measured.

To date, there have been many published attempts at alleviating some or all of the aforementioned limitations. All the new methodologies tried to address the limitations of the traditional APF algorithm while keeping its inherent conceptual problem. In other words, the published work in this field has tried to treat the bad outcomes of the APF algorithm instead of fixing its actual conceptual problem. Therefore, most of the new APF-based algorithms have lost some of the main advantages of the classical APF

approach, such as simplicity and less use of computational resources.

However, a novel fundamental method has been introduced by Ibrahim and McFetridge [31], called 'Agoraphilic algorithm', to resolve the conceptual problem of the traditional APF method. The Agoraphilic algorithm introduced a new free space attraction (FSA) concept that allows the algorithm to use just one attractive force to drive the robot to the goal.

As discussed above in APF methods, the robot is navigated to the goal by the vector summation of attractive and repulsive forces created by the goal and obstacles. Under the static environment, the robot can easily get into a position where the attractive force is equal to the repulsive forces acting on the robot. This makes the resultant force on the robot zero. Consequently, the robot gets stuck in these local minima and fail to reach its goal. In contrast, the Agoraphilic navigation algorithm creates only one attractive force on the robot. This force is developed from the free-space concept. Consequently, the robot moves towards the goal as long as there is free-space around it. As a result of this original Agoraphilic algorithm avoid the dead lock situation with its fundamental free space attraction concept. Furthermore, as discussed in [32] original Agoraphilic algorithm can successfully navigate in between closely spaced obstacles without oscillations because of free space attraction concept. Unlike in APF method, in free space attraction method robot is not pushed by the nearby obstacles. As a result, as long as there is enough free space in between closely space obstacles robot is pulled through the passage between the two obstacles without any oscillation, Fig. 1.1. Also the experimental test results presented in [33] show the ability of the original Agoraphilic algorithm to avoid obstacles without performing oscillations in a cluttered area (see Fig. 1.2). In APF-based methods, if the robot is moved through a narrow passage, repulsive forces

are created on the two sides of the robot. These repulsive forces create oscillations. In contrast, as discussed above, in the original Agoraphilic algorithm robot is only pulled by an attractive force created by the free space. Therefore, the Agoraphilic algorithm avoids oscillation in a narrow passage. The corresponding experimental test results are shown in Fig. 1.3. Another advantage of the original Agoraphilic algorithm is, that it has addressed the GNRON problem. Although there is a large obstacle behind the goal the robot is still pulled by the free space between the robot and the goal. Therefore, the robot is capable to reach the goal. Experimental test results shown in Fig.1.1 is an example for this case.

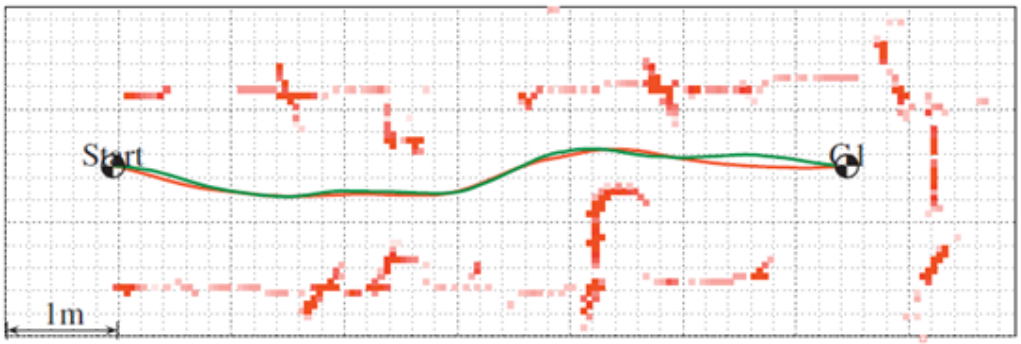


Fig. 1.1 This experimental test results demonstrates the original Agoraphilic algorithm's ability to navigate between closely spaced obstacles (walls) without oscillations and ability to address the GNRON problem [32].
Note. The two paths shown in this figure are for different tuning setups of the original Agoraphilic algorithm

As mentioned, the original Agoraphilic algorithm addressed the principal issues of the APF-based method by introducing the new FSA (Agoraphilic) concept. This concept is a fundamentally new approach to mobile robot navigation.

However, the main problem with the original Agoraphilic algorithm was that it was restricted to being used in static environments. This presents significant problems to most real-world applications for mobile robots, as they frequently operate in unknown dynamic environments. This limitation of the original Agoraphilic algorithm provided

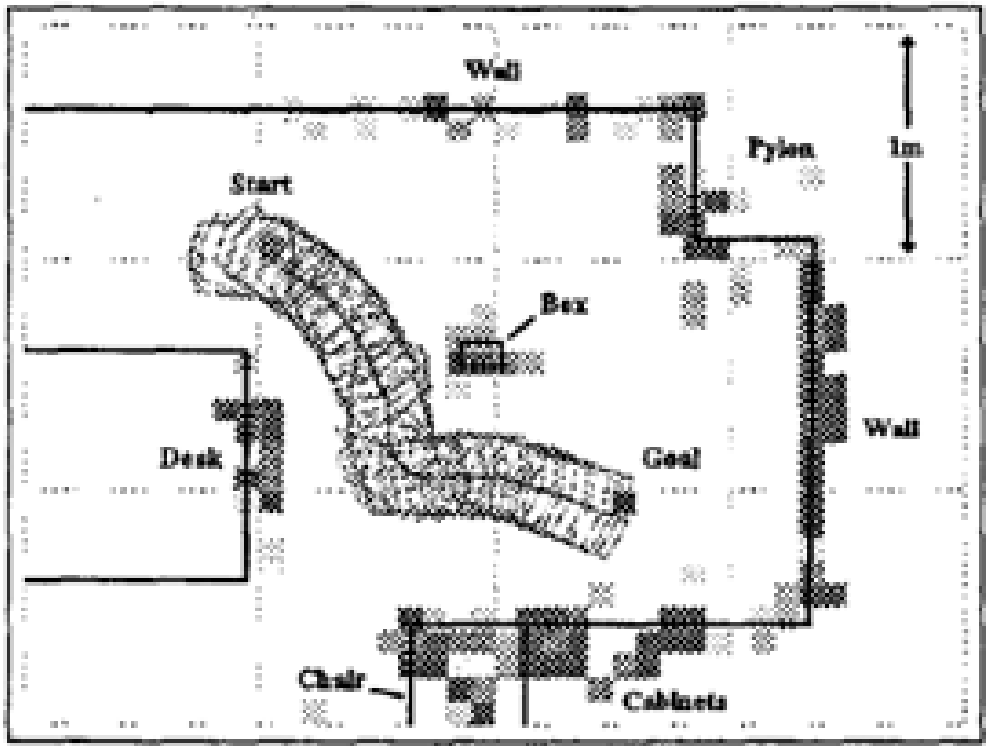


Fig. 1.2 This experimental result demonstrates the original Agoraphilic algorithm's ability to navigate in a cluttered environment [33].

the primary motivation for this research: the ANADE.

Dynamic environments are characterised by high levels of uncertainty, thereby presenting significant challenges to navigation algorithms [35]. Navigation techniques based on artificial intelligence (AI) are frequently used to address this problem. Among these, FL is commonly used to overcome uncertainties at a low computational cost. Most alternative navigation algorithms fail to use the velocities of MOs as an input for decision-making (future discussed in Chapter 2). This is due to the difficulty of estimating the velocity vectors of MOs [36]. Consequently, most existing navigation algorithms are unable to navigate robots in uncertain dynamic environments, which leads to a suboptimal performance or system failure in complex situations [36].

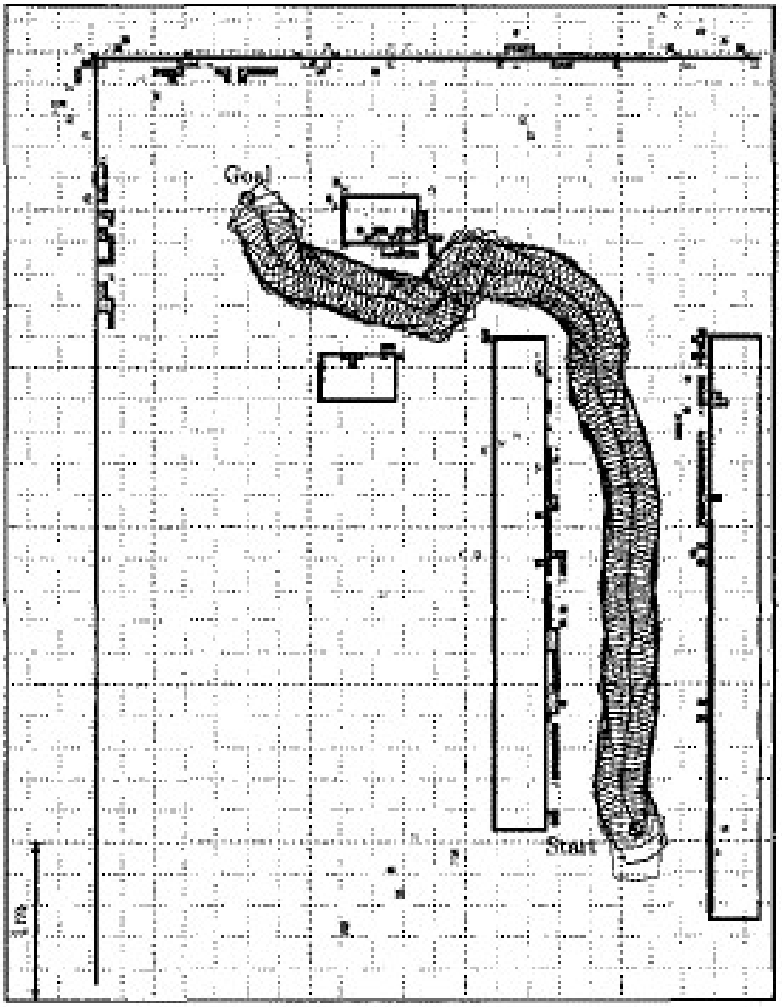


Fig. 1.3 This experimental result demonstrates the original Agoraphilic algorithm's ability to avoid oscillation in a narrow passage [34].

1.2 Expected Outcomes and Contribution of the Research

This research introduces a novel navigation method based on the FSA concept proposed in the original Agoraphilic navigation algorithm. This new navigation algorithm is capable of navigating robots not only in static environments but also in unknown dynamic environments with moving targets. The principal contributions that originate from this research are as follows.

1. Agoraphilic Navigation Algorithm in Dynamic Environment with Tracking

(ANADE I)

The Agoraphilic Navigation Algorithm in Dynamic Environment with tracking (ANADE I) that was developed in this research was proven to navigate mobile robots in simple dynamic environments. The ANADE I is an FSF-based optimistic algorithm. It uses only attractive forces created by current free space in the robot's surrounding environment. The ANADE I uses an object tracking method to accurately identify the states (location) of moving objects. This algorithm drives the robot through current free spaces leading to the goal. This proposed methodology was able to overcome the main limitation of the original Agoraphilic algorithm, which is limited only to static environments, while keeping its advantages in static environments.

2. Agoraphilic Navigation Algorithm in Dynamic Environment with Tracking and prediction (ANADE II)

In dynamically cluttered environments, having a short-term path prediction of unknown MOs helps the algorithm make smart decisions. The new Agoraphilic Navigation Algorithm in Dynamic Environment with tracking and prediction (ANADE II) was developed to improve ANADE I's performances by incorporating prediction. The ANADE II is also a local path planner to navigate mobile robots in unknown static and dynamic environments. The new algorithm takes an optimistic approach to solving navigation problems. It does not look for obstacles to avoid but for space solutions to follow. Compared to the ANADE I, the ANADE II uses a dynamic obstacle prediction methodology to identify the robot's future environments. The ANADE II pulls the robot through the future growing free space passages leading to the goal, whereas the ANADE I only considered the current free space. Further, the ANADE II proposes a novel architecture for the ANADEs that can also be adapted by other virtual

force–based or field-based navigation algorithms to improve their performances. The ANADE II successfully improved the performances of the ANADE I in dynamically cluttered environments. The optimistic approach of the new algorithm combined with future dynamic objects prediction makes the ANADE II superior to the ANADE I.

3. Artificial Intelligence (AI)- based Agoraphilic Navigation Algorithm in Dynamic Environment (ANADE III)

Unknown dynamic environments are characterised by high levels of uncertainty. AI-based methods are frequently used to address this problem. A novel controller based on FL was developed and combined with the new FSA concept to provide optimal navigational solutions for this at a low computational cost. Further, the effectiveness of the ANADE II was further improved upon by incorporating the velocity vectors of MOs in decision-making.

4. Self-tuning Artificial Intelligent based Agoraphilic Navigation Algorithm in Dynamic Environment (ANADE IV)

In mobile robot navigation, the robot faces new and different circumstances regularly. Therefore, if the navigation algorithm can identify new circumstances and change the decision-making behaviour, the robot can adapt to new situations successfully. This adaptation increases the safety and efficiency of the navigation process. Therefore, to further improve the performance of the algorithm, a new self-tuning Fuzzy Logic Controller (FLC) was developed. The Self-Tuning Artificial Intelligence–Based Agoraphilic Navigation Algorithm in Dynamic Environment (ANADE IV) has maximised decision-making flexibility.

5. Agoraphilic Navigation Algorithm in Dynamic Environment with a moving goal (ANADE V)

Only a few navigation algorithms can track and hunt a moving goal in an unknown dynamic environment. However, there are practical applications when the robot has to follow or hunt a moving goal. Therefore, in this research, the ANADE was further improved to address this issue. In this algorithm, new goal tracking, goal path prediction modules and a machine learning (ML)–based controller are used. These new modules are combined with other modules used in the ANADE IV to give the Agoraphilic Navigation Algorithm in Dynamic Environment with a Moving Goal (ANADE V) the capability to track and hunt a moving target in a dynamically cluttered environment while keeping all the advantages of the ANADE V. Further, this novel algorithm introduces the use of the FSA concept in hunting moving targets in both static and dynamic environments.

1.3 Outline of the Thesis

The remainder of the thesis is divided into eight chapters. These chapters describe the general background of the study, the continuous development of the proposed novel algorithm, experimental validation of each stage of development and a detailed conclusion.

Chapter 2 presents the general overview of navigation techniques capable of navigating robots in dynamic environments. This literature review also shows the common weakness and research gaps in the field.

Chapter 3 discusses about moving obstacle tracking. Tracking is one of the main components identified as needed in the process of developing an Agoraphilic (FSA)-based navigation method capable of navigating a robot in unknown dynamic environments. Different types of tracking methods are discussed in this chapter. Also, a suitable

tracking method to mix with the FSA concept was identified.

Chapter 4 discusses the ANADE I, which is the algorithm that introduces the fundamental building blocks for Agoraphilic-based navigation algorithms capable of navigating robots in static and dynamic environments.

Chapter 5 introduces a new architecture with a prediction method. The ANADE II was developed to further improve the performances of the ANADE I. This chapter shows the step-by-step development of the ANADE II. Also, an experimental validation of the ANADE II and a detailed comparison between the ANADE II and the ANADE I is provided.

Chapter 6 discusses the Artificial Intelligence–Based Agoraphilic Navigation Algorithm in Dynamic Environment (ANADE III), which is the novel AI-based version of the ANADEs. This chapter gives an overview of both the development of the ANADE III and what improvements were made. This chapter also shows the integration of an AI-based technique for force shaping, which improved the performance of the algorithm. This chapter also validates the new algorithm and its improvements via experimental results.

Chapter 7 discusses the ANADE IV, which is an improved version of the ANADE III. This chapter gives detailed information about the different types of behaviours needed by any navigation algorithm to be successful in local path planning. This chapter also demonstrates how the force-shaping module can be altered to create different types of decision-making behaviours. The approach proposed in this chapter allows the ANADE IV to self-tune the algorithm according to the robot's current circumstances. This chapter also validates the algorithm and its improvements via experimental results.

Chapter 8 discusses the ANADE V, which is the first Agoraphilic-based naviga-

tion algorithm capable of hunting a moving target in a static or dynamically cluttered environment. This chapter shows how we can use the basic FSA concept with other developed modules to track and hunt a moving target. Further, simulation and experimental test results are provided in this chapter to validate the algorithm.

Chapter 9 presents a detailed summary of the research outcomes.

2

Chapter 2: Literature Review

2.1 Background and Significances

Mobile robots play an important role in many sectors in the contemporary world. They are used in many applications to improve efficiency, increase accuracy and reduce any risk involved for humans. Most of the time, mobile robots' environment is unknown and dynamic. Therefore, mobile robot navigation is a complicated task. This has sparked the interest of many researchers in developing methodologies to address the mobile robot navigation problem. The primary task in navigation is either to reach a predetermined goal or to follow a predetermined path without any collisions.

Autonomous navigation is subdivided into four main subtasks [37] (see Fig. 2.1):

1. The robot collects information from its environment using a sensory system (perception)..
2. The robot identifies its location in the environment (localisation).
3. The robot decides how to manoeuvre to reach the goal without collision (path planning).
4. The robot controls its motions to follow the desired path.

Among the above four tasks, path planning is one of the most important areas that are the focus of this research. This chapter discusses the different up-to-date types of robot path planning methods used for navigating mobile robots in dynamic environments.

In general, the path planning methodologies are classified into two main groups: classical approaches and heuristic approaches. In the literature, the roadmap approach, cell decomposition, mathematical programming and the APF method can be identified as frequently used classical path planning methods (see Fig. 2.2).

The classical methods are simple, have low computational cost and are easy to implement. Therefore, they are preferred in many real-time path planning applications and may produce good results, especially in static environments [38].

However, most of the classical methods, except the APF method, failed to handle the high uncertainties in dynamic environments. Therefore, the APF method is the only classical method that became famous among researchers developing navigation algorithms suitable for dynamic environments [39].

Conversely, heuristic methods such as a genetic algorithm (GA), FL, Artificial Neural Networks (ANNs), Reinforcement Learning (RL), Particle Swarm Optimisation (PSO), Bacterial Foraging Optimisation (BFO) and Ant Colony Optimisation (ACO) are becoming famous among the researchers in the field (see Fig. 2.2).

In this chapter, the major aspects of the mobile robot navigation methods in dynamic environments are analysed (study was conducted using 62 papers with 75% papers in last ten years), and classifications of the diverse approaches that identify recent trends, weaknesses and active areas are provided.

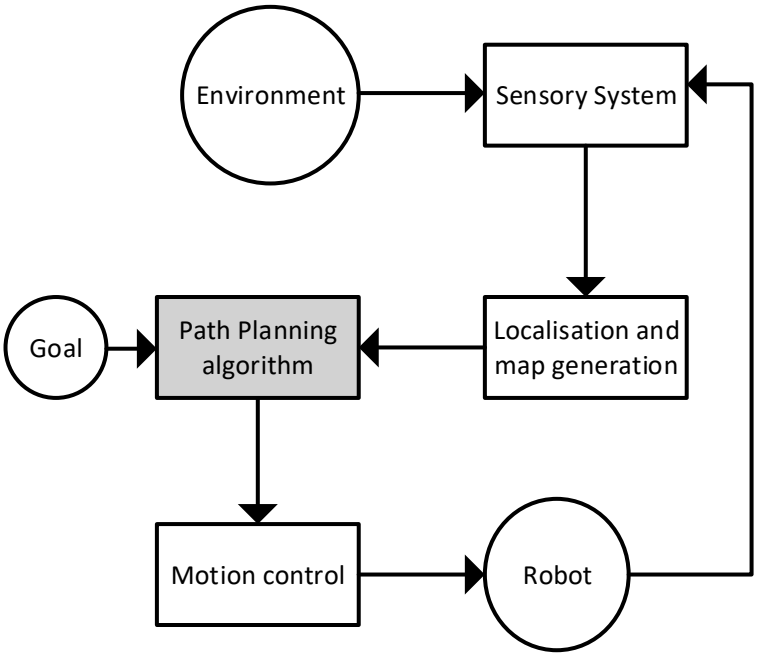


Fig. 2.1 Basic Steps Needed for Mobile Robot Navigation

2.2 Popular Navigation Techniques Capable of Navigating Robots in Dynamic Environments

2.2.1 Artificial Potential Field (APF)

In the APF model, the robot is considered a point mass while the goal and obstacles are modelled as force fields. The goal creates an attractive force, and the obstacles create repulsive forces on the robot. Those forces push and pull the robot towards the goal while avoiding obstacles. The direction and the magnitude of the resultant force vector denote the direction of the robot's velocity vector and magnitude, respectively. The

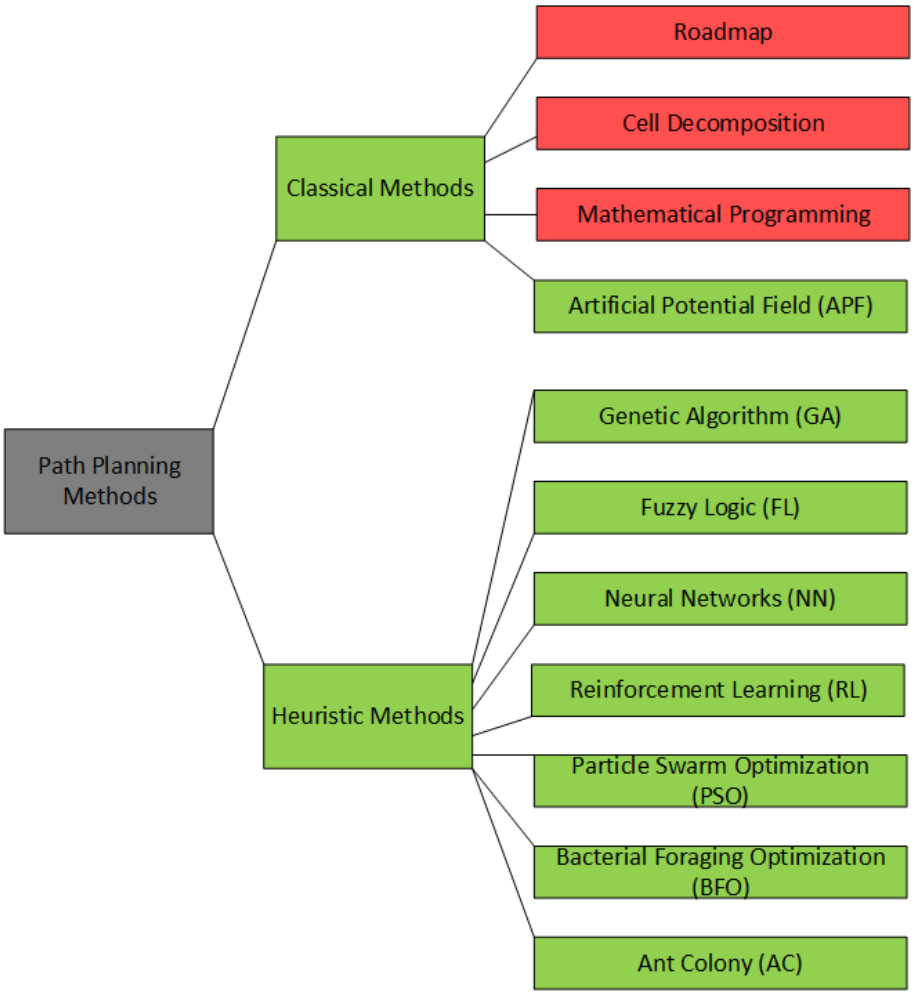


Fig. 2.2 Basic Path Planning Methods.

original research focused only on SOs.

However, the concept was also extended to dynamic environments. In a dynamic environment, the algorithm assumes that there is only one obstacle close to the robot. The possibility of colliding with the obstacle is calculated by a function of the distance between the robot and the obstacle and the relative velocity of the obstacle with respect to the robot [40]. When the distance between the robot and the obstacle increases and the relative velocity decreases, the avoidability measure increases. Nak et al. used a virtual distance function as the avoidability measure, which emphasises the distance metric over the speed. The algorithm can make the robot avoid obstacles that are closer

or further away by tuning this function. A potential force is generated accordingly to match with the original potential field method. Numerous studies have used this concept to navigate robots in dynamic environments. Further, some studies have used this concept when the goal is moving.

2.2.2 Genetic Algorithm

GA is an optimisation methodology that is commonly used to generate solutions for combined optimisation and search problems. This method follows the basic principles of genetic and natural selection. Applications of GA were mainly focused on the field of computer science. However, GA-based methods are also used in the field of mobile robot navigation. The GA starts without any knowledge of the correct solution and depends completely on the responses of the environment and the evolutionary operators to arrive at the best solution [41] (see Fig. 2.3).

2.2.3 Fuzzy Logic

FLCs are either a rule- or knowledge-based system. Those FLC systems comprise a set of fuzzy rules. These fuzzy rules are generated based on the domain knowledge of human experts. The simplicity of rule-based systems, the low computational cost and the capability to perform a wide variety of tasks make FL-based methods quite popular among researchers [42]. FL is a very versatile AI-based technique for mobile robot navigation, with the ability to represent the fuzzy rules using linguistic terms. Further, FL systems are identified as reliable decision-making systems under high uncertainty and with incomplete information.

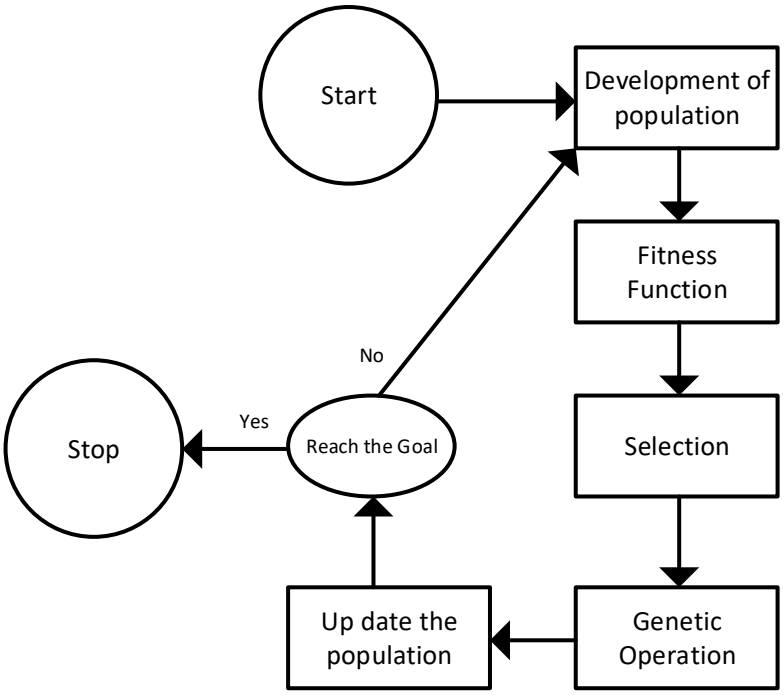


Fig. 2.3 Process Flowchart of a Basic Gnetic Algorithm.

2.2.4 Artificial Neural Networks (ANN)

The ANN techniques were initially used for computer science–based work, such as image processing and pattern recognition. ANNs are inspired by the mechanism of the human brain. ANNs are complicated networks of artificial neurons, commonly known as nodes. These networks are capable of solving AI-based problems. ANNs can behaviour handle high levels of uncertainties that exist in dynamic environments. There have been numerous attempts at developing ANN-based navigation algorithms capable of navigating robots in dynamic environments.

2.2.5 Reinforcement Learning

RL is an unsupervised ML algorithm that tries to maximise a reward signal. In RL, an agent is taught how to map situations to actions. The agent is not directly told which action to perform but instead is told which action can offer the highest reward. In challenging situations, actions may affect not only the next reward but also all the subsequent rewards throughout the process. In RL, trial and error and delayed rewards are the main distinguishing features [43]. There are several different learning algorithms that fall into an RL category, such as Q-learning, deep RL and Double Q-learning. However, all these methods have a common goal of teaching the agent to reach a target by interacting with the environment [44].

2.2.6 Particle Swarm Optimisation

PSO is a population-based optimisation technique that adopts the motion of schooling fish and bird flocks. The PSO technique was introduced in 1995 by Eberhart et al. [45]. It has some similarities to other evolutionary computation techniques. The process starts with a population of random solutions and seeks the optimal solution. In PSO, potential solutions are known as particles. In the process, a set of current optimum particles is identified. The other particles follow the current optimal particles. PSO's high efficiency in terms of speed and memory requirements inspired most of the researchers to adapt this technique to use for mobile robot navigation in dynamic environments. The decision-making process of the algorithm is shown in Fig. 2.4.

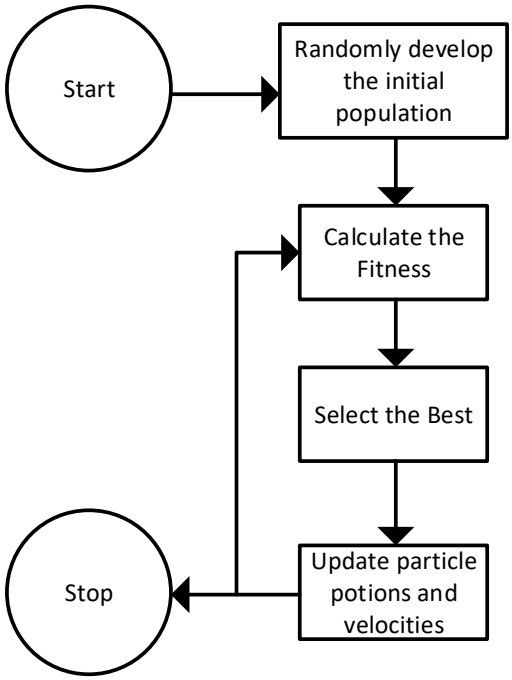


Fig. 2.4 Steps of Particle Swarm Optimisation

2.2.7 Bacterial Foraging Optimisation

The BFO algorithm was introduced by Passino in 2002 [46]. This optimisation algorithm adapts the behaviour of Escherichia coli bacteria. E. coli is a cell body with eight to 10 rotating flagella attached to the body randomly. E. coli bacteria move in their surrounding environment by performing two main actions: 'run' and 'tumble'. The bacteria run or tumble by rotating their flagella in an anticlockwise or clockwise direction. In addition to motion, E. coli bacteria propagate to renew their colony and eliminate the weaker or older individuals. There are three main parts of the behaviour of the bacteria:

1. Chemotaxis: Bacteria use flagellation to move.

2. Reproduction: Reproduction is a way of renewing the colony, as well as removing

the inefficient individuals. The efficient individuals are used for reproduction. Reproduction is also an optimisation behaviour of BFO.

3. *Elimination and dispersal:* One major variation in the environment is the reduction of food concentration. At certain reproduction rounds in the bacterial lifecycle, the diffusion processes take place [47].

2.2.8 Ant Colony Optimisation

ACO is used to solve the combinatorial optimisation problem. The population-based ACO algorithm adopts the behaviour of ants. Ants are skilled to discover the shortest path to their food source from their nest [48]. The ACO algorithm finds the solutions to the Travelling Salesman Problem (TSP) using agents called ants. Ants use a special pheromone, which they deposit on the edge of the TSP graph to communicate with others [49]. This optimisation algorithm is also used as the base algorithm to develop navigation algorithms capable of navigating robots in dynamic environments.

2.3 Discussion

Based on the methodologies presented in this study, it was apparent that navigation algorithms can be classified as classical and heuristic approaches. A number of classical approaches are discussed in this chapter, including APF, cell decomposition, the roadmap approach and mathematical programming. Those methods were popular in the early days in developing navigation algorithms. The classical methods are simple, have a low computational cost and are easy to implement. Therefore, they were chosen for many real-time path planning applications. Also, classical methods have obtained good results, especially in static environments. However, most of these methods failed

to handle the high uncertainties in dynamic environments. Thus, modified APF methods are still used as the base method in many new navigation algorithms for navigating robots under dynamic environments (see Fig. 2.5).

Conversely, heuristic methods are considered more intelligent and effective compared to classical methods as they can adapt to the uncertainty of constantly changing environments. Consequently, heuristic-based methods are used in most navigation algorithms for robots in dynamic environments. However, most of the heuristic approaches need a learning phase with high computational cost (see Table 2.1).

Table 2.2 provides a detailed analysis of the algorithms used to date for navigating robots in dynamic environments. The analysis was based on some key parameters, such as the basic concept used, the ability to navigate through multiple moving objects, the ability to track a moving goal and the ability to track and estimate the states of moving objects. The analysis also considered the algorithms that considered the velocities of moving objects in decision-making and the ability of the algorithm to predict the future environments of the robot and included validation of the algorithm via simulation and experiments. This analysis showed the number of classical methods suitable for dynamic environments is extremely low compared to the heuristic methods that are suitable (see Fig. 2.6). According to this study, APF is the only classical method used for developing navigation algorithms for dynamic environments. Among the considered algorithms, 19% use FL, 7% use GA and 15% use the APF algorithm as the base algorithm (see Fig. 2.5). It was also found that there has also been a considerable number (12%) of successful navigation algorithms developed using other methods.

It is important to take the velocity of the moving objects into account in the decision-making for navigation in dynamic environments [50]. However, there are only a few

algorithms that incorporate the velocity information for decision-making (see Fig. 2.7). This is a major drawback of most of the algorithms. Most of the existing sensory systems do not give velocity information; they only give distance information. This is one of the main reasons for not incorporating velocity information in navigation. Some algorithms have estimated the velocities by simply finding the rate of change of displacement based only on distance information. In some studies, it has been argued that noisy sensory information can give inaccurate velocity estimations. A feasible solution for this issue would be fusing a low computational cost tracking system to estimate the velocities of moving objects with the basic navigation algorithms.

Only a few algorithms use a prediction system in unknown dynamic environments (see Fig. 2.7). A short-term prediction system could improve the efficiency of navigation algorithms. Further, a prediction system can help the robot navigate in critical conditions with multiple MOs challenging the robot at the same time [51]. However, there are few algorithms that can perform the navigation task successfully when there is a moving goal (see Fig. 2.8).

Almost all the algorithms have been verified using simulation tests (see Fig. 2.10). However, only 35% of the navigation algorithms surveyed have been validated via real experiments (see Fig. 2.10). Moreover, very few algorithms have been tested for their behaviour when multiple moving objects challenge the robot at the same time and for their decision-making when the robot is heading to a trap.

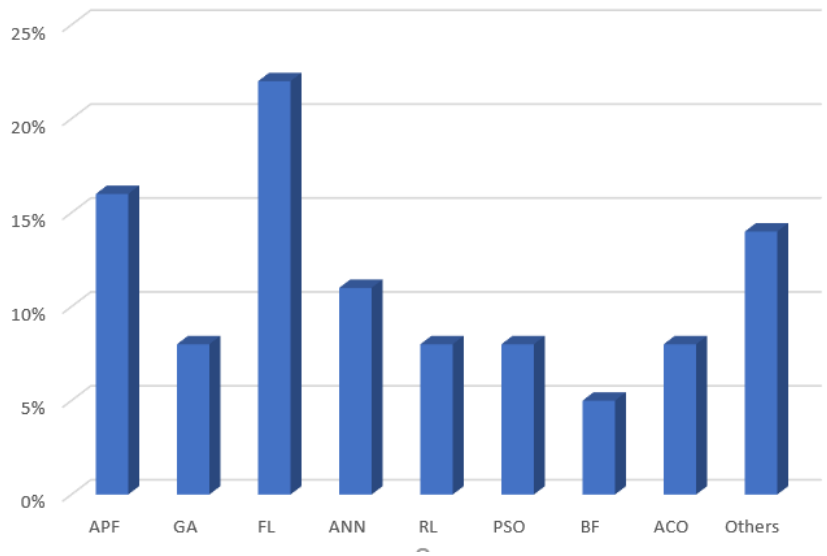


Fig. 2.5 Navigation Algorithms Capable of Navigating Robots in Dynamic Environments

Tab. 2.1 General Advantages and Disadvantages of Base Algorithms

Base-algorithm	Advantages	Disadvantages
Artificial Potential Field	1. Less computational cost 2. Simple to implement 3. High adaptability	1. Tedious parameter tuning 2. Local minima problem 3. Oscillations 4. Inability to reach a goal when a large goal is nearby
Genetic algorithm	1. Generate high quality accurate solutions	1. Can develop oscillations 2. Hard to implement in dynamic environments
Artificial Neural Networks	1. Nonlinear mapping capability	1. Needs to go through a huge learning process

Base-Algorithm	Advantages	Disadvantages
	2. Capable of parallel processing 3. Learning ability 4. System can be retrained when the conditions are changed	2. Computational cost is high Impossible to come up with a transfer function
Fuzzy Logic	1. Capable of representing human thinking 2. Can mimic the actions of a manual robot operator. 3. Fuzzy rules are transparent and clear. (as using the linguistic variables) 4. Can handle high uncertainties	1. Selecting the right rules and membership functions are critical
Reinforcement Learning	1. Learn by collecting experiences	1. High computational cost 2. Needs some initial training.
Particle Swarm Optimisation	1. Less computational complexity	1. Possibilities of getting tarped in local minimums in complex environments

Base-Algorithm	Advantages	Disadvantages
	2. High efficiency in terms of speed and memory requirements	2. Less accurate and practical than genetic algorithm
Bacterial Foraging Optimisation	1. Ability to solve the multi-difficulty scheduling problem effectively	1. High computational cost minimums in complex environments
Ant Colony Optimisation	1. Requires fewer control parameters	1. Slow convergence

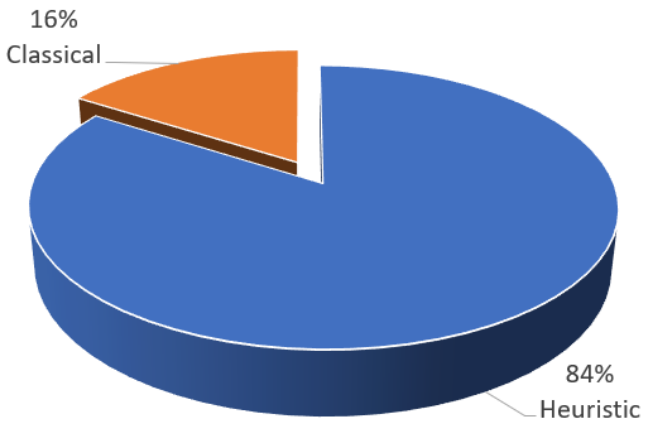


Fig. 2.6 A Comparison of Classical and Heuristic Approaches Based on Published Papers

Tab. 2.2 Detailed Analysis of Published Algorithms Based on Identified Key Parameters.

Base methodolog	APF	GA	FL	ANN	RL	PSO	BF	ACO	Others
Algorithms that can navigate with multiple MOs	[39], [52], [53], [54], [55], [56]	[57], [58], [59]	[60], [61], [62], [55], [63], [64], [65], [66]	[67], [68], [69], [70]	[71], [72], [73]	[60], [74], [75]	[76], [56]	[77], [61], [78]	[50], [79], [80], [81], [82]

Algorithms capable of hunting a moving goal	[54], [55]			[68], [70]					[50], [80], [81]
Algorithms that use a tracking method to track MOs	[39], [53]			[67]					[50], [80]
Algorithms that use a prediction method to predict the motion of MOs	[53]			[69]					[50], [79], [80], [82]
Algorithms that use velocity/ relative velocities of MOs in decision-making	[39], [53], [54], [55]		[61]	[67], [69], [70]	[71], [73]				[50], [79], [80], [82]

Algorithms validated by simulation tests	[39], [52], [53], [54], [55], [56]	[57], [58], [59]	[60], [61], [62], [55], [63], [64], [65], [66]	[67], [68], [69], [70]	[71], [72]	[60], [74], [75]	[76], [56]	[77], [61], [78]	[50], [79], [80], [81], [82]
Algorithms validated by experimental tests	[39], [52], [53]	[57]	[62], [63], [64], [66]	[67], [69]	[71], [73]				[80], [79]

2.4 Summary

This chapter has presented a comprehensive summary of the up-to-date methodologies that are attempted to be used for mobile robot navigation in dynamic environments. The discussed methods were divided into two main groups: (i) classical methods and (ii) heuristic methods. Path planning approaches such as APF, GA, FL, ANN, RL, PSO, BFO and ACO were considered for this review. An investigation was also conducted on path planning algorithms based on identified key features. This investigation helps to understand the common drawbacks of existing algorithms and the research gaps in the field of interest.

In general, it was noticed that many path planning algorithms used in dynamic en-

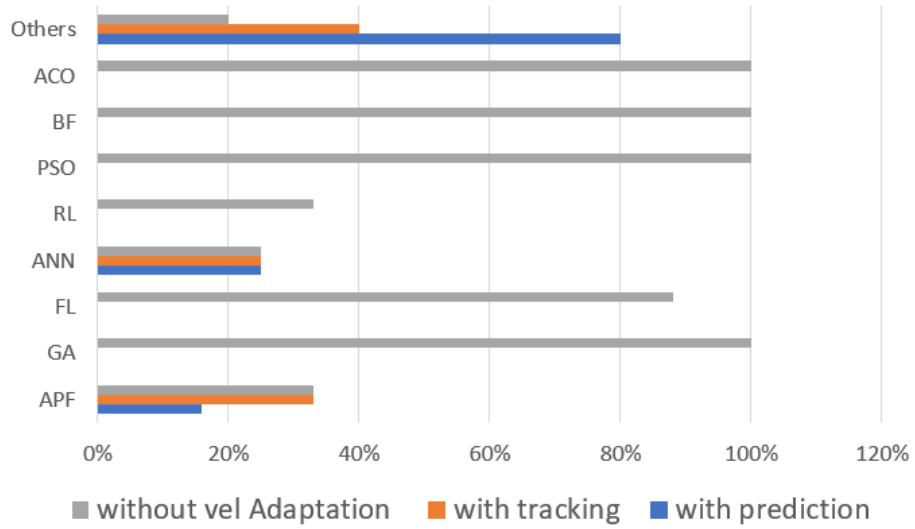


Fig. 2.7 Navigation Techniques Based on Velocity Adaptation and Moving Object Prediction

vironments not only do not use a proper method to track MOs but also do not adapt the velocities of moving objects in the navigation decision-making process.

From this literature review, it was also identified that AI-based techniques are more suitable to handle high uncertainties in unknown dynamic environments. Also, the review has shown that there are only a few navigation algorithms capable of navigating robots in dynamic environments with a moving goal. Almost all of the algorithms used in this review have been verified by simulation tests. However, a conceivable number of algorithms (63%) have not been experimentally validated.

As mentioned in Chapter 1, in this research, we developed a navigation algorithm capable of navigating robots in unknown dynamic environments using an Agoraphilic (FSA) concept while eliminating the aforementioned common weaknesses. Table 2.3 shows the proposed solutions to address these limitations along the path of developing the novel ANADE.

Chapter 3 further investigates suitable object tracking methodologies to combine with an FSA concept to successfully track the locations and estimate the velocities of

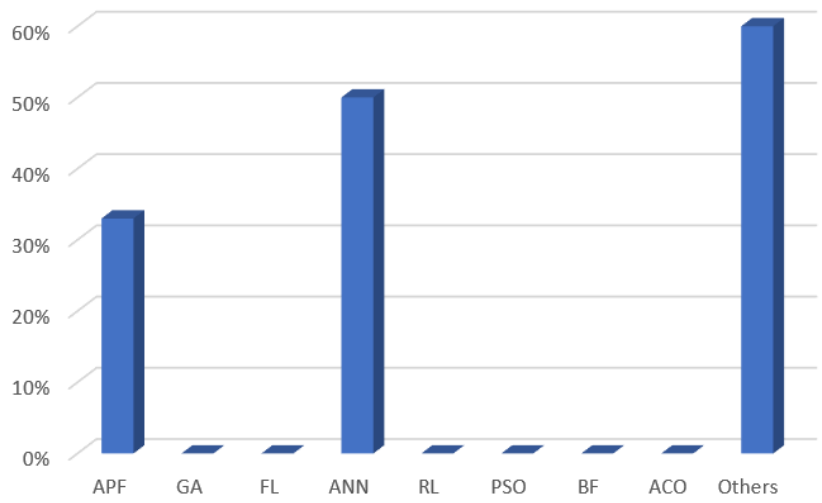


Fig. 2.8 Navigation Techniques Based on the Ability to Navigate in a Dynamic Environment with a Moving Goal

MOs.

Tab. 2.3 Identified General Weaknesses and Proposed Solutions in the ANADEs.

Identified general weakness	Proposed solution
Not having an integrated tracking method to locate moving obstacles.	A tracking algorithm is introduced in the ANADE I (Chapter 4), which is also used by the ANADE II, III, IV and V to track moving obstacles.
Not having a methodology to adapt the velocities of moving objects in the navigation decision making process.	Obstacle moving velocity–based path prediction method is introduced in the ANADE II (Chapter 5), which is also used by the ANADE III, IV and V.

Identified general weakness	Proposed solution
Not handling high uncertainties in unknown dynamic environments well	Fuzzy Logic Controller (FLC) is introduced in the ANADE III (Chapter 6). A further improved self-tuning FLC was used in the ANADE IV (Chapter 7) and the ANADE V (Chapter 8).
Not hunting a moving goal in unknown dynamic environments well	ANADE V (Chapter 8) is an FSA-based navigation system capable of following and hunting a moving target in unknown dynamic environments.
Limited experimental validation	The ANADE II, III, IV, and V are validated via experiments.

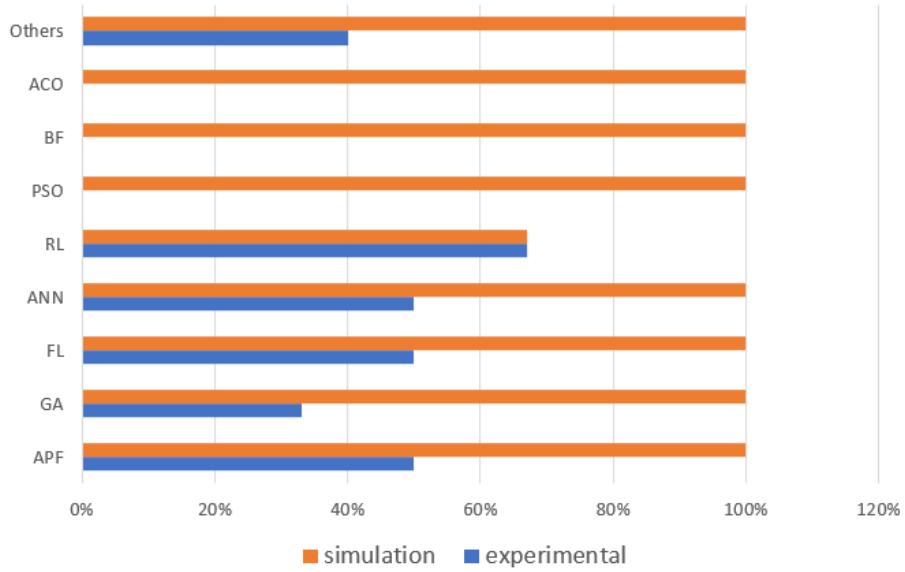


Fig. 2.9 Published Validation Methods (Simulations and Experiments) of Different Navigation Techniques

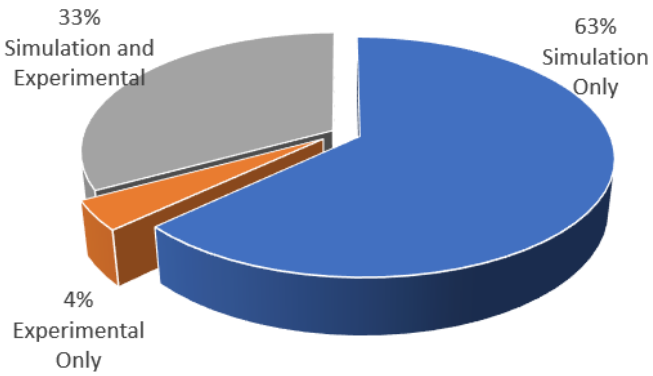


Fig. 2.10 Methods Used for Validating Navigation Techniques

3

Chapter 3: Object Tracking

3.1 Introduction

Navigating mobile robots and autonomous vehicles in unknown, dynamically cluttered environments is a challenging task. As discussed in Chapter 2, to navigate mobile robots safely and efficiently in dynamically cluttered environments would require moving obstacle tracking with reliable locations and velocity (states) estimation. The short-term state prediction of obstacles and accurate obstacle tracking are fundamental requirements for the safe and efficient navigation of mobile robots in unknown, dynamically cluttered environments.

A considerable amount of research is being done in the fields of obstacle tracking and target tracking. Estimating the locations of MOs can be achieved by using obstacle tracking and prediction algorithms [83–89]. In this chapter selected tracking methods are modelled and tested under common benchmarking obstacle motion conditions to evaluate the suitability of selected algorithms for local path planning problem.

The main steps of obstacle tracking can be identified as obstacle detection, obstacle classification and obstacle tracking [90]. There are two main strategies used for obstacle detection: detection-free tracking (DFT) and detection-based tracking (DBT) [91] (see Fig. 3.1). In DFT, obstacles are tracked manually. In DBT, some specific al-

gorithms—such as frame difference, background subtraction and optical flow are used [83]. When multiple obstacle tracking is considered, various sensors are used, such as laser range finder sensors [85, 92, 93], sonar sensors [92] and cameras [83, 94], where computer vision also plays an important role [89, 90, 95, 96].

Once the obstacles are detected, obstacle differentiation methods are used to identify obstacles separately. Obstacle classification methods have been developed based on obstacles' appearance and motion. In the final step, obstacle tracking and the estimation of the states of the obstacle is conducted. Obstacle tracking algorithms can be divided into three main groups [97] (see Fig. 3.1):

1. physical-based methods
2. statistical-based methods
3. cooperative-based methods.

The physical-based obstacle tracking algorithms can be further divided into three main groups [91]:

- point-based methods
- Kernel-based [98]
- Silhouette-based methods [99].

Physical-based tracking methods are commonly used to estimate the positions of MOs and track them in a dynamic environment. Reasons for using physical-based tracking methods will be discussed in Section 3.2.

MOs can be divided into two groups: slow-manoeuvring and fast-manoeuvring obstacles. Slow-manoeuvring obstacles have a constant velocity or small acceleration.

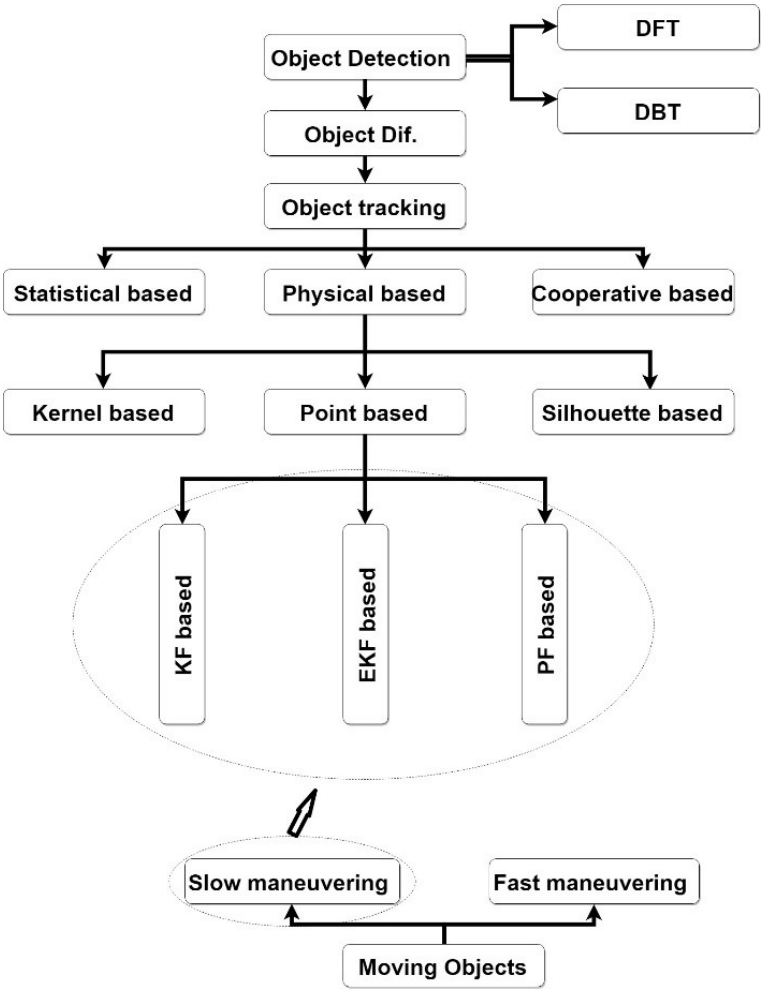


Fig. 3.1 Obstacle Tracking Techniques

Fast-manoeuvring obstacles rapidly change their velocity [100]. Generally, dynamic obstacles can be considered slow-manoeuvring obstacles if a sufficiently high sampling rate is maintained [101]. Those obstacles can be tracked by point-based algorithms.

When slow-manoeuvring obstacles are tracked by point-based algorithms, motion models for these obstacles must be derived [97, 101]. A few motion models have been developed for slow-manoeuvring obstacles. Constant velocity models and constant acceleration models are widely used to model the motion of slow-manoeuvring obstacles. This chapter analyses the physical-based obstacle tracking algorithms used for the online obstacle tracking of slow-manoeuvring obstacles for mobile robot navigation.

3.2 Physical- and Statistical-Based Obstacle Tracking

Obstacle tracking algorithms can be divided into three main groups based on their motion estimation approach, as mentioned in Section 3.1: physical-based, statistical-based and cooperative-based methods. Cooperative-based tracking methods are developed by combining the physical- and statistical-based methods. Cooperative-based tracking methods require knowledge about the behaviours of MOs, which has to be collected and processed in advance [97]. Because of this limitation, cooperative-based methods are not suitable for most online tracking applications in unknown environments. A thorough comparative study on physical- and statistical-based algorithms has been provided in this section.

3.2.1 Physical-Based Tracking

Physical-based location tracking techniques generally assume that a motion model can be developed to represent the movements of a dynamic obstacle. This can be achieved if the required parameters (e.g., measurements, system matrix and sampling time) are known. Those techniques usually employ an algorithm to estimate the current location, velocity, acceleration and orientation of MOs. Coefficients of the motion model are learned by an online learning process. The motion model is then used to compute the next states of the MOs [97].

Physical-based tracking and prediction can be completed by using a Kalman Filter (KF), Extended Kalman Filter (EKF) or Particle Filter (PF).

3.2.2 Statistical-Based Tracking

In this method, MOs tend to move in accordance with certain trajectories that depend on the nature of the obstacle and the architecture of the environment. These estimation methods use a two-stage process involving a learning stage and a prediction stage. Despite that this method can be used for long-term prediction, the learning stage requirement makes it unsuitable for online tracking [102]. Grid-based techniques [103, 104] and cluster-based techniques [105] are some of the common algorithms used in this method.

3.2.3 Comparison of Statistical-Based Tracking and Physical-Based Tracking

As shown in Table 3.1, physical-based tracking velocity in the nth sample is good for online, short-term prediction of slow-manoeuvring acceleration obstacles in an unknown dynamic environment. Statistical-based tracking methods are good for offline, short- or long-term prediction of known MOs in a known dynamic environment. Therefore, it is essential to use a physical-based tracking method if the application needs online tracking. Point-based algorithms can be identified as a subgroup of physical-based tracking.

3.3 Benchmarking Model

As stated in Section 3.1, if a good sampling rate is maintained, an MO can be considered a slow-manoeuvring obstacle. In this section, a slow-manoeuvring obstacle is modelled with constant acceleration. The same model was used to examine and compare the tracking performance for KF-, EKF- and PF-based algorithms. The motion model is expressed in Equations 3.1-3.8

Tab. 3.1 Advantages and Disadvantages of Physical-Based and Statistical-Based Tracking

	Advantages	Disadvantages
Physical based	Online prediction method	Needs a proper motion model
	Can start prediction from the first time step(in most cases)	In most cases, it takes contest acceleration or velocity models which will reduce the accuracy of estimation.
	Low computation cost	Inaccurate in long term prediction.
	Can predict location, velocity , acceleration (in some cases), orientation, angular velocity, angular acceleration (in some cases)	Sudden changes in obstacles' motion may create issues in prediction.
	Good for short-term prediction	Noisy measurements due to noisy sensors
Statistical based	No need of a motion model	Needs offline training
	There are no acceleration or velocity constrains	Motion patterns have to be derived.
	Has fewer issues with noises	Motion rules have to be made
	Can predict long-term states of obstacles	

$$s = ut + 1/2at^2 \tag{3.1}$$

$$v = u + at \tag{3.2}$$

where:

$s \stackrel{\text{def}}{=}$ distance travelled in a sample

$u \stackrel{\text{def}}{=}$ velocity in n^{th} sample

$a \stackrel{\text{def}}{=}$ acceleration

$v \stackrel{\text{def}}{=}$ velocity in $n + 1^{th}$ sample

Applying 1 and 2 to the x-direction

$$x(t+1) = x(t) + dx(t)/dt \times t + 1/2 \times a_x(t) \times t^2 \tag{3.3}$$

$$dx(t+1)/dt = dx(t)/dt + a_x(t) \times t \tag{3.4}$$

Applying 1 and 2 to the y-direction:

$$\begin{aligned} y(t+1) &= y(t) + dy(t)/dt \times t + 1/2 \times a_y(t) \times t^2 \qquad (3.5) \\ dy(t+1)/dt &= dy(t)/dt + a_y(t) \times t \end{aligned}$$

$$\begin{bmatrix} x(t+1) \\ y(t+1) \\ dx(t+1)/dt \\ dy(t+1)/dt \end{bmatrix} = \begin{bmatrix} 1 & 0 & t & 0 \\ 0 & 1 & 0 & t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x(t) \\ y(t) \\ dx(t)/dt \\ dy(t)/dt \end{bmatrix} + \begin{bmatrix} 1/2t^2 \times a_x \\ 1/2t^2 \times a_y \\ t \times a_x \\ t \times a_y \end{bmatrix} 1 + w(t) \tag{3.7}$$

$x(t) \stackrel{\text{def}}{=}$ position in the x-direction

where: $y(t) \stackrel{\text{def}}{=}$ position in the y-direction

$w(t) \stackrel{\text{def}}{=}$ process noise

When an obstacle is tracked in two-dimensional space, its location (x, y) can be measured. Therefore, the output equation can be developed as shown in Equation 3.8

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ y(t) \\ dx(t) \\ dy(t) \end{bmatrix} + v(t) \tag{3.8}$$

where:

v(t)$\stackrel{\text{def}}{=}$ measurement noise

3.3.1 Simulation Setup

A simulation platform was developed using MATLAB to analyse the tracking performances of the KF-, EKF- and PF-based algorithms. The developed simulation platform consists of four main parts,

1. Single moving object motion simulator
2. Sensory system
3. Tracking algorithm running section
4. Output display, results plotting and statistical data providing system

The single moving object simulator mimics the motion behaviour of a moving object. The objects starting position, starting velocity and its instantaneous acceleration

can be changed accordingly. The simulator measures the location of the object. The developed simulation platform assumes that the existing sensory system and its software systems detect the object and measure its location, Fig. 3.2. There is a number of different types of object detection methods and object location measuring systems. However, in this thesis they are not the main research areas. Therefore, the developed simulation platform assumes that the existing sensory system and its software systems detect the objects and measure their location with noise. Consequently, the behaviour of the sensory system was simply mimicked by adding noise to the actual location of the moving object (However, when the real-world experiments are conducted to test the navigation algorithm developed in this research, a colour-based object detection and location measuring systems were used to detect multiple objects and measure their locations. These will be further discussed in Section 4.4 and 5.4).

In the tracking algorithm running section, KF-, EKF- and PF- based algorithms were used to track a moving object and estimate its location and velocity in real-time. The last section of this simulation platform is the output display, results plotting and statistical data providing system. This system shows the actual motion of the moving object, real-time measurements of the noisy sensory system and estimated locations of the moving objects by tracking algorithm. Furthermore, at the end of the simulation test this provides charts and other statistical data to analyse the used tracking algorithm.

3.3.2 Point-Based Obstacle Tracking Algorithms

In point-based obstacle tracking, feature points of MOs are used to represent and track the MOs [106]. A KF, EKF and PF are some of the powerful approaches used in point-based multiple obstacle tracking algorithms.

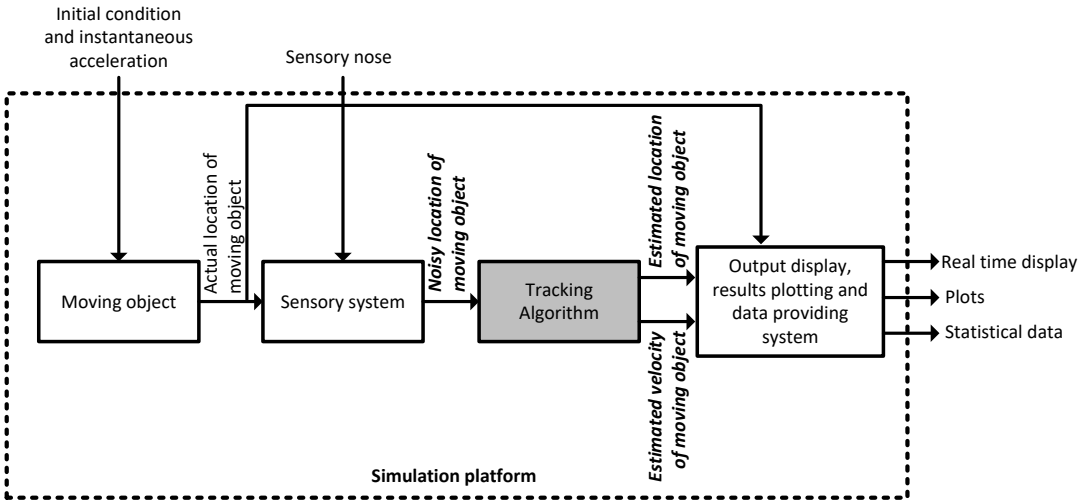


Fig. 3.2 Block diagram of the developed simulation platform

3.3.3 Kalman Filter–Based Obstacle Tracking Algorithms

KF-based algorithms work in two process states: the prediction state and the correction or update state. In the prediction state, a KF yields the estimates of current state variables, along with their own uncertainties. In the update state, pre-estimates are updated by a weighted average [107].

These algorithms are recursive and use the current measurement and estimation in the previous sample to track an MO. A KF does not assume that the errors are Gaussian. However, the filter does produce the exact conditional probability estimate in the special case of if all the errors are Gaussian distributed [107]. If a system can be described by linear system equations, the standard KF-based algorithms can be used to estimate the states of the system.

3.3.3.1 Example Algorithms Using Kalman Filter Techniques

Two examples of algorithms using KF technique are presented and analysed in this chapter.

a) First example

Elbagir et al. [108] demonstrated the use of a KF to predict future positions and orientations of freely moving obstacles. However, the proposed algorithm and framework has limited constraints when it is applied to robot motion planning in dynamic environments. The main advantage of the proposed framework is its ability to start predicting the states from the first time step without needing a prior history [108].

b) Second example

In this example, Gomez-Ortega et al. presented a predictive navigation system for mobile robots in [109]. The system deals with unexpected MOs whose future positions over a prediction horizon are estimated with a KF approach. GAs have been used for real-time optimisation problems involved in the model-based predictive control problem [109].

3.3.3.2 Analysing the Algorithms

The performance of the KF-based algorithms was analysed using the benchmarking motion model described in Section 3.3. Both algorithms used a common filter for obstacle tracking. This filter was redeveloped using MATLAB. The obstacle model developed in Section 3.3 was tracked in one direction using the developed filter (used in the above two algorithms). Observed graphs are shown in Fig. 3.3. Simulation conditions are shown in Tab. 3.4.

The obstacle was tracked for 10 s with a sampling period of 0.1 s. The graph in Fig. 3.3 shows the actual position of the MO, its measured or observed position from the sensor and its estimated position using the KF with respect to time. When the measured and estimated positions are compared, it is clear that the KF technique has improved tracking accuracy.

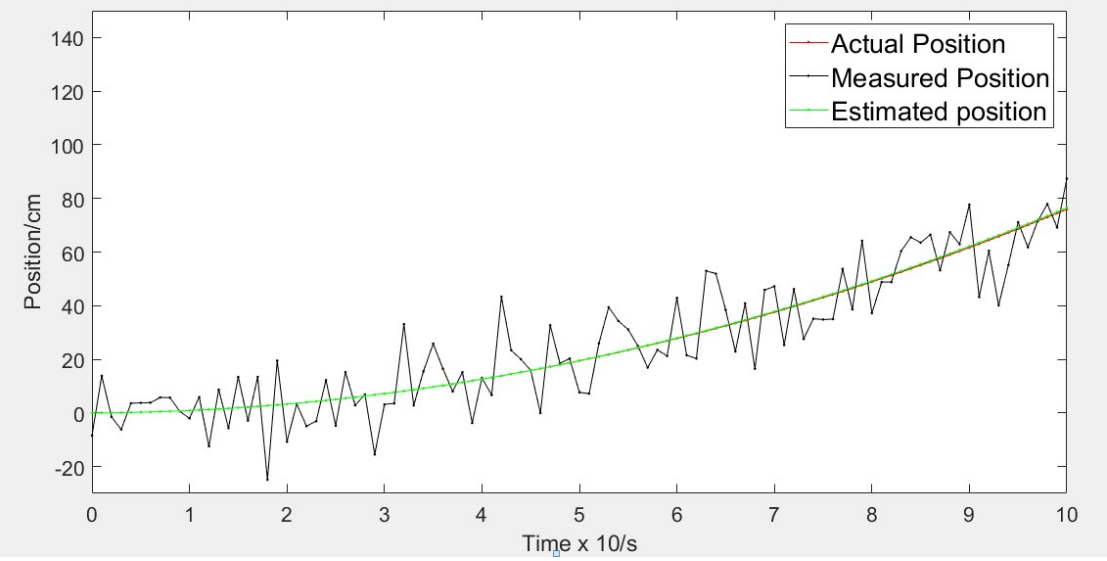


Fig. 3.3 Estimation Results of the Kalman Filter–Based Algorithm

From the analysis, it can be justified that the KF-based obstacle tracking algorithms:

- use a developed motion model for dynamic obstacles
- can predict states of the MO one-time step ahead
- assume that previous and current positions are available from sensory devices
- are capable of tracking slow-manoeuvring obstacles

3.3.4 Extended Kalman Filter (KF)-Based Obstacle Tracking Algorithms

When the system is nonlinear, KF-based algorithms do not converge [110]. Therefore, the KF cannot be used for tracking nonlinear systems. The EKF-based algorithms linearises the state space model using first-order approximation. The mean and covariance of states are propagated using Jacobian matrices [111]. Once the function is linearised using a Jacobian matrix, these algorithms use the filter operating point to estimate the states. This process is performed at each time step in EKF-based algorithms.

To use EKF-based algorithms for a nonlinear system, transition and observation models can be nonlinear, but they should be differentiable. At the same time, process

noise and measurement noise should have zero mean Gaussian distributions.

3.3.4.1 Example Algorithms Using Extended Kalman Filter Techniques

The following two examples have attempted to use an EKF.

a) First example

Madhavan et al. [92] reported work that had been conducted using an EKF-based algorithm for an MO prediction framework for off-road autonomous navigation. This method can predict an MO's future position for the path planning of unmanned ground vehicle navigation in dynamic environments. The short-term predictions are based on an EKF working symbiotically with a probabilistic obstacle classification scheme. The proposed framework was shown to deliver reliable position estimates for a wheeled vehicle.

b) Second example

Rebai et. al. [93] proposed a method to detect and track MOs. The method was based on an EKF algorithm for tracking mobile obstacles. In this approach, the authors have used a laser range finder to detect MOs. Once the obstacles are detected and classified, the EKF-based algorithm deals with dynamic obstacle tracking (DOT). This is achieved by a set of EKF, where an EKF is initialised for each mobile obstacle and evaluated for the prediction and update phases. The authors have modelled the MO nonlinearly; hence they had to use an EKF-based algorithm. The motion model that was used is expressed in Equations 3.9-3.11.

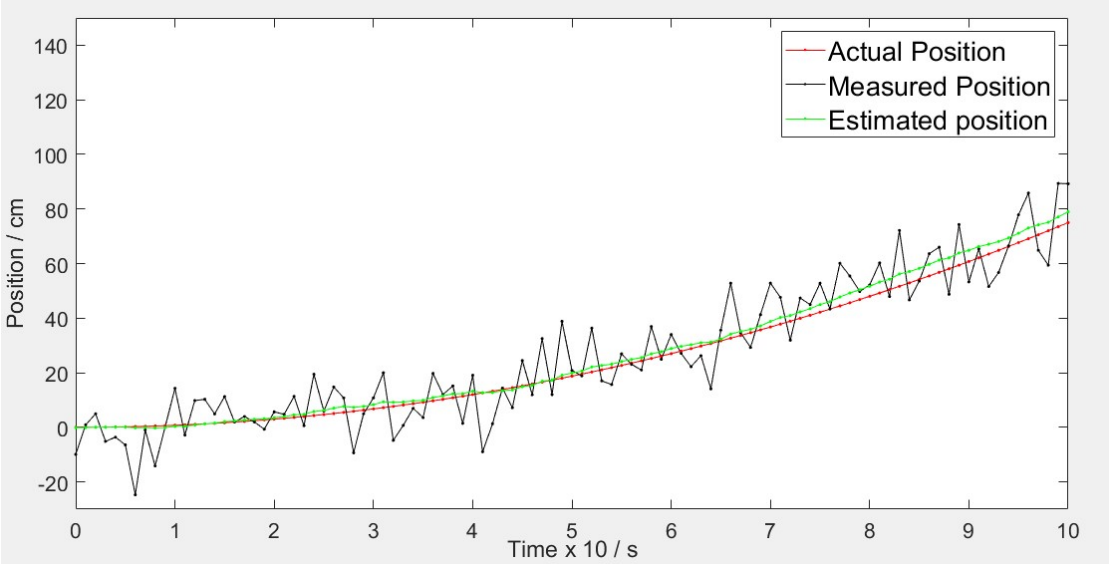


Fig. 3.4 Estimation Results of the Extended Kalman Filter–Based Algorithm

$$x_{k+1} = x_k + v_k.\Delta t.cos(\theta_k) \tag{3.9}$$

$$y_{k+1} = y_k + v_k.\Delta t.sine(\theta_k) \tag{3.10}$$

$$\theta_{k+1} = \theta_k + \Delta\theta \tag{3.11}$$

Where the state vector is $X = [x, y, \theta]'$. Also, x, y are the coordinates of the obstacle centre, θ is the direction of the velocity vector, $'v'$ is the translational velocity and $\Delta\theta$ is the orientation change.

3.3.4.2 Analysing the Algorithms

These two algorithms have used an EKF-based technique to predict and track MOs. The filters used in the above two algorithms were redeveloped using MATLAB. The same benchmarking model in Section 3.3 was tracked by the developed filter with the same conditions previously used. Observed results are shown in Fig. 3.4. Simulation conditions are shown in Tab. 3.4.

Fig. 3.4 shows the actual position of the MO, its measured or observed position from the sensor and its estimated position using the PF with respect to time. It was observed that the sensors' measured or observed positions were noisy and had a considerably higher margin for error. However, it was also observed from the obtained results that the estimated positions are much more accurate than the measured position (see Fig. 3.4). That was the effect of applying the filter.

From the analysis, the EKF-based obstacle tracking algorithms:

- can be used to track slow manoeuvring obstacles
- use an obstacle model
- can predict one step ahead
- are capable of tracking non-linear systems (developed filter proves the linearisation)

3.3.5 Particle Filter-Based obstacle Tracking Algorithms

PF-based algorithms are sequential Monte Carlo methods based on point mass (or 'particle') representations of probability densities [112, 113]. PF-based methods eliminate the main constraints of KF methods. Therefore, PF-based methods can be applied to any state space model (nonlinear and non-Gaussian). To model dynamic systems, the focus will be on the discrete-time formulation of the problem. At least two models are required for PF-based algorithms to analyse and make inferences about a dynamic system [113]: (i) a model describing the evolution of the state with time (the system model) and (ii) a model relating the noisy measurements of the state (the measurement model). It has been assumed that these models are available in a probabilistic form.

A posterior probability density function (PDF) of the state is constructed in a Bayesian approach. All available information, including the measurements, are used to develop the PDF. This PDF embodies all available statistical information. Therefore, it represents a broader solution to the estimation problem. In principle, an optimal estimate of the state may be obtained from the PDF. For smoother navigation, an estimate is required every time that a measurement is received. In this case, a recursive filter is used as a convenient solution. The prediction stage uses the system model to predict the state PDF for the next iteration. The update operation uses the latest measurement to modify the prediction PDF. This is achieved using Bayes' theorem. This is the mechanism for updating knowledge about the target state in light of the extra information from new data [113].

3.3.5.1 Example Algorithms Using Particle Filter Techniques

Two examples of algorithms using the PF technique are presented and analysed in this chapter.

a) First example

The presented method in [112] is based on PFs and Sample-Based Joint Probabilistic Data Association (SJPDAF). This method can be used to deal with nonlinear and non-Gaussian models. It uses a probability density description of the model instead of a linear and Gaussian model. It is mentioned that PFs are used to estimate the states of nonlinear and non-Gaussian systems. PFs are used to approximate a posterior density function as a set of weighted samples ('particles') to estimate states.

b) Second example

Almeida et al. presented a method for tracking multiple MOs in dynamic environ-

Tab. 3.2 Performance of the Particle Filter with Different Numbers of Particles

Number of particles	Max. Error/m	Variation/m^2	Computation time/s
100	5.169	10.08	0.2864
1000	3.7494	1.8363	4.4930
5000	2.6349	1.2633	123.3613

ments using particles filters and SJPDAFs [114]. However, the research problem that was addressed was the development of a sensor-based method to track MOs around the robot. This framework is capable of modelling the dynamic environment and predicting the states of MOs. This allows the robot to detect and track several MOs using a range finder sensor. PFs and SJPDAFs are the methods used in this approach.

3.3.5.2 Analysing the Algorithms

A PF algorithm was developed using MATLAB to analyse its performance. An obstacle with the same motion model (described in Section 3.3) was tracked using a PF model (simulation conditions are shown in Tab. 3.4.). Then results were plotted, as shown in Fig. 3.5 (PF with 100 particles). The performance of the PF was further analysed by changing the number of particles used in the filter results, as shown in Table 3.2. For the comparative study, a PF with 100 particles was used to keep the computation time in-rage with the other two methods.

Fig. 3.5 shows the actual position of the MO, its measured or observed position from the sensor and its estimated position using the PF with respect to time. It was noticed that the measured or observed positions were noisy and had a considerably higher margin for error. However, this noise was suppressed, resulting in a more accurate estimation following the application of the filter.

It was also observed that when a higher number of particles were used (see Table

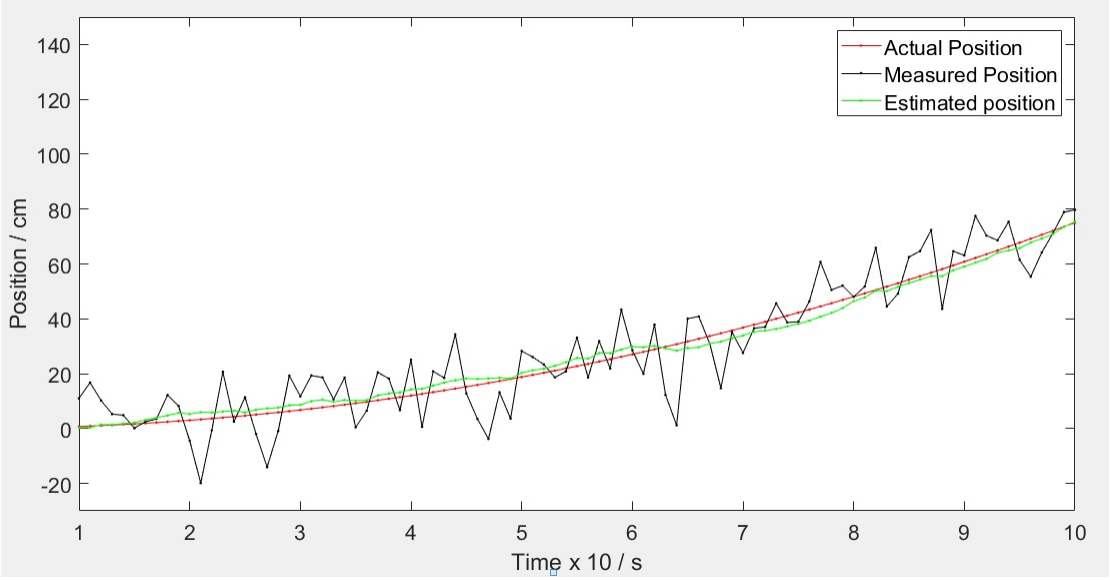


Fig. 3.5 Estimation Results of a Particle Filter–Based Algorithm

3.2), the estimation error was further reduced, but the computation time was considerably increased (see Table 3.2).

From the analysis, the PF-based obstacle tracking algorithms:

- can be used to track slow-manoeuvring obstacles
- can be used to track non-Gaussian models
- use an obstacle motion model
- become more accurate when the filter uses more particles
- can track non-linear systems

3.3.6 Comparison of Algorithms According to the State Estimator

A general comparison of the algorithms based on error distribution, computation cost and the system model is provided in Table 3.3. KF-based algorithms can be used to track linear Gaussian systems. EKF-based state estimators are capable of tracking non-

Tab. 3.3 Comparison of Algorithms

State estimator	System model	Error distribution	Computation cost
KF	Linear	Gaussian	Low
EKF	Non-linear (locally linear)	Gaussian	Low
PF	Non-linear/linear	Non- Gaussian /Gaussian	High

Tab. 3.4 Summarised simulation conditions

Starting point of the MO (x coordinate)	0 cm
Initial speed of MO in x direction	0 cm/s
Base acceleration	1.5 cm/s2
simulated measurement available for tracking algorithm	Location of MO (with sensory noise)
Simulation scenario	An obstacle moves (an instantaneous random noise was added to the acceleration of obstacle to mimic the natural motion pattern) on a straight line (on x axis) is tracked by different tracking algorithms

linear but Gaussian systems. PF-based algorithms can track nonlinear and non-Gaussian systems.

3.4 Results and Discussion

3.4.1 Testing Results

Three selected algorithms (KF-, EKF- and PF-based algorithms) were used to track a slow-manoeuvring obstacle that moves in a straight line with a constant acceleration with random acceleration noise. Gaussian distributed noise was used for measurement and process noise in all three cases. The same sampling rate was maintained in all three algorithms. The discussed simulation platform in Section 3.3.1 was used to conduct these tests. As shown in Fig. 3.2 the only available simulated measurement for the tracking algorithm was the location of the object with sensory noise. Summarised simulation conditions are shown in Table 3.4.

In all three cases, the actual, measured and estimated positions were plotted in the same graph, as shown in Fig. 3.3-3.5. The difference between the actual and estimated positions, which is known as estimation error, was then calculated for each estimation. The estimation errors for all three algorithms were plotted in the same graph, as shown in Fig. 3.6. Maximum estimation error, the variance of estimation errors and computation time for each case were calculated and displayed in the bar chart shown in Fig. 3.7. The developed graphs and bar chart were used to analyse the performances of each algorithm in terms of accuracy. A brief analysis of computation time was also completed.

Fig. 3.3–3.5 show the actual, measured and estimated positions of a single MO, which was tracked using KF-, EKF- and PF-based algorithms, respectively. The number of particles in the PF algorithm was adjusted to 100 to compare the accuracy and keep the computation time of the PF in the same range as the KF and EKF. The results show that tracking was significantly improved by using these algorithms. As shown in Fig. 3.3–3.5, the measured positions (black lines) are quite deviated from the actual positions (red lines). After applying the tracking algorithms, the accuracy of the estimated positions (green lines) improved. Fig. 3.6 shows the estimation error of the algorithms during the entire process. The results in Fig. 3.6 show that when a slow-manoeuvring obstacle is tracked by the KF-based algorithm, the estimation error and variation are smaller compared to that of the other two methods. The PF-based algorithm shows higher variations in estimation error.

The maximum estimation error was found with PF-based algorithms compared with that of the other two algorithms. The error was around 5.1 cm (for a = 1.5cm/s2, and the sampling period is 0.1 s, then the maximum error was 5.1 cm). The KF-based algorithm

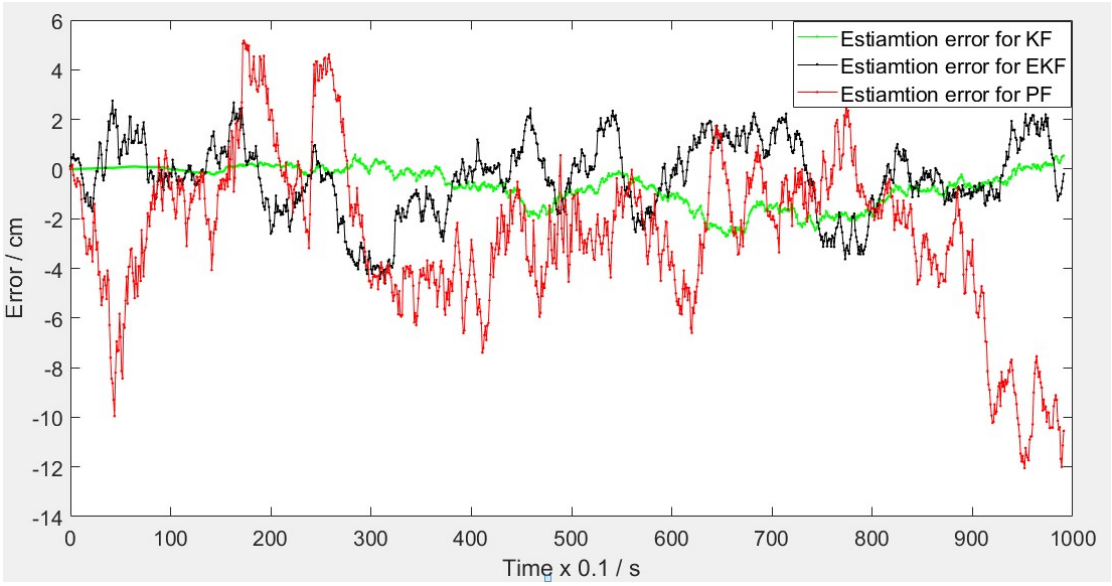


Fig. 3.6 Estimation Error Comparison between Kalman Filter–, Extended Kalman Filter– and Particle Filter–Based Algorithms

has the lowest maximum estimation error, as shown in Fig. 3.7. The maximum estimation error for the KF-based algorithm was 0.7 cm. The PF-based algorithm had the highest estimation error variance, with the lowest being in the KF-based algorithm, as shown in Fig. 3.7. The accuracy of the PF can be improved by increasing the number of particles, as shown in Table3.2. However, the computation time is increased compared with that of the other two methods. As shown in Fig. 3.7, the computation times for the KF, EKF and PF were 0.21834 s, 0.2938 s and 0.2864 s, respectively. The KF-based algorithm was slightly faster than the other tested algorithms.

When the overall estimation error is considered, during the entire estimation process, the KF-based algorithm had the lowest error, while the PF-based algorithm had the highest, as shown in Fig. 3.7. As mentioned in Section 3.1, by maintaining a good sampling rate an MO can be considered a slow-manoeuvring obstacle [101]. Therefore, during two neighbouring samples the motion model can be taken as a linear model. This is the reason why KF produced the lowest positioning error in these tests although a moving object with a random acceleration noise was tracked.

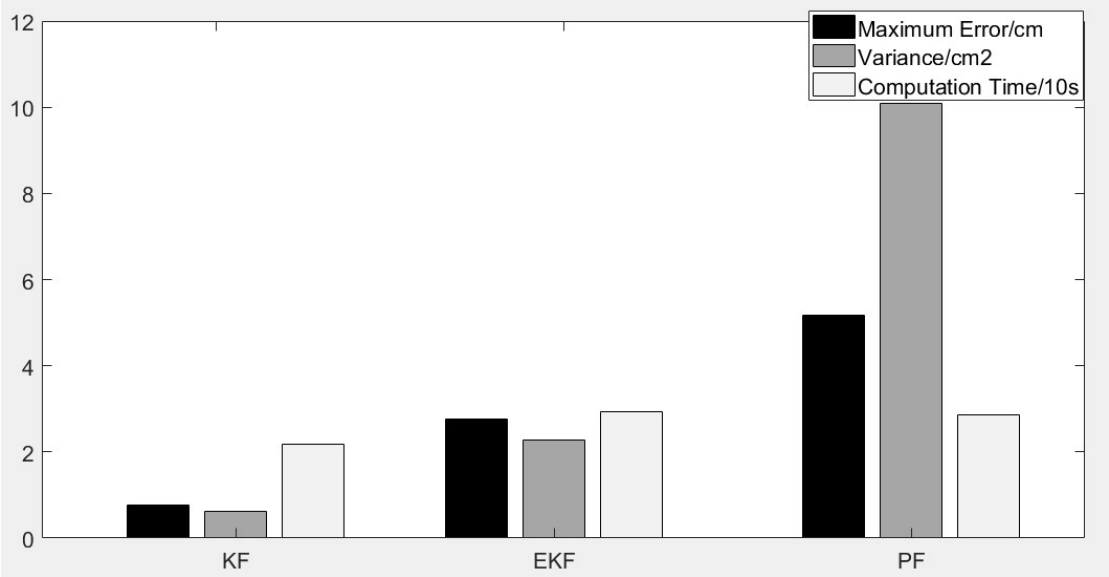


Fig. 3.7 Estimation Error Comparison between Kalman Filter–, Extended Kalman Filter– and Particle Filter–Based Algorithms

3.5 Summary

In this chapter, KF-, EKF- and PF-based dynamic obstacle tracking methods were developed to track slow-manoeuvring obstacles in a dynamic environment. The moving obstacles were modelled as constant acceleration obstacles by maintaining a sufficient sampling rate.

The conducted comparative simulation tests showed that KF-based algorithms are more accurate than are EKF- and PF-based algorithms. EKF-based algorithms are more suitable for tracking nonlinear but Gaussian models because the EKF linearises the state space model. If the system model is nonlinear and non-Gaussian, then PF-based algorithms would be a better option due to their point mass representations of probability densities. However, the accuracy of the PF-based algorithm was low when tracking the constant acceleration model under the used benchmarking motion model. This can be improved by increasing the number of particles; however, this will lead to an increased computation cost. It can be concluded that KF-based algorithms perform better in track-

ing slow-manoeuvring obstacles for mobile robot navigation in unknown, dynamically cluttered environments.

Chapter 4 discusses how a KF-based tracking algorithm is used in the development process of a novel Agoraphilic-based navigation algorithm. This novel algorithm is capable of navigating robots in unknown dynamic environments.

4

Chapter 4: Agoraphilic Navigation Algorithm in Dynamic Environment with Tracking (ANADE I)

4.1 Introduction

This chapter presents the development of a novel ANADE: the ANADE I. The ANADE I takes an optimistic view of the environment and employs the FSA concept. This novel algorithm is capable of programming mobile robots to manoeuvre in unknown environments that are both static and dynamic.

The ANADE I is an 'optimistic' navigation algorithm that introduces the fundamental building block for FSA-based navigation algorithms to operate in dynamic environments. The navigation of the mobile robot is based on the principle of looking for free space rather than avoiding obstacles. This approach has eliminated many drawbacks (mentioned in Chapter 1) of the potential field-based algorithms. The ANADE I uses a KF-based tracking methodology to estimate the location and velocity of unknown moving objects with high accuracy (see Section 3.3). The estimated locations of the moving objects are combined with static object locations in the robot's visible region to generate time-varying free space attractive forces. These time-varying forces facilitate the collision-free manoeuvring of the robot to the goal in unknown dynamic environments.

The ANADE I models the robot as a particle in the space. The free space around

the robot creates attractive forces on the robot. These force vectors can move the robot in an unknown environment arbitrarily without any collision. However, these forces do not guarantee that the robot will move to a specific target. Therefore, a force-shaping method is used to focus the force vectors towards the goal. As mentioned in Section 3.4, in dynamic environments, sensory systems do not accurately identify the locations of moving objects. Therefore, a tracking methodology is also included in this algorithm. ANADE I was developed to perform the overall navigation task according to the architecture discussed in the next section.

4.2 The Architecture of the ANADE I

A modular-based architecture was used in developing the ANADE I algorithm (see Fig. 4.1). This architecture introduces two fundamental modules with four submodules for FSA-based navigation algorithms in dynamic environments, as follows:

1. Dynamic Obstacle Tracking (DOT) module
2. Free Space Attraction (FSA) module
    (a) Free-Space Histogram (FSH) generation module
    (b) Free-Space Forces (FSF) generation module
    (c) Force shaping module
    (d) Force simulation module

These modules will be used (with improvements) in all the future ANADEs discussed in this thesis.

In the ANADE I, the tracking module takes the sensory data as its input then generates a map using the accurately estimated locations of MOs combined with the location

of SOs. The generated map of the robot's environment is fed into the FSH generation module (see Fig.4.1). The generated FSH is then converted to a set of FSFs by the FSF generation module. Subsequently, the force-shaping module regulates the FSFs such that the robot will move towards the goal. Finally, the force simulation module generates the final driving force of the robot for the current iteration. This process occurs in every iteration until the robot reaches the goal. The behaviour of the ANADE I is further explained by the pseudocode provided in Section 4.2.1.

In ANADE I following assumption was made

1. The robot is considered as a particle in the space. (This assumption is addressed by introducing a safety boundary. Further explained in sub-section 4.3.2)

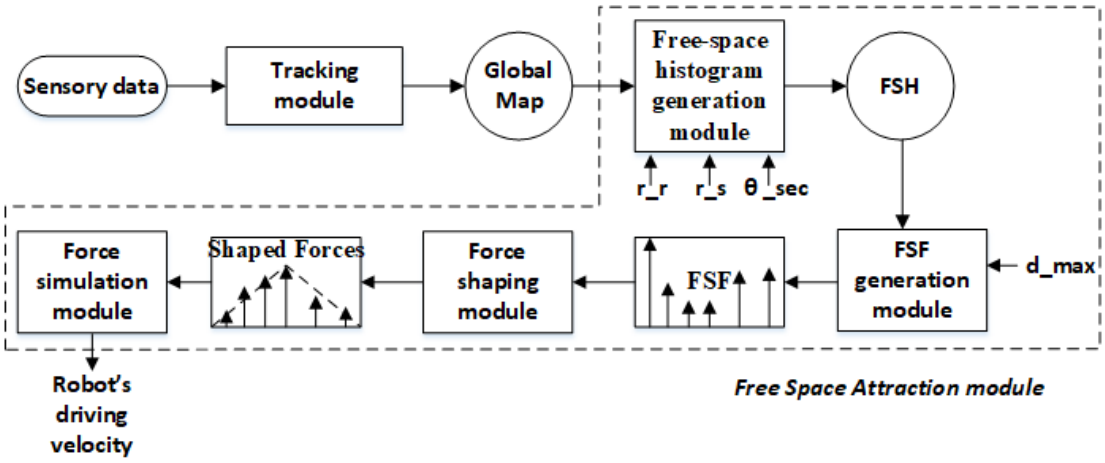


Fig. 4.1 Block Diagram of the ANADE I

4.2.1 Pseudocode

Step 1: Start the algorithm.

Step 2: Sensory data are taken and fed into the tracking module.

Step 3: The tracking module estimates the location of moving objects.

Step 4: Based on the current locations of MOs (from one output of the tracking module) and the locations of SOs (from the sensory module), a global map is generated.

Step 5: The free space passages in a global map are identified by the FSH generation module. Convert the global map into an FSH (see Fig. 4.1).

Step 6: The generated FSH is converted to a set of FSFs by the FSF generation module.

Step 7: The force-shaping module focuses the FSFs towards the goal.

Step 8: The set of shaped forces are fed into the force simulation module. This module produces one single driving force for the global map.

Step 9: If the robot has reached the goal, go to Step 10. If not, repeat Step 2–9.

Step 10: Stop

4.3 Main Modules in the ANADE I

4.3.1 Dynamic Obstacle Tracking module

The DOT module tracks the position (centre) of obstacles processed by the sensory system (further discussed in Chapter 5). Then it generates the estimated states of MOs, which provides their velocity and position. The obstacle boundary is applied to the estimated position (centre) to incorporate the size of the obstacle.

The proposed DOT module is broken down into three sections: (i) measurement model, (ii) process model and (iii) KF.

The proposed process model is a mathematical model that tries to mimic the behaviour of MOs. This process model is used to pre-estimate the states of moving objects. The pre-estimates from the process module does not have a high accuracy due

to process noise (w_k) generated by the process model. Conversely, the locations of the moving objects are measured by the sensory system and based on the output of the sensory system, and the measurement model was developed. The measured states (locations of MOs) also contain errors due to the measurement noise (v_k) of the sensors. Therefore, it is essential to use both pre-estimated states from the process model and measured states from the measurement model to estimate the actual position and velocity of moving objects with increased accuracy (see Chapter 3). As shown in Section 3.4, a KF is used to combine the outputs of these two models. The final estimated locations (output of the KF-based algorithm) of moving objects are combined with static object locations (the algorithm does not use the tracking modules for static obstacles) to generate a global map.

The development of the KF-based tracking module is shown in the following equations. This tracking solution is repeatedly used to track multiple moving obstacles. In one program cycle multiple KFs are sued. The number of used KFs are equal to the available number of moving obstacles.

The kinematic model of MOs is defined in Equations 4.1 and 4.2.

$$\begin{bmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \dot{x}_{k-1} \\ \dot{y}_{k-1} \end{bmatrix} + \begin{bmatrix} a_x \times \frac{dt^2}{2} \\ a_y \times \frac{dt^2}{2} \\ a_x \times dt \\ a_y \times dt \end{bmatrix} \times u(t) + w_k \tag{4.1}$$

$$\begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} + v_k \tag{4.2}$$

The prior estimation (state estimation based on the previous state estimation) and globalised sensory data (measurements [$x_{k,m}$,$y_{k,m}$]) are combined using the filter shown in Eq. 4.3 to derive the optimal state estimations for each moving obstacle in each iteration.

$$\begin{aligned} \begin{bmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{bmatrix} = & \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \dot{x}_{k-1} \\ \dot{y}_{k-1} \end{bmatrix} + \\ & \begin{bmatrix} a_x \times \frac{dt^2}{2} \\ a_y \times \frac{dt^2}{2} \\ a_x \times dt \\ a_y \times dt \end{bmatrix} \times u(t) + k_k \left\{ \begin{bmatrix} x_{k,m} \\ y_{k,m} \end{bmatrix} - \right. \\ & \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}^T \left(\left(\begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \dot{x}_{k-1} \\ \dot{y}_{k-1} \end{bmatrix} \right. \right. \\ & \left. \left. + \begin{bmatrix} a_x \times \frac{dt^2}{2} \\ a_y \times \frac{dt^2}{2} \\ a_x \times dt \\ a_y \times dt \end{bmatrix} \times u(t) \right) \right\} \end{aligned} \tag{4.3}$$

In this expression k_k (Kalman Gain) is found by using Eq. 4.4;

$$k_k = p_k \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}^T \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}^T p_k \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \right)^{-1} \tag{4.4}$$

Where:

$$p_k = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} p_{k-1} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ dt & 0 & 1 & 0 \\ 0 & dt & 0 & 1 \end{bmatrix} + Q \tag{4.5}$$

At each iteration following the optimal state estimation of every moving obstacle, p_k is updated as shown in Eq.4.6.

$$p_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - k_k \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ dt & 0 & 1 & 0 \\ 0 & dt & 0 & 1 \end{bmatrix} \times p_{k-1} \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ dt & 0 & 1 & 0 \\ 0 & dt & 0 & 1 \end{bmatrix} + Q \tag{4.6}$$

The updated p_k is used as $p_{(k-1)}$ in Eq.4.5 to re-calculate the new p_k in the next iteration.

where:

$a_x = (\dot{x}_{k-1} - \dot{x}_{k-2})/dt$

$a_y = (\dot{y}_{k-1} - \dot{y}_{k-2})/dt$

$v_k \sim$ N(0,R) (R=measurement error covariance)

$w_k \sim$ N(0,Q) (Q=process noise)

$p \stackrel{\text{def}}{=}$ error covariance

$(x_k, y_k) \stackrel{\text{def}}{=}$ estimated position of the MO

$(x_{km}, y_{km}) \stackrel{\text{def}}{=}$ measured position of the MO

$(a_x, a_y) \stackrel{\text{def}}{=}$ acceleration of the MO

The generated global map is used as an input for the FSA module in the ANADE I.

4.3.2 Free Space Attraction Module

The FSA module consists of four submodules:

1. the FSH generation module
2. the FSF generation module
3. the force-shaping module
4. the force simulation module.

4.3.2.1 Free Space Histogram Generation Module

The Free Space Histogram (FSH) generation module creates a robot-centred polar map using a global map. This polar map is then converted to a histogram that represents the distance profile of a global map with respect to the robot. The FSH is the first step of generating the Free Space Forces (FSFs). The FSH generation module takes a global map (the coordinate frame with respect to the global axis system) as its input. From this global map, the obstacle location information is extracted and a new two-dimensional local map is generated with respect to the robot's location (see Fig. 4.2). The process

involved in transforming the global map to a polar map is shown in Fig. 4.2. The corresponding coordinate relationships are described in 4.7 to Equations 4.9.

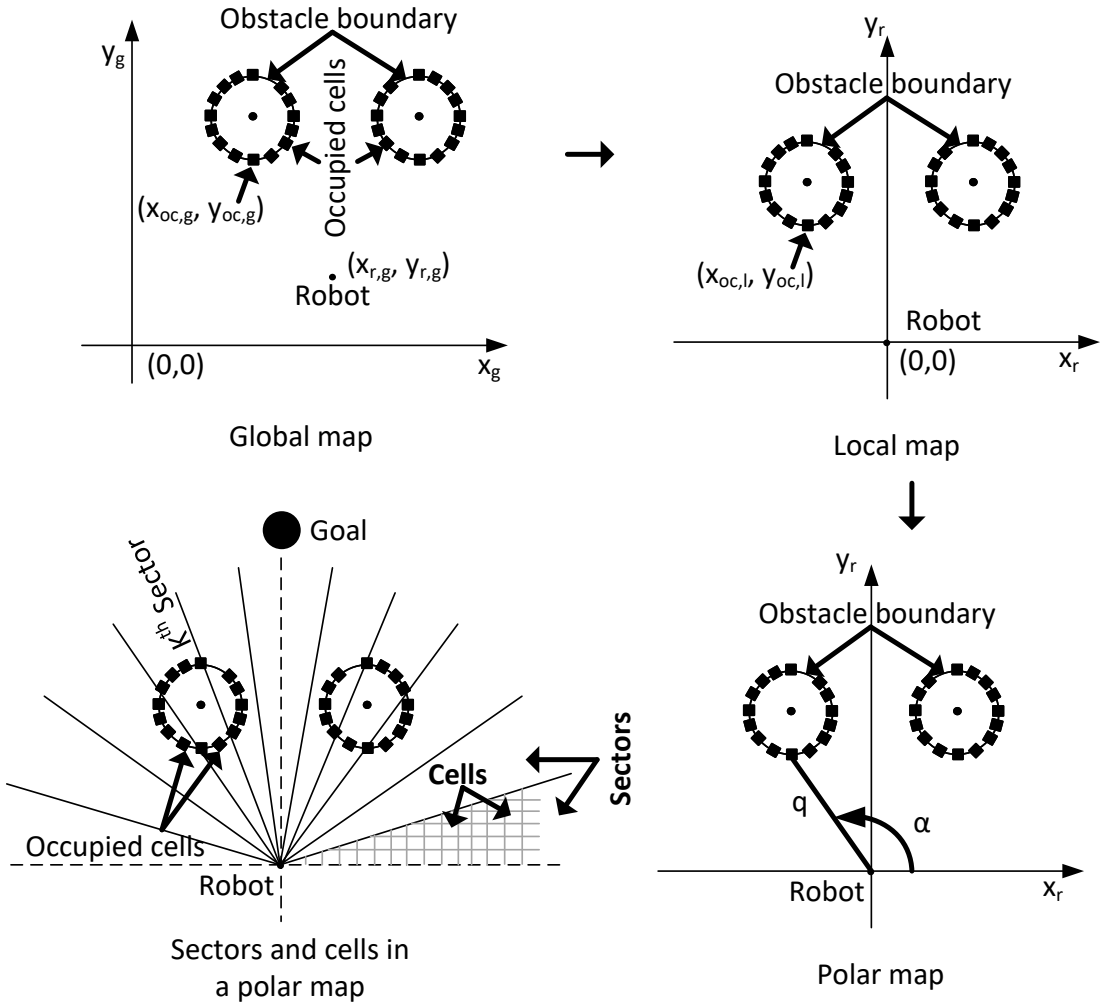


Fig. 4.2 Transformation Processes of a Global Map to a Polar Map

$$\begin{bmatrix} X_{oc,g} \\ Y_{oc,g} \end{bmatrix} = \begin{bmatrix} cos(\beta) & -sin(\beta) \\ sin(\beta) & cos(\beta) \end{bmatrix} \times \begin{bmatrix} X_{oc,l} \\ Y_{oc,l} \end{bmatrix} + \begin{bmatrix} X_{r,g} \\ Y_{r,g} \end{bmatrix} \tag{4.7}$$

$\beta \stackrel{\text{def}}{=}$ orientation of the robot

$$q = \sqrt{X_{oc,l}^2 + Y_{oc,l}^2} \tag{4.8}$$

$$\alpha = tan^{-1}\left(\frac{Y_{oc,l}}{X_{oc,l}}\right) \tag{4.9}$$

The data reduction methodology involved in this module converts this two-dimensional local map into a one-dimensional histogram. The conversion process is described below.

The polar map is initially divided into Z neighbouring sectors, which themselves are divided into cells. The cells occupied by the obstacles boundaries (see Fig. c4RevFig1) in all the sectors are identified. As previously mentioned, the Agoraphilic algorithm assumes the robot is a particle in the space. The ignored robot's volume is compensated by this module. This is accomplished by enlarging the size of all the occupied cells. The radius of each cell is increased to the sum of the robot's radius (r_r) and a predetermined safety boundary (r_s). These elements are shown in Fig. 4.3. In each sector, the closest occupied cell to the robot is considered. The minimum distance to the safety boundary corresponding to this cell is the sector distance (d_k). The derivation of dk is described by Equations 4.10 and 4.11. The FSH is then obtained, as described in Equation 4.12.

$$l_i = min\left\{\sqrt{(x_r - x)^2 + (y_r - y)^2}\right\} \tag{4.10}$$

x $\in \{x_{i,1}, x_{i,2}, ..., x_{i,j}, ..., x_{i,n-1}, x_{i,n}\}$

y $\in \{y_{i,1}, y_{i,2}, ..., y_{i,j}, ..., y_{i,n-1}, y_{i,n}\}$

where:

l_i = minimum distance to the safety boundary of i^{th} occupied cell

($x_{i,j}$,$y_{i,j}$)= j^{th} boundary coordinate of i^{th} occupied cell's safety boundary

n=number of boundary coordinates

$$d_k = \min\{L\} \tag{4.11}$$

where:

L= $\{l_1,l_2,...,l_i,...,l_m\}$

m=number of occupied cells in k^{th} sector

$$FSH = \{d_1,d_2,...,d_k,...,d_{o-1}, d_o\} \tag{4.12}$$

where:

o=number of sectors

As an example, a point on the safety boundary corresponding to obstacle 1 (OB1)'s boundary is 28.3 m (d_k) away from the robot with an angle of 148^0 to the x-axis. This information is represented by the spick 'A' in the FSH (see Fig. 4.4).

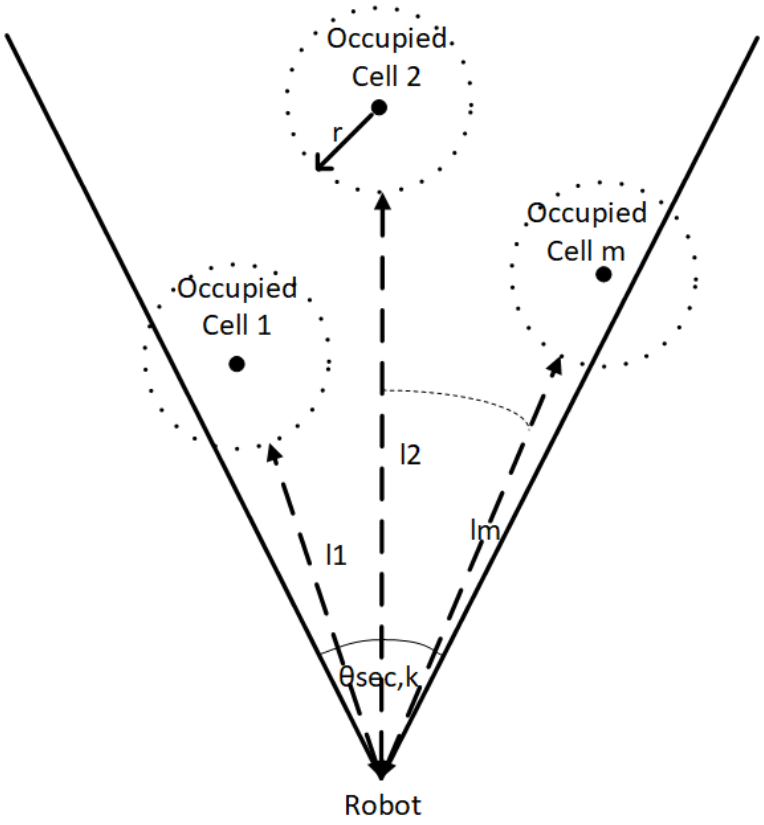


Fig. 4.3 Kth Sector with a Sector Angle of θ_{sec}, Having 'm' Number of Occupied Cells, Each with an Enlarged Radius of 'r'

4.3.2.2 Calculation of Free Space Forces

As mentioned in Chapter 1, unlike potential field-based methodologies that use two virtual forces to navigate the robot towards the goal, the Agoraphilic method employs only one force. This force is an attractive force that pulls the robot towards the open space in the general direction of the goal [115]. The initial force (F_k) generated by each sector in the FSH is directly proportional to the square of the normalised distance correspondent to that sector (see Equation 4.13). If the distance in the FSH is greater than the predetermined d_{max}, the initial force for that sector is taken as uk (see Equation 4.13). The square of the normalised distance creates a greater bias towards the free space (see Figure 4.5).

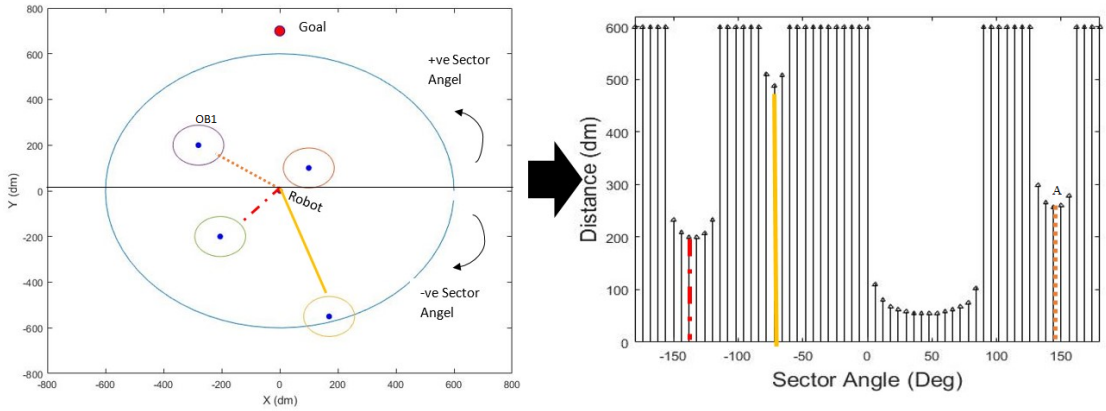


Fig. 4.4 Free Space Histogram for a Simple Environment

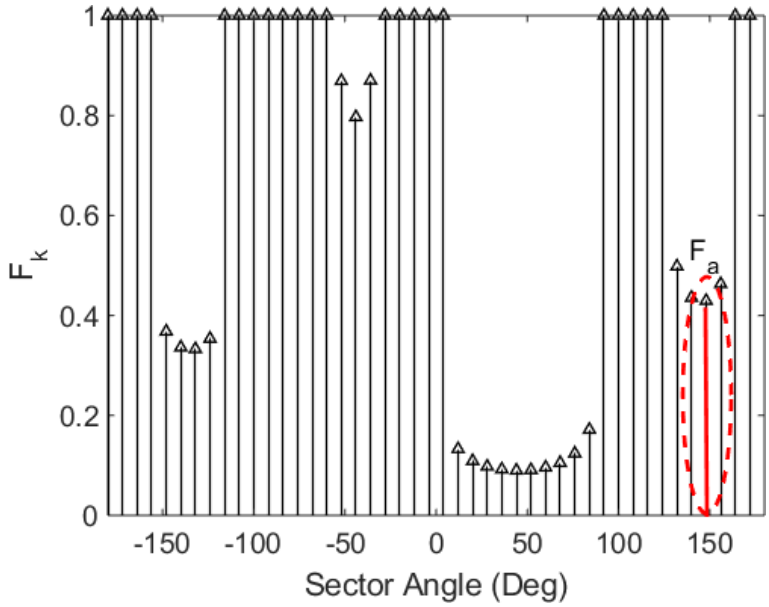


Fig. 4.5 Free Space Force Created by Normalised the Free Space Histogram

If the robot reaches a safety boundary of an object, d_k reaches 0. This makes the initial force(s) in that particular section(s) reduced to 0. This causes the robot to stop moving any closer to the object and prevents collisions.

$$F_k = \left(\frac{d_k}{d_{max}}\right)^2 \times u_k \tag{4.13}$$

Where:

u_k = [cos(θ_k) sin(θ_k)]

θ_k = sector angle

4.3.2.3 Generation of the Force-Shaping Coefficients

The attractive forces need to be shaped to navigate the robot to its goal. In a force-shaping task, a weighting methodology is used in the ANADE I to increase the magnitude of FSFs towards the goal with respect to the FSFs pointing away from the goal. The shaping coefficient (δ) is decided according to Eq. 4.14. As shown in Fig. 4.5, shaping coefficient values have a maximum of 1 when an initial force of a sector (F_k) is aligned with the goal and pointed towards the goal. If an initial force of a sector is perpendicular to the goal or when F_k does not produce a positive force component towards the goal, the corresponding shaping coefficient gets its minimum value of 0. Scaled sector forces are generated by scaling the initial sector forces using force-shaping coefficients (see Fig. 4.6).

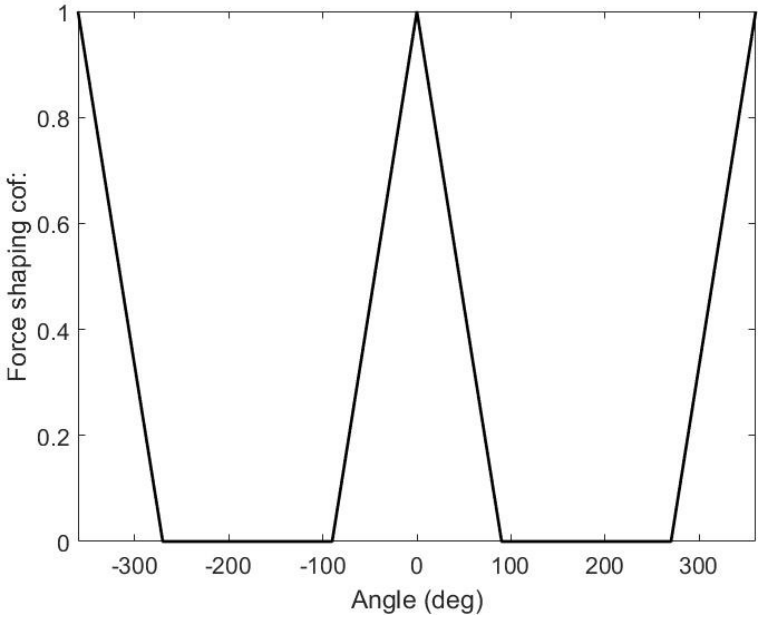


Fig. 4.6 Force-shaping Coefficient.

$$\delta(\theta) = \begin{cases} \frac{2}{\pi}\theta + 1, & \text{if } -\pi/2 \leq \theta \leq 0; \\ \frac{-2}{\pi}\theta + 1, & \text{if } 0 \leq \theta \leq \pi/2; \\ \frac{2}{\pi}\theta - 3, & \text{if } 3\pi/2 \leq \theta \leq 2\pi; \\ \frac{-2}{\pi}\theta - 3, & \text{if } -3\pi/2 \leq \theta \leq 2\pi; \\ 0, & \text{otherwise.} \end{cases} \tag{4.14}$$

4.3.2.4 Summation of Scaled Force Vectors

In the ANADE I, the robot's motion commands (RMCs) are generated by this module (this is not the case for the ANADE II, III and IV). This module takes a set of shaped FSFs as its input and generates an Instantaneous Driving Force Component (IDFC) (F_c; see Eq. 4.15). In Eq. 4.15, F_1 to F_K are a set of FSFs, and the corresponding force-shaping coefficients are denoted by δ_1 to δ_K.

In ANADE I, the IDFC is derived by taking the vector summation of shaped FSFs and multiplying by a constant scalar (λ), which is defined according to Eq. 4.16, Fig. 4.7. This keeps the upper bound of F_c to be 1. In Eq. 4.16, θ is the angle difference between two sectors. F_kSin(kθ) gives the force component of the F_k (initial force of sector k) in the direction of the goal (the sector on the robot goal line gets kθ = 90). When the robot's surroundings are free of obstacles, all initial forces (F_k) become u_k. Therefore, the IDFC becomes its maximum (F_{cmax}). To keep the upper bound of the IDFC (F_c) as 1, λ is defined as Eq. 4.16.

$$\vec{Fc} = \lambda(\delta_1\vec{F_1} + \delta_2\vec{F_2} + ... + \delta_K\vec{F_K}) \tag{4.15}$$

Fc$\Rightarrow Fc_{max}$ when $F_k \Rightarrow u_k$

Therefore,

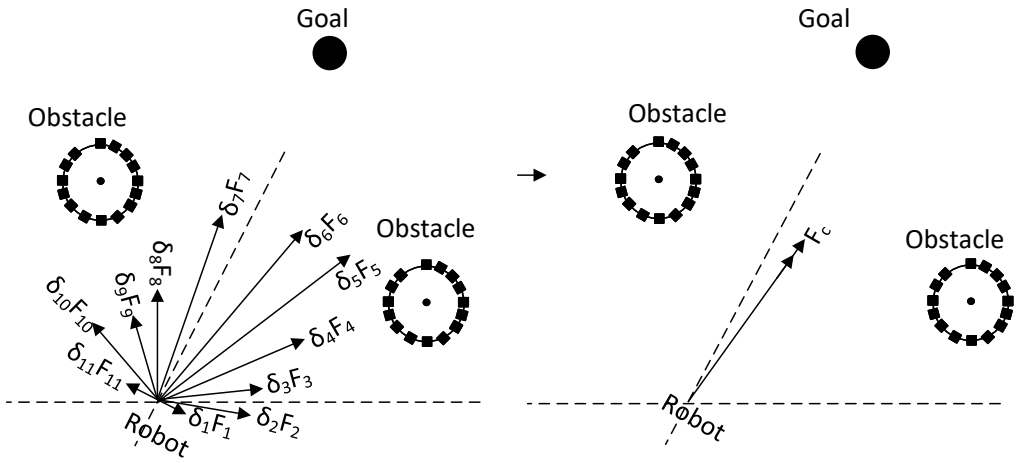


Fig. 4.7 Summation of Scaled Force Vectors.

$$\vec{Fc}_{max} = \lambda \int_0^{\pi} \vec{u_k}.Sin(\theta)d\theta$$

$$\vec{Fc}_{max} = \lambda \sum_{k=1}^{K} \vec{u_k}.Sin(k\theta)$$

but, Fc_{max} is defined as

$$\vec{Fc}_{max} = 1$$

Therefore,

$$\lambda = \frac{1}{\sum_{k=1}^{K} \vec{u_k}.Sin(k\theta)} \tag{4.16}$$

4.4 Results and Discussion

A simulation platform was developed using MATLAB to analyse the performances of the developed ANADE algorithm. The developed simulation platform consists of six main parts,

1. Moving obstacle motion simulator

2. Static obstacle simulator

3. Robot's motion simulator

4. Sensory system simulator

5. ANDE algorithm running section

6. Output display, results plotting and statistical data providing system

The moving obstacle motion simulator mimics the motion behaviour of moving obstacles. The obstacles' starting positions, stating velocities and their instantaneous acceleration can be changed accordingly, Fig. 4.8. The static obstacle simulator places static obstacles in the robot's simulation environment. The simulation module used as the sensory system measures the location of the moving and static obstacles. The developed simulation platform assumes that the existing sensory system and its software systems detects the obstacles and measure their location. There are number of different types of obstacle detection methods and obstacle location measuring systems, however, in this thesis, these are not the main research areas. Therefore, the developed simulation platform assumes that the existing sensory system and its software systems detects the obstacles and measure their location with a noise. Consequently, the behaviour of the sensory system was simply mimicked by adding noise to actual location of the moving obstacles. However, when the real-world experiments are conducted to test the navigation algorithm developed in this research, a colour-based obstacle detection and location measuring systems were used to detect multiple obstacles and measure their locations. These will be further discussed in Section 5.4.2. Also, the sensory system assumes the robot's onboard sensors and localization systems provides the robot's location and velocity. In real-world experiments the used research robot platform is capable of giving its current location with respect to the global axis system through an inherent simple

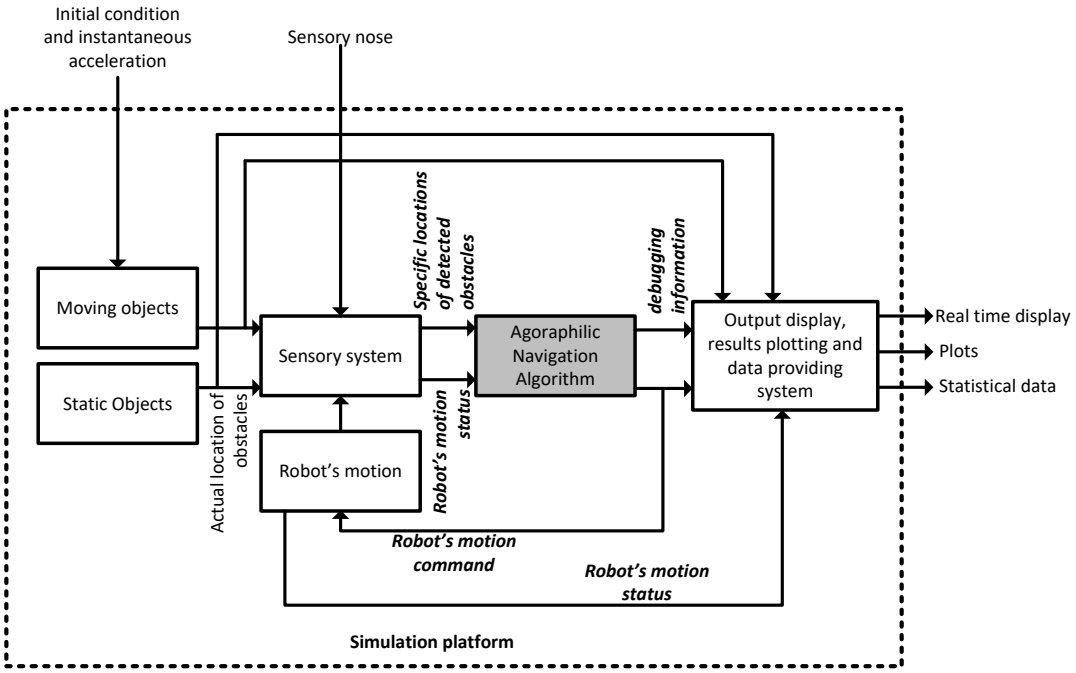


Fig. 4.8 Block diagram of the developed simulation platform to test ANADE

localization method provided by the research robot platform.

In the algorithm running section, Fig. 4.8, the developed ANADE I algorithm was used to navigate a robot in different dynamic environments. The last section of this simulation platform is the output display, results plotting and statistical data providing system. This system shows the actual motion of the moving obstacles, the robot and static obstacles. This system also shows the real time robot's environment, the path of the robot, paths of moving obstacles and other statistical data to analyse the algorithm.

Using the above discussed simulation platform, a number of simulation tests were conducted to test the performance of the ANADE I. Of them, two random tests are presented in Section 4.4.1. Further, the new proposed algorithm was tested and bench-marked against an APF-based algorithm in a similar dynamic environment (see Section 4.4.2). The ANADE I was also tested under a dynamic environment with cluttered MOs to check the limitations of the algorithm (see Section 4.4.3).

4.4.1 Navigation in Dynamic Environment with Two Moving Obstacles

Case 1: A dynamic working environment used in [116] was remodelled and used in this test to validate and compare the performances of the algorithm in a dynamic environment. In this test, the working environment consists of static objects and two moving objects. The position of static and dynamic obstacles and velocities of the dynamic obstacles are initially unknown by the robot. In the simulation, according to [116], the initial coordinates of the robot's position were set to (0, 0), with zero initial velocity. The goal was placed at (100, 100) dm. The preliminary coordinates of MO1 and MO2 are established to (55, 72) dm and (30, 15) dm, respectively. MO1 got a velocity of –2 dm/s in the y-direction. The velocity of MO2 was –2 dm/s towards the x-direction. From this environment landscape, the closest distance between the robot and MO1 occurs when MO1 is at (55, 55) dm position. The minimum distance between the robot and MO2 occurs when MO2 is at coordinates of (25, 15) dm. In both cases, MO1 and MO2 are not close enough to the robot to make a collision. Therefore, in this case, MO1 and MO2 have no influence on the robot to change its path. Therefore, the robot did not change its moving direction from start to goal (see Fig. 4.9).

Case 2: As the next experiment, MO1 and MO2 were placed at (15, 18) dm and (25, 15) dm, respectively, at the start. In this experiment, the robot's starting position and goal locations were kept unchanged. It was observed in this case when MO2 moved from its starting point to (18, 15) dm location that the distance between the robot and MO2 was less than the allowed safety distance. Therefore, MO2 changed the free space around the robot. This influenced the FSH, which led to changing the FSFs in the particular sectors where MO2 was located. As FSFs changed, the resultant force that had pointed to the goal in previous controlled cycles was also changed. After the

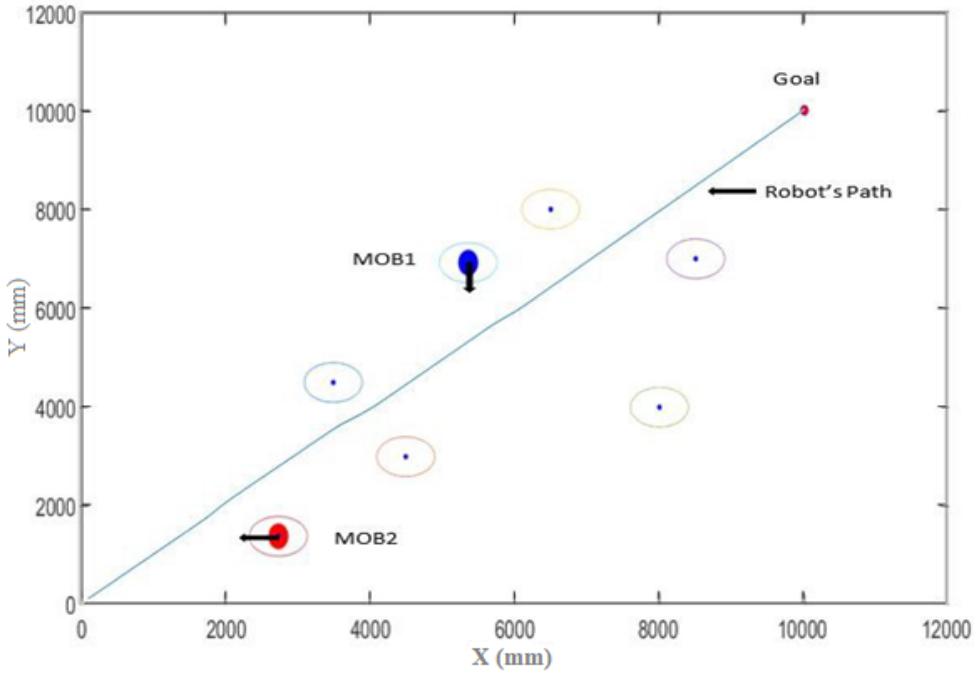


Fig. 4.9 Robot's Path without Influence of Moving Objects (case 1)
Note. MOB1 = moving obstacle 1; MOB2 = moving obstacle 2.

distance between the robot and MO2 was more than the allowed safety distance, the robot started following 'space', which is pointed towards the goal (see Fig. 4.9.). The changes that occurred on the robot's path due to MO2 are shown by section A in Fig. 4.10. MO1 crossed the safety boundary at (55, 53) dm. In this region, MO1 influenced the FSHs, where the sector forces pointing to MO1 become smaller, and the resultant force points to a free space and manoeuvres the robot to the goal without making a collision. These deviations are clearly visible in Fig. 4.10.

These results prove that the modified Agoraphilic algorithm discussed in this chapter has improved the original Agoraphilic algorithm by eliminating one of its major limitations and allows the Agoraphilic algorithm to be used in unknown dynamic environments.

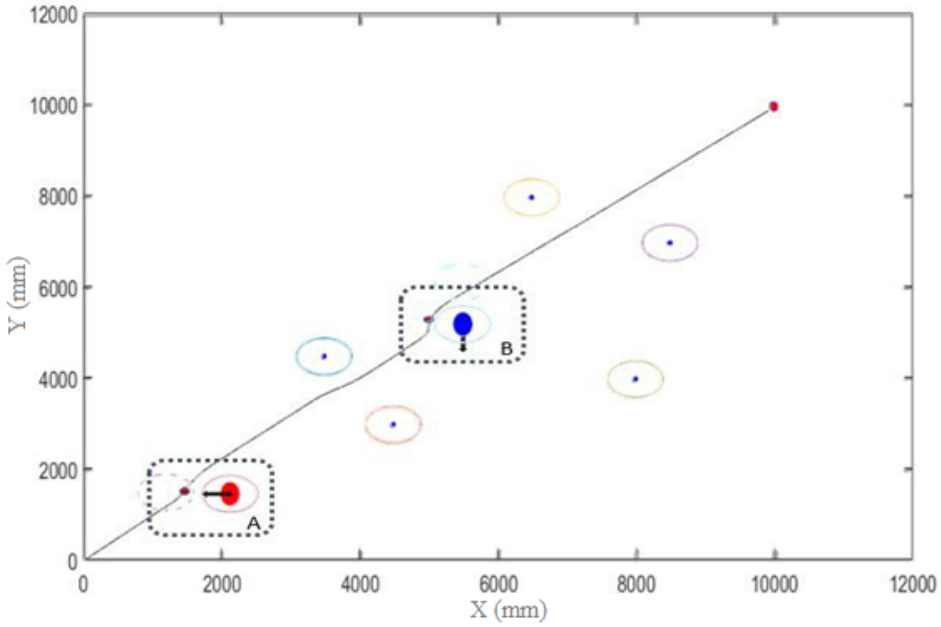


Fig. 4.10 Robot's Path without Influence of Moving Objects (case 2)

4.4.2 Benchmarking of the ANADE I

Two independent simulation runs are shown in Fig. 4.11: one with the new Agoraphilic algorithm and the other with the improved artificial potential field method (IAPFM) proposed in [116]. The performances of both methodologies were compared under the environmental conditions.

Both the algorithms have found a collision-free path to navigate the robot to the goal. However, it could identify some different decisions taken by two algorithms during certain cases. In section A, both algorithms steered the robot almost in the same path, but when MO2 attempted to collide with the robot, it was observed that the robot changed its path to be longer to avoid the obstacle in the IAPFM approach compared to in the new Agoraphilic approach. In section B, it can be observed that the IAPFM approach tries to change its moving direction because of the repulsive forces created by the SOs. Conversely, the new Agoraphilic algorithm produced a path with lesser changes to the moving direction. The Agoraphilic algorithm is an optimistic algorithm

Tab. 4.1 Benchmarking test results of ANADE I

Algorithm	Path length	Number of direction changers
Agoraphilic	148 dm	3
IAPFM	168 dm	5
Comparison	10% shorter (with Agoraphilic)	40% less direction changers (with Agoraphilic)

that finds the space to move instead of avoiding obstacles. Due to this characteristic, the Agoraphilic algorithm manages to minimise the unwanted direction changes in navigation. In section C, when MO1 comes closer to the robot. IAPFM-based algorithms avoid the object by passing the object in its moving direction. This decision taken by the IAPFM is costly, as it has increased the length of the path. In this case, the robot is followed by MO1, and the repulsive force created by MO1 has kept pushing the robot away from the goal. The robot with the new Agoraphilic algorithm faced the same situation, but it made an interesting, human-like decision (see Fig.4.11).

The new Agoraphilic algorithm manoeuvred the robot to move against MO1 and pass it. This made the path shorter (10%) compared to when the IAPFM was used, Tab. 4.1. After this point, the robot moved directly to the goal, but when the IAPFM was used, some direction changes were observed. In the IAPFM, five direction changes were observed, while our algorithm made only three direction changes (40% fewer changes). Therefore, the newly developed Agoraphilic algorithm had fewer interruptions in the navigation to the goal. As the Agoraphilic-based algorithm creates fewer interruptions than does the IAPFM-based algorithm, the robot could move towards the goal with the maximum speed in most sections of the navigation path. As mentioned above, the new Agoraphilic algorithm produced a much shorter and effective path compared to that produced by the IAPFM-based algorithm.

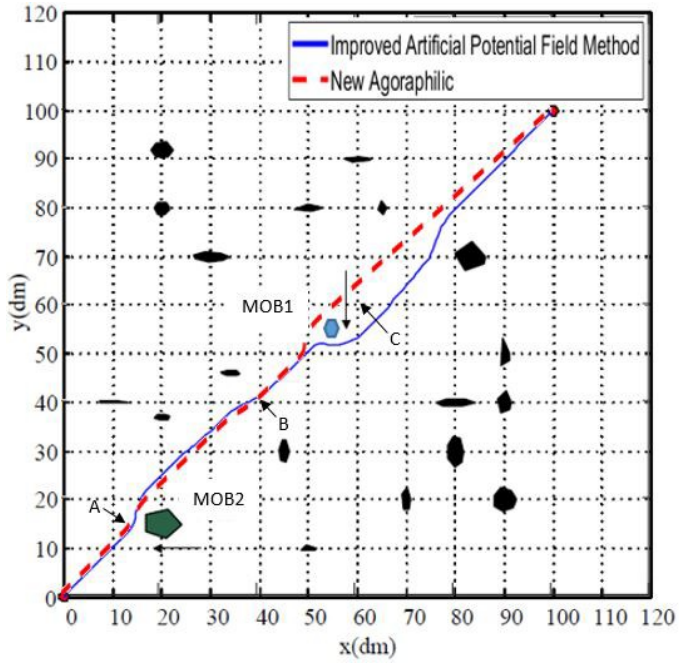


Fig. 4.11 Comparison of Navigation Paths under the Improved Artificial Potential Field Method and the ANADE I

Tab. 4.2 Experimental Conditions of Experiment 2

Object	Initial Position (x,y) (cm)	Initial Velocity (x,y) (cm/s)
Goal	(-200,950)	(0,0) stationary
Robot	(-150,-1000)	(0,0)
Moving Obstacle1 (MO1)	(-500, -200)	(1.5, 1.5)
Moving Obstacle2 (MO2)	(-550,100)	(4.3, 0.4)
Moving Obstacle3 (MO3)	(250, 350)	(-2.0, -0.8)

4.4.3 Limitations of the ANADE I

This experiment was conducted under conditions mentioned in Table 4.2 with three SOs. The aim of the experiment was to test the behaviour of algorithms in dynamically cluttered, complex environments with many MOs challenging the robot at the same time (see Fig. 4.12).

During this experiment, it was observed at around the seventy-fifth iteration that two passages (i.e., Passage 1 and Passage 2) were created by MOs.

Key observations at the seventy-fifth iteration (t = 75 s) are as follows:

Observation 1: Passage 1 has more free space compared to Passage 2.

Observation 2: The free space of Passage 1 is decreasing, and the free space of Passage 2 is increasing (see Fig. 4.13).

At around the seventy-fifth iteration, the ANADE I pulled the robot towards Passage 1. The ANADE I chose this option because of Observation 1, as outlined above. At around the one hundred and tenth iteration (t = 110 s), the states of Passage 1 and Passage 2 had changed.

Key observations at the one hundred and tenth iteration (t = 110 s) are as follows:

Observation 3: Passage 1 has more free space compared to Passage 2.

Observation 4: The free space of passage 1 is decreasing, and free space of Passage 2 is increasing (Fig. 4.14).

At around the one hundred and tenth iteration, the ANADE I kept pulling the robot towards Passage 1 because of Observation 3, as outlined above. However, the intelligent decision should be to change the robot's direction towards Passage 2, although it has less free space at that moment. As a result of these decisions, the robot controlled by the ANADE I became trapped in Passage 1 and collided with MO3.

The key findings of this experiment are as follows:

1. In dynamic cluttered environments, the ANADE I may fail to navigate robots to the goal safely.

2. The algorithm could be more effective in dynamic environments if it can make decisions based not only on current free space but also on future growing and diminishing free space.

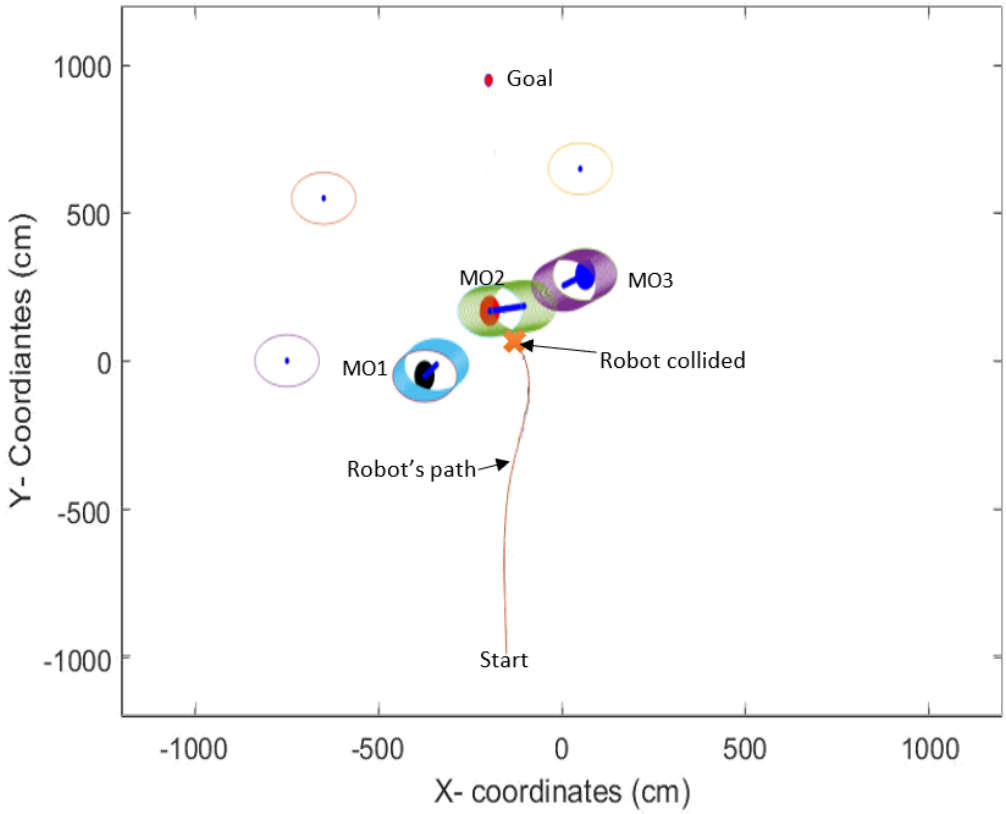


Fig. 4.12 The Robot's Path with Three Moving Obstacles Challenging the Robot at the Same Time

Note. Different coloured (blue, green and purple) multiple circles depicted in the figure are the movement paths of MO1, MO2 and MO3

4.5 Summary

The ANADE I developed in this research was proven to successfully navigate mobile robots in dynamic environments. The ANADE I is a free space attraction-based optimistic algorithm. It uses only attractive forces created by the current free space in the robot's surrounding environment. The new algorithm drives the robot through current free spaces, leading it to the goal. This new proposed methodology was able to overcome the main limitation of the original Agoraphilic algorithm (i.e., that the algorithm is limited to static environments) and keep its advantages in static environments.

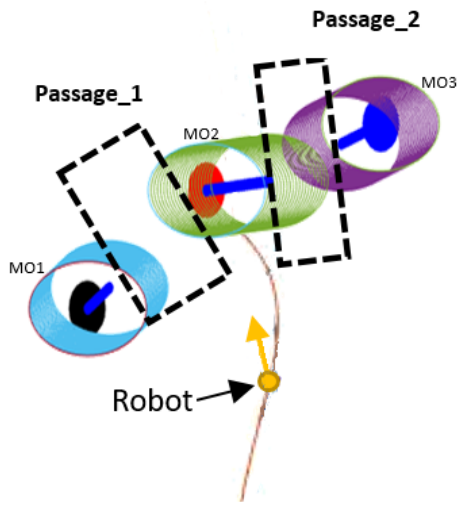


Fig. 4.13 The Robot's Surroundings at around t = 75 s with a Dynamically Cluttered Environment

Note. Different coloured (blue, green and purple) multiple circles depicted in the figure are the movement paths of MO1, MO2 and MO3

Further, this chapter presented a comparative analysis with currently published work on robots' navigation using the improved artificial potential field-based method. The conducted comparative study proved that due to its 'optimistic' decision-making approach, the new ANADE I is able to navigate robots in a path that is approximately 10% shorter and with 40% fewer direction changes than when the improved artificial potential field-based algorithm is used. Therefore, it can be concluded that the ANADE

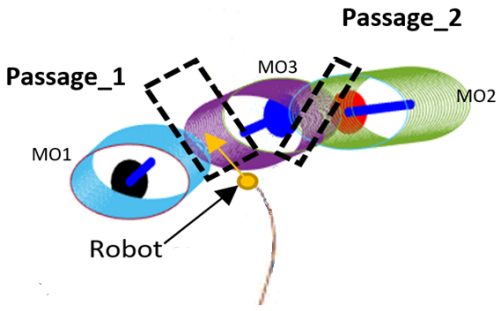


Fig. 4.14 The Robot's Surroundings at around t = 110 s with a Dynamically Cluttered Environment

Note. Different coloured (blue, green and purple) multiple circles depicted in the figure are the movement paths of MO1, MO2 and MO3

I is a successful method to use to navigate robots in dynamic environments.

However, the ANADE I was designed only to navigate robots in dynamic environments where multiple MOs do not challenge the robot at the same time. The ANADE I does not use any obstacle path prediction method. As mentioned in Chapter 2, most of the navigation algorithms fail to adopt the velocities of MOs for its decision-making. The ANADE I also suffers from this common weakness. Consequently, the ANADE I considers only the current free space around the robot in its decision-making. However, when the environment is complex and challenging, decision-making that is based only on the current environment is not sufficient. The robot may not reach the goal safely and efficiently. This was verified by the simulation experiment conducted and presented in Section 4.3.3. However, this problem can be overcome by improving the algorithm. Chapter 5 presents the ANADE II, which makes decisions based on current and future growing and diminishing free spaces.

5

Chapter 5: Agoraphilic Navigation Algorithm in Dynamic Environment with Tracking and Prediction (ANADE II)

5.1 Introduction

A new mobile robot navigation algorithm, the ANADE II, is proposed in this chapter. The ANADE II is a natural extension of the ANADE I that primarily focuses on overcoming the challenges associated with the ANADE I when tackling multiple MOs attacking the robot at the same time. These challenges were explained in detail in Section 4.4.

The ANADE II incorporates a predictive module to predict future free spaces. This chapter proposes a new architecture to allow Agoraphilic-based navigation algorithms to accommodate short-term predictions in decision-making. The new algorithm (ANADE II) does not look for obstacles to avoid but instead looks for current free space leading to future growing free space passages towards the goal. This also follows the fundamental characteristic of the FSA (Agoraphilic) concept. As the proposed algorithm also uses the FSA concept, it could eliminate some of the common issues in general navigation approaches [28].

1. This stops the robot getting oscillated in narrow corridors.

2. The robot can reach the goal even when there is a large obstacle behind the goal.

3. The algorithm allows the robot to pass through narrow corridors.

Further, the tracking module is used effectively by the new algorithm to estimate not only the current location accurately but also the velocity of MOs. Consequently, the novel algorithm has added advantages, such as:

1. knowing the accurate locations of MOs

2. being able to make decisions based on the velocities of MOs.

The new algorithm also predicts the future environments with respect to the global axis system and calls them Future Global Maps (FGMs). Main submodules run in the prediction stage to generate future driving forces. These future driving forces that are generated from the novel FGM concept influence the robot's current driving force to navigate the robot successfully in challenging environments.

5.2 The Architecture of the ANADE II

A new modular-based architecture was used to incorporate short-term predictions involved in the ANADE II. This architecture increases the flexibility of improving the algorithm's modular vice without performing major modifications to other modules. The ANADE II comprises the following eight main modules:

1. Sensory Data Processing (SDP) module

2. Dynamic Obstacle Tracking (DOT) module

3. Dynamic Obstacle Position Prediction (DOPP) module

4. Current Global Map (CGM) generation module.

5. Future Global Map (FGM) generation module.

6. Free Space Attraction (FSA) module.

    (a) current and future FSH generation module

    (b) FSF generation module

    (c) force-shaping module

    (d) Instantaneous Driving Force Component (IDFC) generation module

7. IDFC weighting module.

8. Robot's Motion Command (RMC) generation module

The SDP module collects sensory data from the sensors mounted on the robot (RealSense^{TM} LiDAR Camera, encoders and associated equipment) and transforms all robot-centric data to a world reference frame (global positioning system). The tracking module then generates two outputs, a set of locations of MOs and a set of velocity values of MOs within the map using the globalised sensory data (see Fig. 5.1). Subsequently, the algorithm is divided into two sections, as follows (see Fig.5.2):

1. the IDFC (F_c) generation for the CGMs (CGM-F_{c1})

2. the IDFC (F_c) generation for FGM-F_{c2}, F_{c3}, ..., F_{cN} (Prediction)

The IDFC generation for the CGM-F_{c1} section of the algorithm uses the current environment of the robot (the CGM) and generates the component force (F_{c1}) based on the current surroundings of the robot. In this process, the CGM becomes the input for the FSH generation module. The generated FSH is then converted to a set of FSFs by

the FSF generation module. Subsequently, the force-shaping module regulates the FSFs such that the robot will move towards the goal. Finally, the IDFC generation module generates the final driving force component (F_c) for the CGM using the shaped FSFs.

The second output of the tracking module (the current states of MOs) is used as the input for the dynamic objects position prediction module. This returns N-1 FGMs, where N is the final prediction of the DOPP module. Each FGM is input into the FSH generation module. The generated FSH is then converted to a set of predictive FSFs by the FSF generation module. Subsequently, the force-shaping module shapes the FSFs such that the robot will move towards the goal. Then, the IDFC generation module generates the final force component (F_c) for the corresponding FGM. This process occurs for each FGM.

Finally, all the IDFCs (F_{c1}, F_{c2}, ..., F_{cN}) related to the CGMs and FGMs are input into the IDFC weighing module. This module generates the final driving force (the robot's actual driving force) for the current iteration (see Fig. 5.1). This process repeats for each iteration. The behaviour of the ANADE II is further explained in Section 5.2.1.

5.2.1 Main Steps of the ANADE II

Step 1: Start the algorithm

Step 2: Sensory data are taken from the robot's sensory system and fed into the SDP module.

Step 3: The SDP module converts all the sensory data to a global axis system.

Step 4: The tracking module estimates the location and velocity of moving objects.

Step 5: Based on the current locations of MOs (one output of the tracking module) and

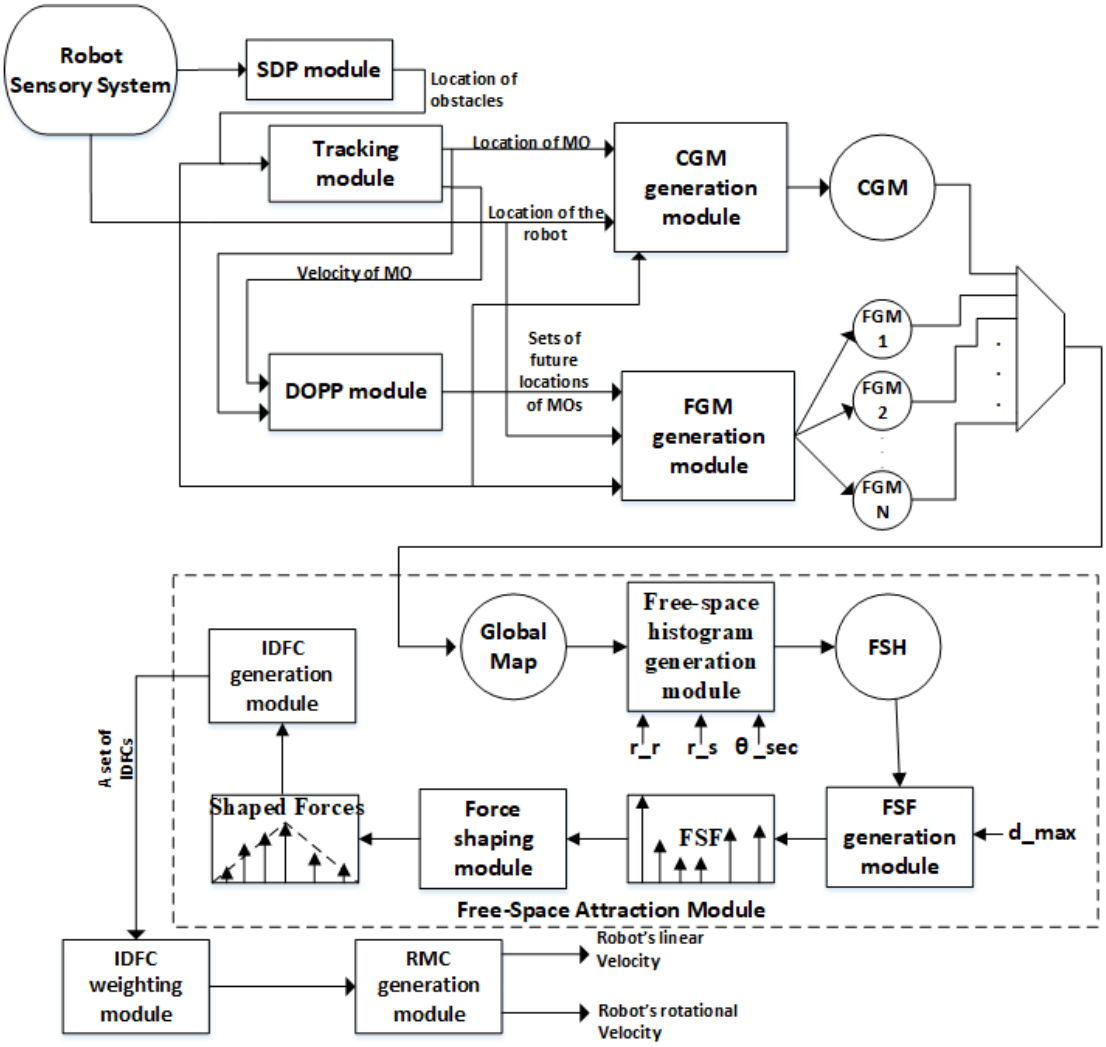


Fig. 5.1 Block Diagram of the ANADE II

Note. SDP = Sensory Data Processing; MO = moving obstacle; CGM = current global map; DOPP = Dynamic Obstacle Position Prediction; FGM = future global map; FGM N = ; IDFC = instantaneous driving force component; FSH = Free Space Histogram; FSF = Free Space Force; RMC = robot's motion command.

locations of SOs (from the sensory module), the CGM is generated.

Step 6: Based on the estimated locations and velocities of MOs, the location of SOs and the robot's states (location and velocity) surrounding the robot in future iterations are predicted. The predicted future environments are generated (FGMs). (After Step 6, there is one CGM and multiple FGMs. Each of these maps goes through Steps 6–10.)

Step 7: The free space passages in a global map are identified by the FSH generation

module. The global map is also converted into an FSH (see Fig. 5.2).

Step 8: The generated FSH is converted to a set of FSFs by the FSF generation module.

Step 9: The force-shaping module focuses the FSFs towards the goal.

Step 10: The set of shaped forces are fed into the IDFC generation module. This module produces one single force for the particular global map known as the IDFC ($F_{c,N}$).

Step 11: All the IDFCs (F_{c1}, F_{c2}, ..., F_{cN}) are fed into the IDFC weighing module. This creates the instantaneous driving force for the current iteration. This is the robot's actual driving force.

Step 12: If the robot has reached the goal, go to Step 12. Repeat Steps 2–12, inclusive.

Step 13: Stop

The development of the main modules of the ANADE II is discussed in Section 5.3.

5.3 Main Modules in the ANADE II

The ANADE II imports the FSA module from the ANADE I, with minor modifications. The new proposed algorithm in this chapter also uses the tracking module used in the ANADE I, with minor modifications. Therefore, in this section, the modifications made to the tracking module are discussed. Further, the development process of the other novel modules is discussed in this section.

5.3.1 Sensory Data Processing Module

The ANADE II is validated via experimental tests. Therefore, this new SDP module had to be introduced. The SDP module takes sensory data from the robot's sensory system

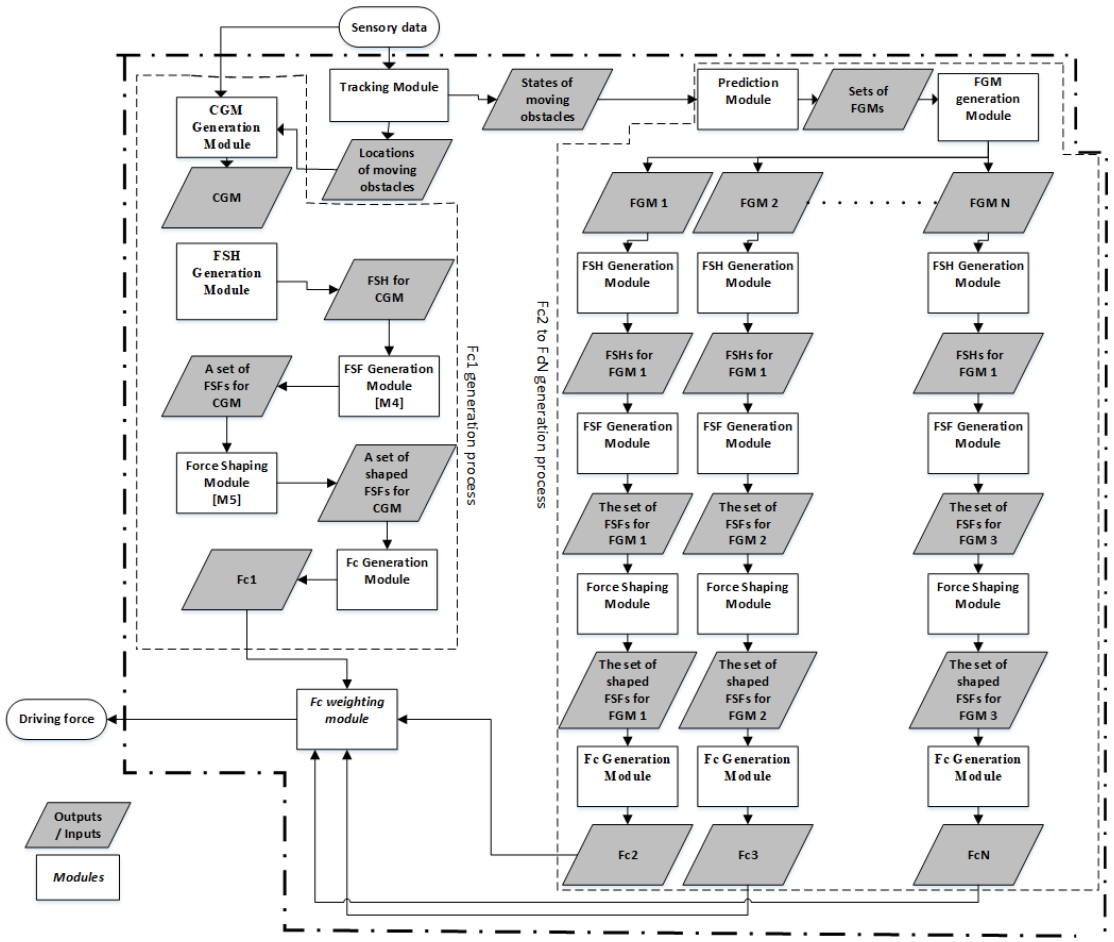


Fig. 5.2 Main Steps of the Robot's Driving Force Generation of the ANADE II
Note. CGM = current global map; FGM = future global map; FSH = Free Space Histogram; FSF = Free Space Force; Fc = force component

as its input. There are two sensory systems used to capture the robot's environment:

1. 360^0 LiDAR sensor

2. RealSense^{TM} camera (depth camera)

The SDP module receives LiDAR data as a point cloud. This point cloud is generated with respect to the robot's coordinate system. The red, green and blue data captured by the RealSense^{TM} Camera and the point cloud are pre-processed by a separate system (further discussed in Sections 5.4). This system runs an image processing algorithm and combines depth information to provide the locations of obstacles with respect to the

robot's axis system. All data received with respect to this system are converted to the world reference frame through Eq. 5.1.

$$\begin{bmatrix} Xg \\ Yg \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \times \begin{bmatrix} Xr \\ Yr \end{bmatrix} + \begin{bmatrix} Xrg \\ Yrg \end{bmatrix} \tag{5.1}$$

where:

$\theta \stackrel{\text{def}}{=}$ orientation of the robot

$(Xg, Yg) \stackrel{\text{def}}{=}$ coordinates with respect to the world reference frame

$(Xr, Yr) \stackrel{\text{def}}{=}$ coordinates with respect to the robot's axis system

$(Xrg, Yrg) \stackrel{\text{def}}{=}$ robot's current position with respect to the world reference frame

5.3.2 Dynamic Obstacle Tracking Module

In the ANADE II, the object tracking module takes positions of obstacles from sensors as its input. The module runs the same KF-based algorithm discussed in Chapter 4 but generates two outputs (whereas the tracking module used in Chapter 4 provides just one output, which is a set of locations of moving objects; see Fig. 5.3):

1. the estimated locations of MOs
2. the estimated velocities of MOs.

These two outputs are fed into the dynamic obstacle path prediction module as well as into the CGM generation module.

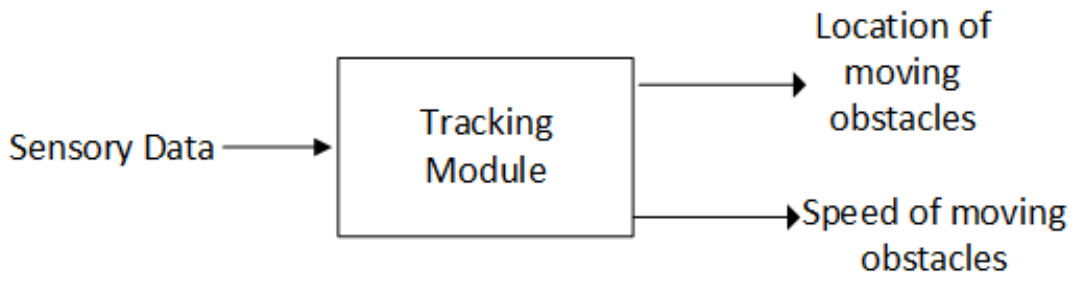


Fig. 5.3 The Input and Outputs of the Dynamic Obstacle Tracking Module

5.3.3 Dynamic Obstacle Path Prediction Module

The navigation algorithm identifies future growing or diminishing free spaces with the assistance of the DOPP module. The DOPP module estimates future states (location and velocity) of MOs in the robot's environment.

The DOPP module uses globalised sensory data (the sensory data captured by the robot are transformed to the global frame as discussed in Section 5.3.1) and information about the current states of MOs from the tracking modules as its primary inputs. The DOPP module then forecasts the robot's anticipated environmental configuration for future iterations and predicts sets of future locations of MOs (see Fig. 5.4).

These future location sets of MOs are the outputs of the DOPP module (if the prediction is performed for t + n iterations, there will be n – 1 future locations in one set). Future states (positions and velocities) of MOs are predicted using the prediction model described in Eq. 5.2 (equation is defined in the global fram). The predictions are updated at each iteration, and new predictions are made according to the updated state data.

$$\begin{bmatrix} x(t+n) \\ y(t+n) \\ dx(t+n)/dt \\ dy(t+n)/dt \end{bmatrix} = \begin{bmatrix} 1 & 0 & nT & 0 \\ 0 & 1 & 0 & nT \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x(t) \\ y(t) \\ \frac{dx(t)}{dt} \\ \frac{dy(t)}{dt} \end{bmatrix} + \begin{bmatrix} \frac{(nT)^2}{2} \times \frac{d^2x(t)}{dt^2} \\ \frac{(nT)^2}{2} \times \frac{d^2y(t)}{dt^2} \\ nT \times \frac{d^2x(t)}{dt^2} \\ nT \times \frac{d^2y(t)}{dt^2} \end{bmatrix}$$

(5.2)

where:

$x(t) \stackrel{\text{def}}{=}$ position in x direction

$y(t) \stackrel{\text{def}}{=}$ position in y direction

$T \stackrel{\text{def}}{=}$ sample period

$n \stackrel{\text{def}}{=}$ sample number

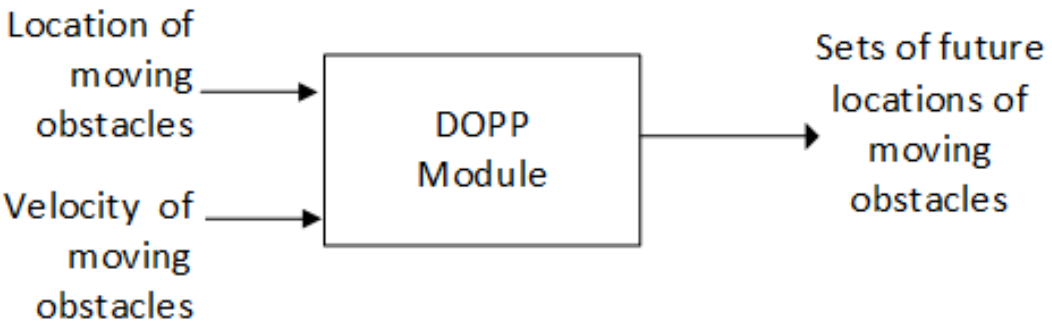


Fig. 5.4 The Input and Outputs of the Dynamic Obstacle Path Prediction Module

5.3.4 Current Global Map Generation Module

The CGM generation module generates a map of the robot's environment with respect to the world axis system. This module takes three main inputs (see Fig. 5.5):

1. the locations of SOs with respect to the robot's axis system from the sensory system
2. the current locations of MOs from the MO tracking module
3. the current location of the robot from the sensory system

The CGM generation module processes these three inputs and creates a map of the environment with respect to the global axis system. This map is called the CGM. Consequently, this CGM is fed into the FSA module.

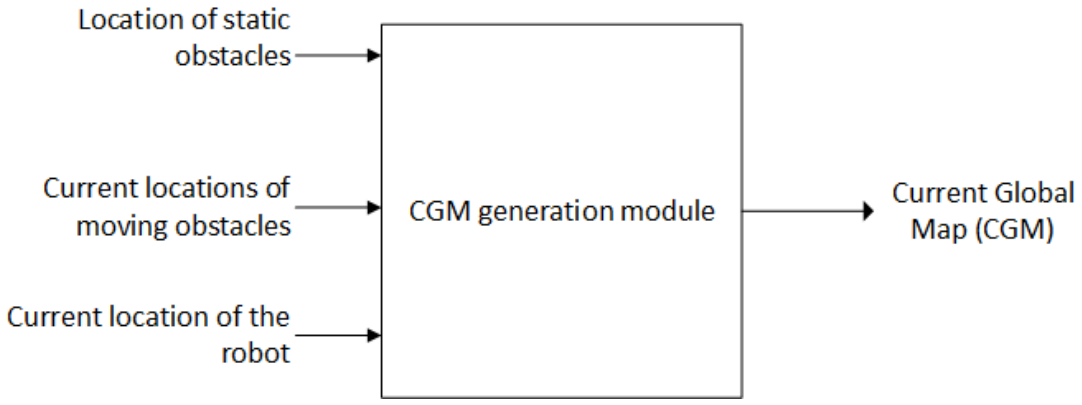


Fig. 5.5 The Inputs and Output of the Current Global Map Generation Module

5.3.5 Future Global Map Generation Module

The main tasks of the FGM generation module are to forecast the robot's future locations, to forecast future environment set-ups and to generate a set of maps called FGMs. This module takes three main inputs (see Fig. 5.6):

1. the current location of the robot (taken from the sensory system)

2. sets of future locations of MOs (taken from the obstacle path prediction module)

3. a map of the locations of SOs with respect to the global axis system.

The FGM at iteration 'N' is generated by combining the estimated location of the robot at nth iteration with forecasted locations of MOs, SOs and the goal at the n^{th} iteration. If the FGM generation module develops predicted maps until t + n iteration, there will be n – 1 FGMs. The FGMs are updated at each iteration with corrections made to position estimations.

The robot's future locations are derived by Eq. 5.3 (equation is defined in the global fram). These FGMs are used for future FSH creation.

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ \theta(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} + \begin{bmatrix} \dot{x}_k \\ \dot{y}_k \\ \dot{\theta}_k \end{bmatrix} n \times T \tag{5.3}$$

where:

(x, y) : the actual position coordinates.

θ : orientation of the robot.

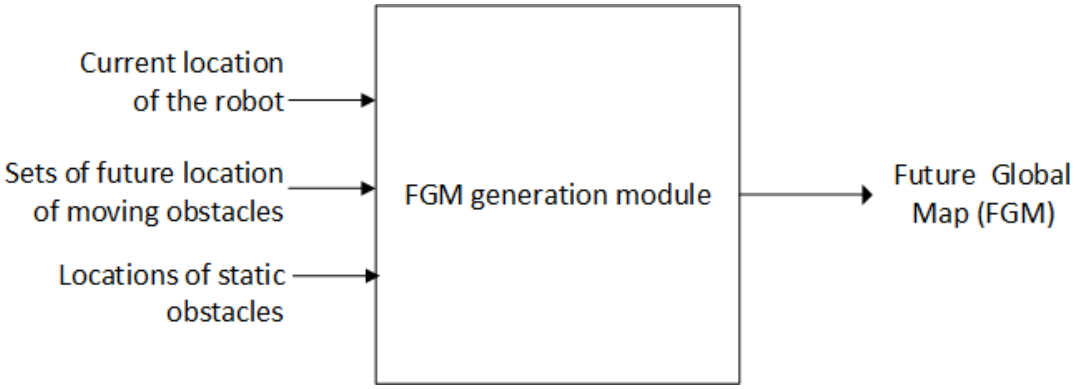


Fig. 5.6 The Inputs and Output of the Future Global Map Generation Module

5.3.6 Free Space Attraction Module

The FSA module used in the ANADE II is slightly different from that used in the ANADE I. The FSH generation module, FSF generation module and the force-shaping module used in the ANADE II are identical to those used in the ANADE I. However, the force simulation module used in the ANADE I is replaced by the new IDFC generation module.

5.3.7 Instantaneous Driving Force Component Generation Module

The IDFC generation module takes a set of shaped FSFs as its input and generates an IDFC (F_c) as its output (see Fig. 5.7). The IDFC is the maximum shaped FSF in the set of shaped FSFs. This keeps the upper bound of F_c to 1, which is expected by the IDFC weighting module.

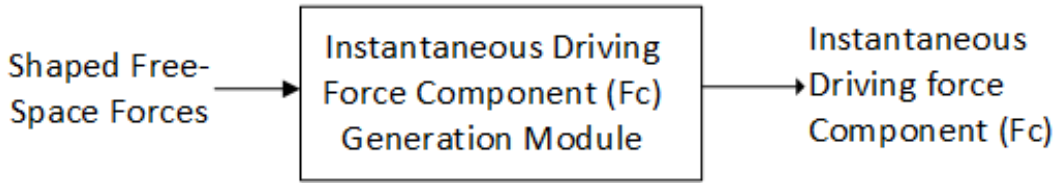


Fig. 5.7 The Input and Output of the Instantaneous Driving Force Component Generation Module

5.3.8 Instantaneous Driving Force Component Weighing Module and Final Robot's Driving Force

This module takes all the IDFCs (F_{c1} to F_{cN}) as its input (see Fig. 5.8). A fixed liner weighting system is used in this module to scale the F_cs. The weighing is done according to the accuracy of the predicted FGM (F_c derived for the CGM gets the highest weight, as there is no prediction involved in CGM; F_c derived for the (n – 1)th FGM gets the lowest weight). The weighted average of the instantaneous driving forces is taken as the final driving force (F_d) of the current iteration, which drives the robot.

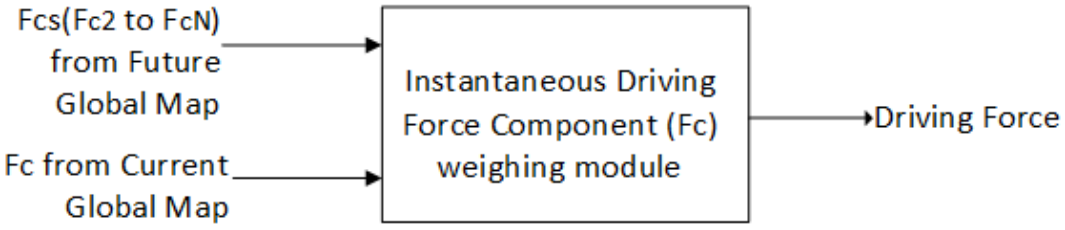


Fig. 5.8 The Input and Output of the Instantaneous Driving Force Component Weighing Module

5.3.9 Robot's Motion Command Generation Module

In each iteration, the ANADE II provides a single accretive force, which is capable of pulling the robot through current free space towards the future growing free space leading to the goal. However, this force is generated assuming the robot is a particle in the space. Therefore, when a real robot is used, the final driving force needs to be converted to linear and rotational velocities of the robot (see Fig. 5.9). This convention depends on the kinematic model of the robot platform used.

In this research work, the TurtleBot3 Waffle Pi (TB3 Waffle Pi) research platform was used. The developed kinematic model for the TB3 Waffle Pi is shown in Eq. 5.4.

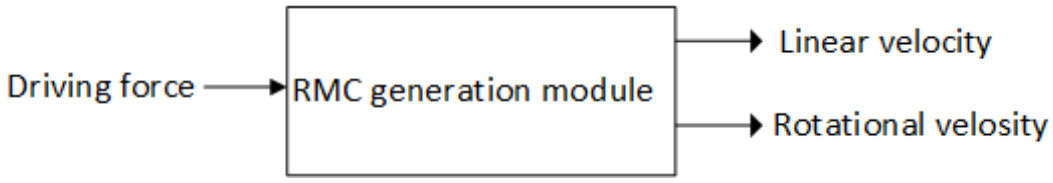


Fig. 5.9 The Input and Output of the Robot's Motion Command Generation Module

The robot's location and orientation are given by (X, Y) coordinates and θ (see Fig. 5.10).

The following variables are defined as:

$V_R \stackrel{\text{def}}{=}$ linear velocity of the right wheel

$V_L \stackrel{\text{def}}{=}$ linear velocity of the left wheel

$w \stackrel{\text{def}}{=}$ angular velocity of the mobile robot

$(x, y) \stackrel{\text{def}}{=}$ the actual position coordinates

$\theta \stackrel{\text{def}}{=}$ orientation of the robot

The kinematic model is given by Eq. 5.4:

$$\dot{x} = \dot{\mathrm{x}}_r \times cos(\theta)$$

$$\dot{y} = \dot{\mathrm{x}}_r \times sin(\theta) \tag{5.4}$$

where:

$$\dot{x}_r = \frac{v_L + v_R}{2}$$

$$\dot{\theta} = \frac{v_L - v_R}{d}$$

and where (X, Y and θ) are the robot's actual position and orientation angle relative to the world reference frame. In the prediction module, this model is used to estimate the robot's states. To drive the TB3 Waffle Pi, the ANADE provides $\dot{x}_r$ and for each iteration (see Equations 5.5 and 5.6).

$$\dot{x}_r = v_i \times F_d \times cos(\theta) \tag{5.5}$$

$$\dot{\theta} = \theta_i \times F_d \times sin(\theta) \tag{5.6}$$

where:

$F_d \stackrel{\text{def}}{=}$ Final driving force from the ANADE II

$V_i \stackrel{\text{def}}{=}$ Learner velocity tuning parameter

5.4 Experimental Testing and Analysis of Results

Two simulation experiments and four real experiments are discussed in this section. These experiments were designed to test the algorithm and verify the importance of the new prediction module.

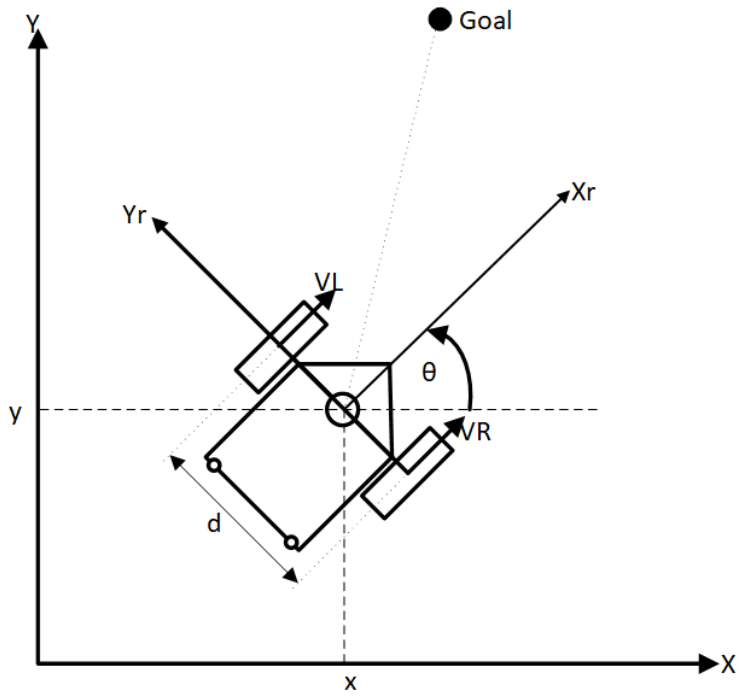


Fig. 5.10 Robot model

Note. Yr = robot's Y axis; VL = linear velocity of the left wheel; Xr = robot's X axis; VR = linear velocity of the right wheel; = orientation of the robot; d = width of the robot.

5.4.1 Simulation Test

Experiments 1 and 2 was conducted with prediction and without prediction under exactly the same dynamic environments. The simulation platform discussed in Section 4.4 was used to conduct the simulation experiments. The developed simulation platform allows the user to create and locate SOs, the goal and MOs in a desired two-dimensional environment. The velocities and accelerations of MOs can be adjusted as needed. Further, users can add random noises to the acceleration of MOs. The developed simulation platform also modelled a sensory system to identify the locations of obstacles. The user can vary the noise level of the sensory system. Therefore, the developed simulation platform was capable of generating random dynamic environments.

5.4.1.1 Experiment 1: Navigation in Dynamic Environment with Three Moving Obstacles

This experiment was conducted under dynamic environment with three MOs and three SOs. In this experiment, MOs challenged the robot independently. The experimental conditions are summarised in Table 5.1.

Fig. 5.11 shows the robot's path under those conditions with the ANADE I and the ANADE II. Results are presented with a comparison to the ANADE I.

Tab. 5.1 Experimental Conditions of Experiment 1

Object	**Initial position (x,y) (cm)**	**Initial velocity (x,y) (cm/s)**
Goal	(-200,950)	(0,0) stationary
Robot	(-250,-700)	(0,0)
Moving obstacle1	(-500,-200)	(1.0, 0.6)
Moving obstacle2	(-550,100)	(1.9, 0.4)
%hline Moving obstacle3	(250,200)	(-5.0,1.9)

Experiment with ANADE I: In Area3_2, the robot has engaged with MO2, which moved towards the x-direction. Initially, the robot moved further to the (+x, +y) direction because the instantaneous free space in this direction was larger. As a result, the robot moved close to MO2. The robot had to stop (see Figure 9) and take a sharp turn to avoid the collision.

Experiment with ANADE II: In contrast to the above, the algorithm with the prediction of MO positions initially manoeuvred the robot towards the (–x, +y) direction. As a result, the robot could pass MO2 without stopping and performing sharp turns. In Area3_3, the robot with prediction went towards the goal while increasing speed without changing direction, as it has knowledge of the future free space.

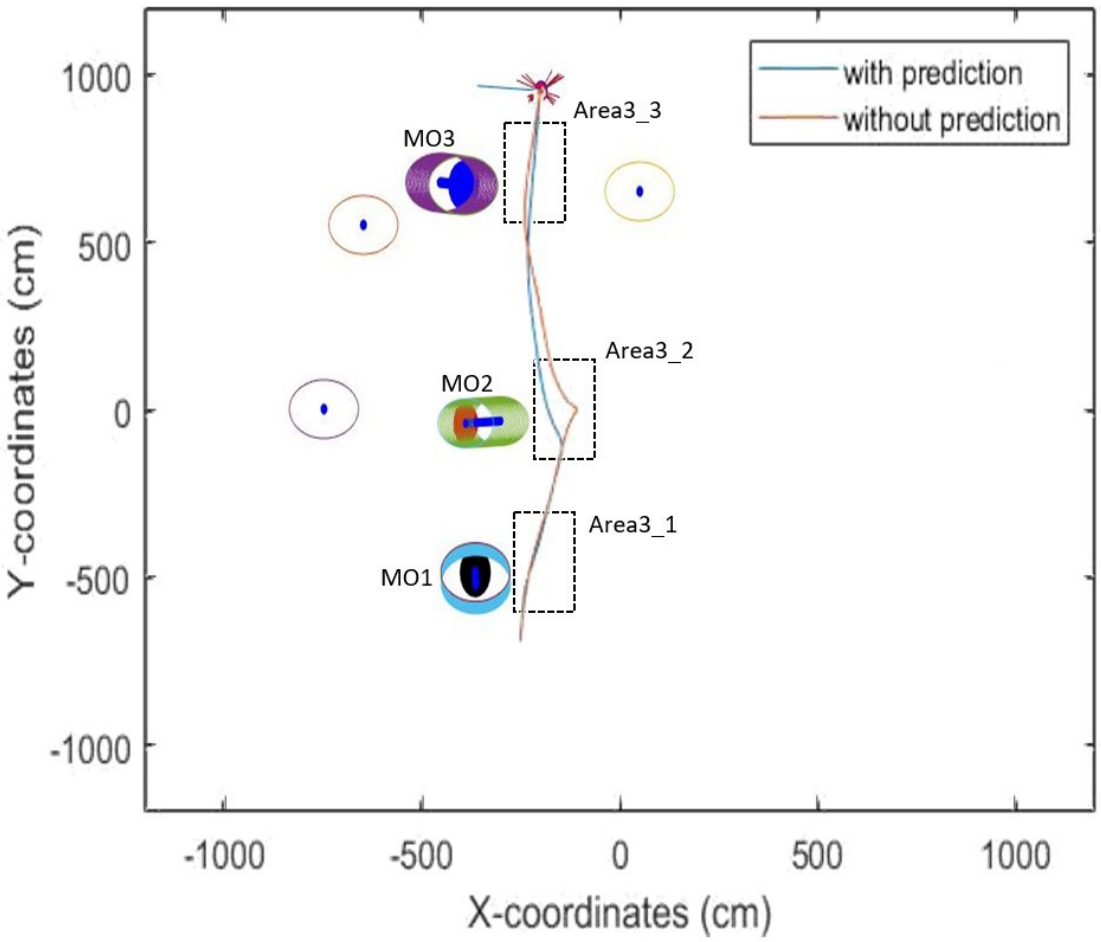


Fig. 5.11 Effect of Prediction on Mobile Robot Path with Three Moving Obstacles

During this task, the robot with prediction took 165 s and travelled around 1666 cm. Conversely, the robot without prediction took 185 s and travelled 2456 cm.

The key findings of this experiment are as follows:

- The algorithm developed for prediction has allowed the robot to reach the goal by maintaining smooth speed variations (see Fig. 5.12) and taking shorter paths more quickly than the algorithm without prediction.
- The algorithm developed for prediction allowed the robot to make prior decisions when it engages with MOs.
- The algorithm developed for prediction stopped the robot from making sharp

turns.

- The algorithm developed to predict MOs' position reduced the path length of the ANADE significantly.

From the comparative results shown in Table 5.2, the ANADE with the prediction module reached the goal 10% earlier using a 32% shorter path than did the ANADE without prediction. During the experiment, the ANADE with prediction never stopped the robot, whereas the ANADE without prediction did.

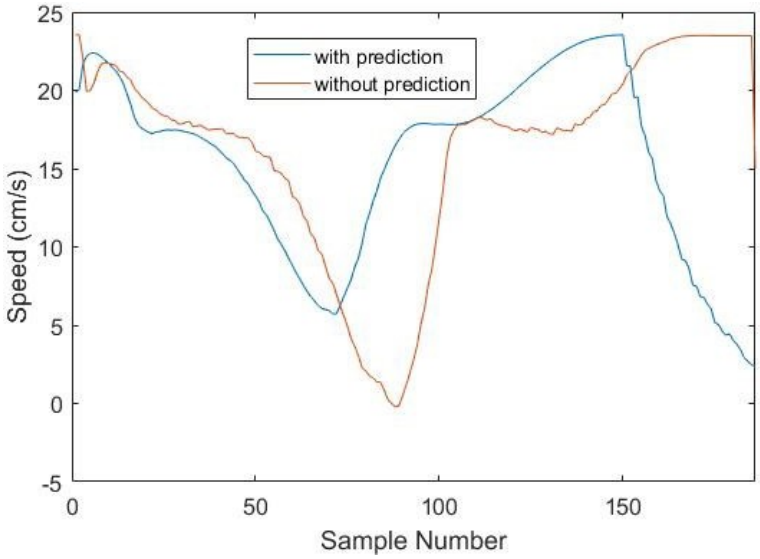


Fig. 5.12 Effect of Prediction on the Robot's Speed Variations in Dynamic Environment with Multiple Moving Obstacles

Tab. 5.2 Comparative Results of Experiment 1

Parameters	With the ANADE I	With the ANADE II
Path length (cm)	2456	1666
Time taken (s)	185	165
Avg. speed (cm/s)	13	10
Min. speed (cm/s)	0	5.7
Max. speed (cm/s)	23.5	23.5

5.4.1.2 Experiment 2: Dynamic Environment with Cluttered Moving Obstacles

This experiment was conducted under the conditions outlined in Table 5.3, with three SOs. The aim of the experiment was to test the behaviour of the ANADE II under dynamically cluttered complex environments with many MOs challenging the robot at the same time (see Fig. 5.13). In this experiment, the results are presented with a comparison to the ANADE I. The experimental conditions are similar to those in the experiment presented in Section 3.3.3.

Tab. 5.3 Experimental Conditions of Experiment 2

Object	Initial position (x,y) (cm)	Initial velocity (x,y) (cm/s)
Goal	(-200,950)	(0,0) stationary
Robot	(-150,-1000)	(0,0)
Moving obstacle1	(-500,-200)	(1.5, 1.5)
Moving obstacle2	(-550,100)	(4.3,0.4)
Moving obstacle3	(250,350)	(-2.0,-0.8)

During this experiment, it was observed that at around the seventy-fifth iteration, two passages (i.e., Passage_1 and Passage_2) were created by MOs.

Key observations at the seventy-fifth iteration (t = 75 s) are as follows:

- Observation 1: Passage_1 has more free space compared to Passage_2
- Observation 2: The free space of Passage_1 is increasing, and the free space of Passage_2 is decreasing (see Fig. 5.14)

At around the seventy-fifth iteration, the algorithms with and without prediction pulled the robot towards Passage_1. The algorithm without prediction chose this option because of Observation 1, as outlined above. Conversely, the algorithm with prediction chose this option because of Observation 2.

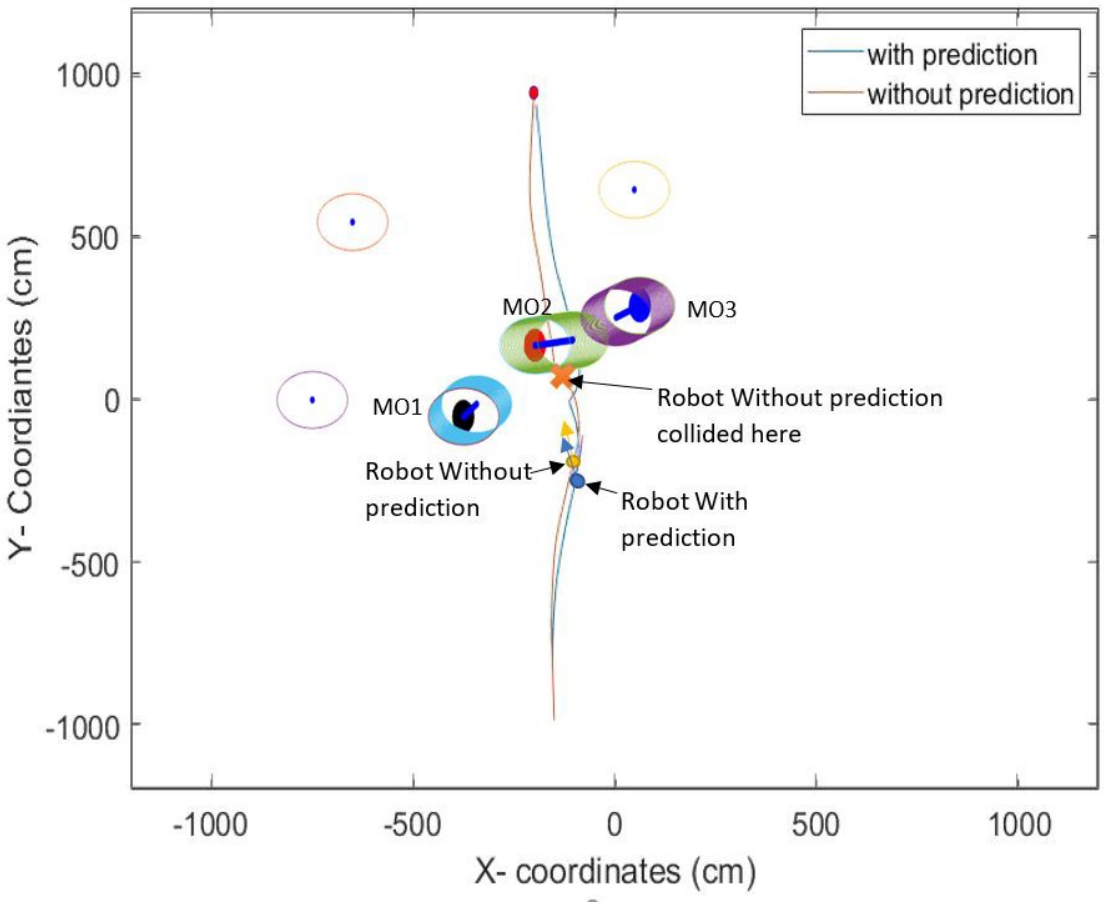


Fig. 5.13 Effect of Prediction on Mobile Robot Path with Three Moving Obstacles Challenging the Robot at the Same Time

At around the one hundred and tenth iteration (t = 110 s), states of Passage_1 and Passage_2 changed.

Key observations at the one hundred and tenth iteration (t = 110 s) are as follows:

- Observation 3: Passage_1 has more free space compared to Passage_2
- Observation 4: The free space of Passage_1 is decreasing, and the free space of Passage_2 is increasing (Fig. 5.15)

At the one hundred and tenth iteration, the algorithm without prediction kept pulling the robot towards Passage_1 because of Observation 3, as outlined above. However, the algorithm with prediction chose to change the robot's direction and move towards

Passage_2, although it had less free space at that moment. This decision was made because of the future position prediction of the MO (Observation 4). As a result, the robot that was controlled without prediction became trapped in Passage_1 and collided with MO3. Conversely, due to the decision made at the one hundred and tenth iteration, the robot with prediction could navigate safely through Passage_2 to the goal.

The key findings of this experiment are as follows:

- Prediction helps the robot to identify future free spaces.
- In dynamic cluttered environments the prediction methodology guides the robot away from traps.

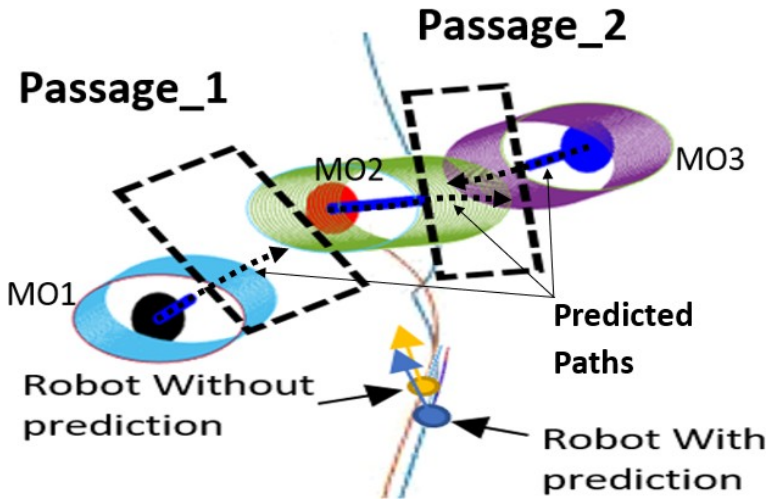


Fig. 5.14 The Robot's Surroundings at around t = 75 s with Dynamically Cluttered Environment

5.4.2 Real World Experiments

The experimental work was conducted to demonstrate and validate the performance of the new algorithm. The experimental platform included a TB3Waffle Pi research robot. The TB3 Waffle Pi uses the Robot Operating System standard platform. It comprises a 360° LiDAR sensor, a signal board computer (robot's PC) that runs with Ubuntu, an

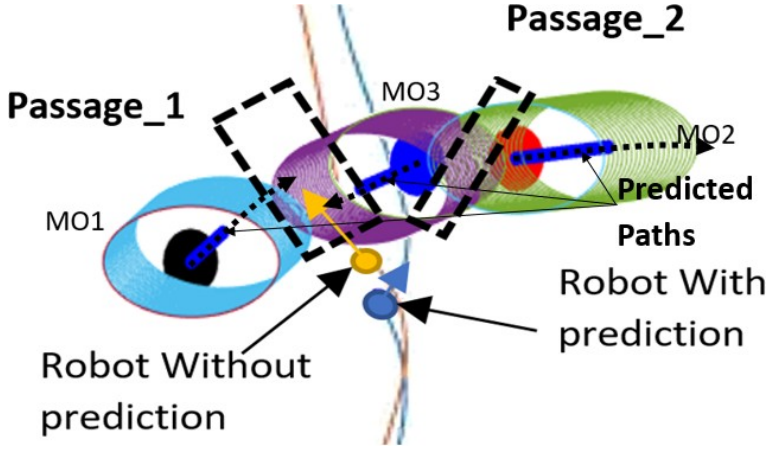


Fig. 5.15 The Robot's Surroundings at around t = 110 s with Dynamically Cluttered Environmentt

OpenCR 32-bit ARM controller, a Bluetooth module and a Raspberry Pi Camera (see Table 5.4). The LiDAR sensor was used to capture the robot's environment. The sensor data were passed to a remote PC (main PC) via the robot's PC using a wi-fi network (Fig. 5.16). The main ANADE algorithm runs in the main PC. Furthermore, a "Re-alsense" depth camera was attached to the TB3 waffle pi to identify moving obstacles (Fig. 5.17) using a separate PC.

There is a number of different types of object detection methods and object location measuring systems however, the main research area of this thesis is local path planning in dynamic environments. Therefore, a simple colour-based object detection method was developed to identify obstacles. The same method was used to identify the moving goal used in the experiments presented in Chapter 8. An image processing algorithm was used to fulfill this need. The used sensor for this take was the depth camera (Re-alsense camera) which provided two main sets of data.

- Real-time 2D RGB data of the robot's environment
- Real-time 3D point cloud of the robot's environment

Firstly, the image proccing algorithm detects the obstacles based on their colour. Once the obstacle's location is found in the 2D plane using the RGB data (Fig. 5.17), the linked point to this location in the 3D point cloud is identified. The obstacle coordinate is extracted from the 3D point cloud . The location information is fed to the main PC via TCP/IP communication protocol.

The TB3 waffle pi research robot platform has its own sensory system and a software to identify the robot's motion status. This sensory data is processed in the robot's PC. The robot's instantaneous location is then provided with respect to the global axis system (localisation of the robot). Furthermore, this sensory system and the built-in robot's software provided the robot's instantaneous linear velocity, orientation and angular velocity. This information is used to localise the obstacles captured by the LiDAR senor and depth camera through the SDP module.

Five experiments were conducted to test the new algorithm's performance under different dynamic conditions (see Fig. 5.18):

1. an MO coming towards the robot
2. an MO crosses the robot's path
3. an MO crosses the robot's path at different speeds
4. an MO challenging the robot in a trapped environment with random motion
5. multiple MOs randomly challenging the robot in a dynamic environment

5.4.2.1 Experiment 1: A Moving Obstacle Coming towards the Robot

The main purpose of this experiment was to identify the robot's behaviour when an MO continuously challenges the robot. The MO continuously moved towards the robot from

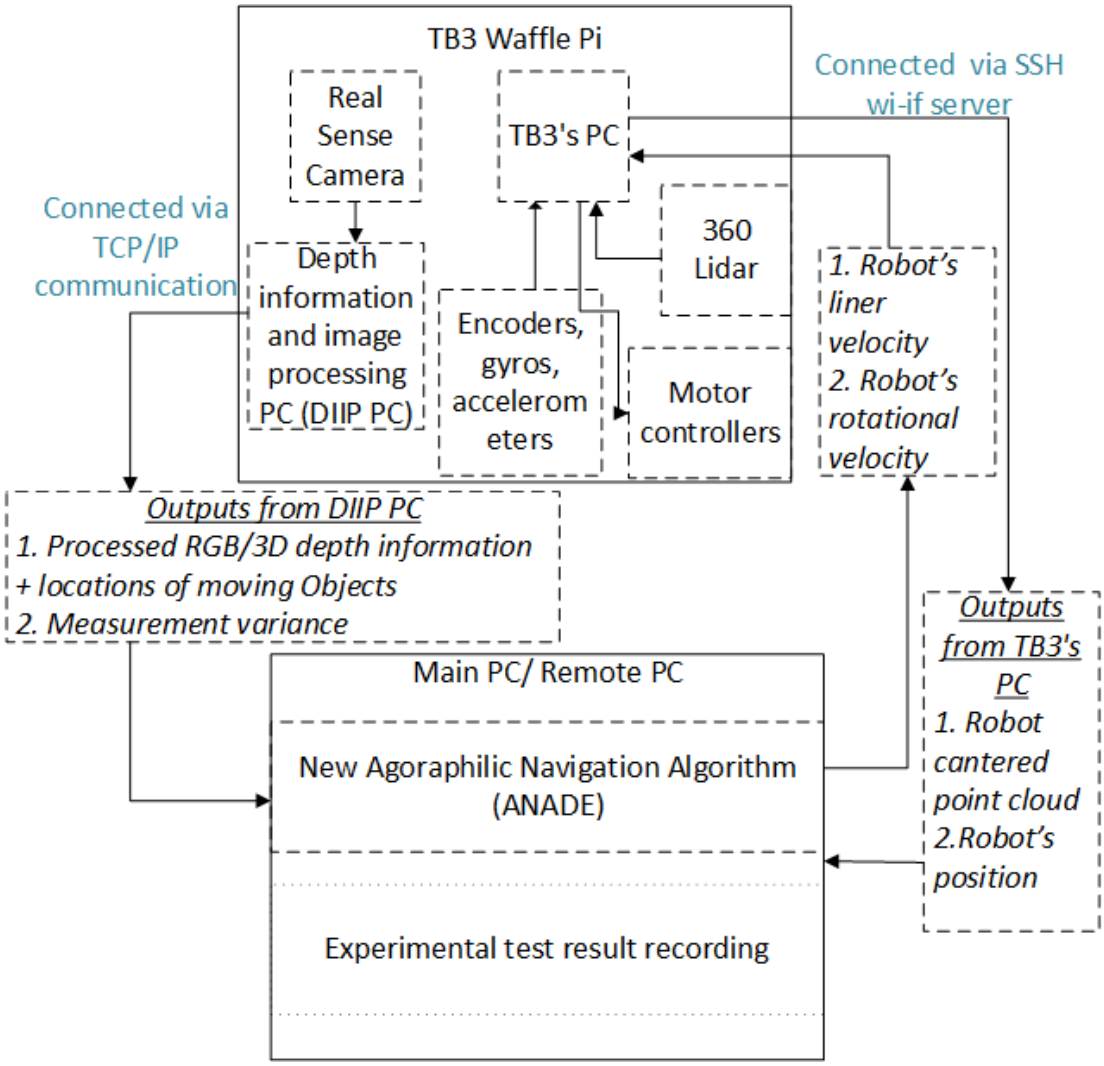


Fig. 5.16 Experimental Setup

the direction of the goal (i.e., the MO moved in the opposite direction of the robot).

The robot and the MO started moving at the instant labelled T1 (see Fig. 5.19). Initially, the robot was placed towards the x-direction. However, under the influence of the FSF-shaping module, the robot immediately changed its orientation towards the goal.

At the instant labelled T2, the robot predicted the path of the MO. The robot identified that the free space in its current moving direction was decreasing. Consequently, the robot started changing its direction even before it moved close to the obstacle. The dynamic objects prediction module and the tracking module is responsible for this be-

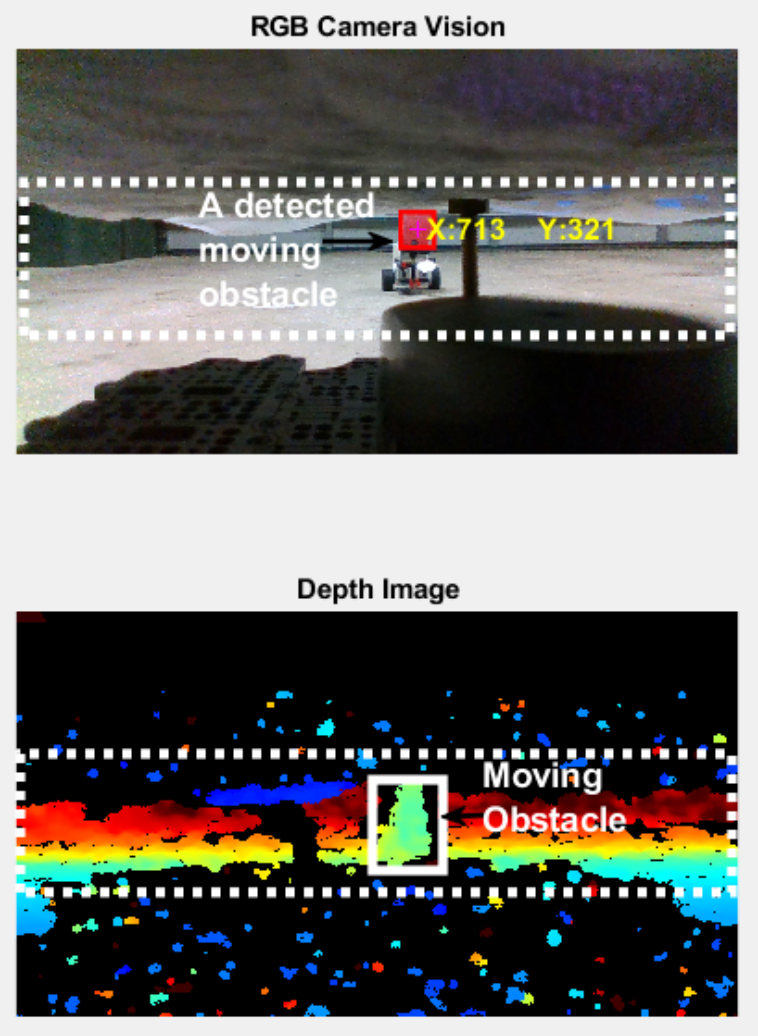


Fig. 5.17 RGB and Depth Images Captured from Real Sense^{TM} Camera

haviour.

At the instant labelled T3, the robot was passing the obstacle safely without any collision. After passing the MO, the robot again changed its direction directly towards the goal. This was due to the influence of the force-shaping module. Consequently, the robot succeeded in reaching its goal without collision at the instant labelled T4 (see Fig. 5.19). In this experiment, the robot travelled 385.0 cm in 91 s at an average speed of 4.2 cm/s.

The process of achieving the instantaneous driving force at time instant T2 is as follows:

Tab. 5.4 Hardware Specifications of TurtleBot3 Waffle Pi

Items	Specification
Max. translational velocity	0.26m/s
Max. rotational velocity	104.27deg/s
Size (L x W x H)	281 mm x 306 mm x 141 mm
Single board computer (robot's PC)	Raspberry Pi 3 Model B+
MCU of OpenCR	32-bit ARM Cortex®-M7 with FPU (216 MHz, 462 DMIPS)
IMU	Gyroscope 3 Axis Accelerometer 3 Axis Magnetometer 3 Axis
Laser Distance Sensor	360 Laser Distance Sensor LDS-01

1. Based on sensory data, the tracking module estimates the position (225.0, -118.5)cm and velocity (2.1cm/s) of the MO (Pseudocode-Step3).

2. Based on the position information of the tracking module, CGM is generated (Pseudocode-Step4; see Fig. 5.20 [a]).

3. Based on the position and velocity information, FGMs are generated (only one FGM is shown in the example; Pseudocode-Step5; see Fig. 5.20 [d]).

4. The FSHs for all the global maps are generated.

5. The FSFs for all the global maps are generated (see Fig. 5.20 [b] and [e]).

6. All the sets of FSFs are shaped towards the goal (see Fig.5.20 [e] and [f]).

7. IDFCs are generated (F_{c1},F_{c2},...,F_{cN}), Fig.5.20 [e] and [f].

8. The final instantaneous driving force is generated for the T2 time instant (see Fig.5.21).

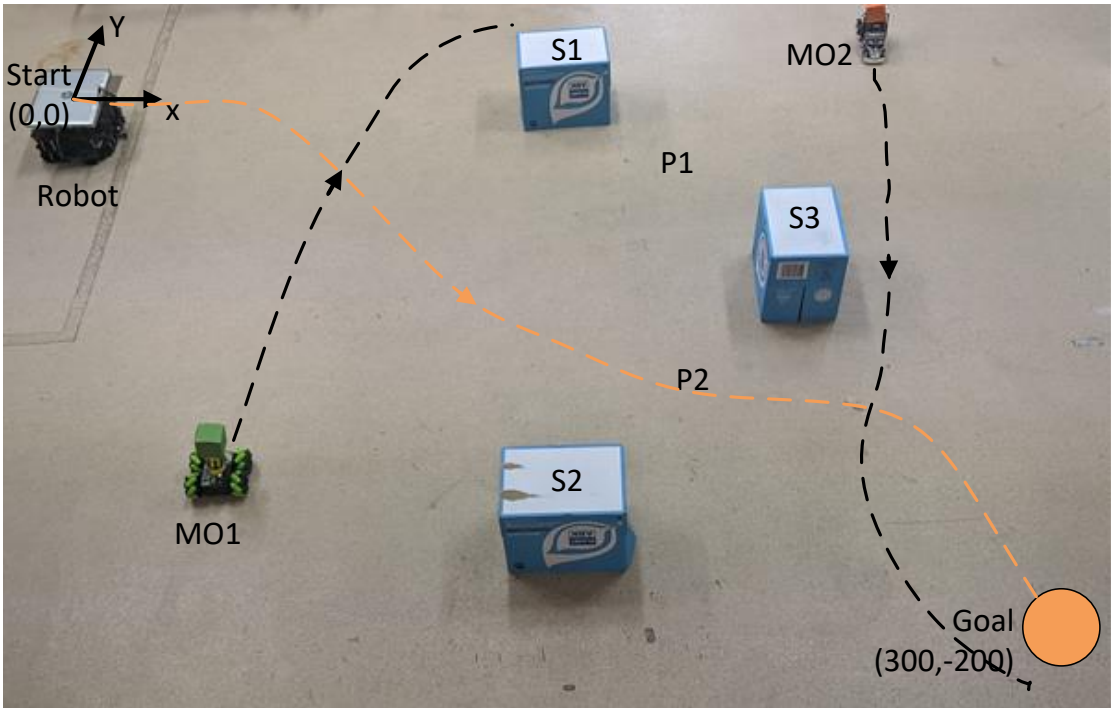


Fig. 5.18 The Experimental Environment—Locations of the Robot, Moving Obstacles and Static Obstacles in Experiment 5 at Time Instant 1

This process occurred in each time instant to generate the instantaneous driving force for the current iteration.

The key findings of this experiment are as follows:

1. In this experiment, the robot avoided the obstacle by changing its direction of motion (not by changing its speed).

2. The robot was placed initially in the x-direction. Although there was free space in the x-direction, the robot succeeded in changing its direction immediately towards the goal (300, –300; instant labelled T1). This was due to the influence of the force-shaping module.

3. The object tracking and the dynamic objects prediction modules have allowed the algorithm to know about the future growing or diminishing free space in the environment (instant labelled T2).

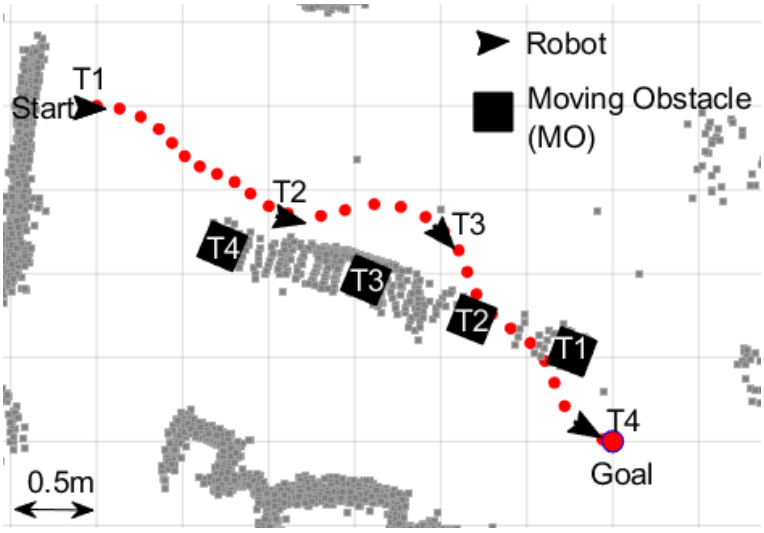


Fig. 5.19 The Robot's and Obstacle's Path Observed in Experiment 1 with their Locations Shown at Time Instants T1 to T4

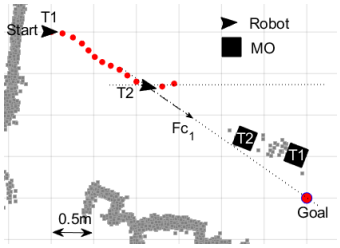
4. Once the robot passed the obstacle, it changed its direction back towards the goal (instant labelled T3). This was also due to the force-shaping module.

5.4.2.2 Experiment 2: One Moving Object Crosses the Robot's Path (Robot Changes its Moving Direction to Avoid the Collision

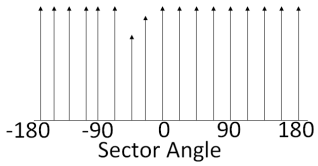
The main purpose of this experiment was to identify the robot's behaviour when an obstacle challenges the robot by crossing its path. The obstacle moved towards the negative y-direction. The MO intersected the robot's path at (170, –140).

At the instant labelled T2 (see Fig. 5.22), the robot succeeded in predicting the path of the MO. The robot identified that the MO would intersect its path around (107, –140). The robot changed its direction towards the y-direction (towards the direction of the moving object) with the intention of passing the obstacle before it intersected the robot's path.

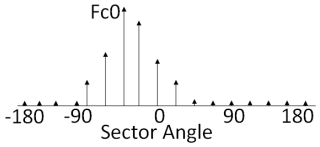
At the instant labelled T3 (see Fig. 5.22), the robot passed the obstacle safely without any collision before the obstacle crossed the robot's path. After passing the MO,



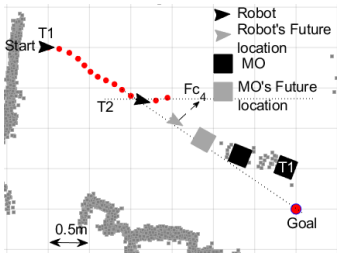
(a) CGM



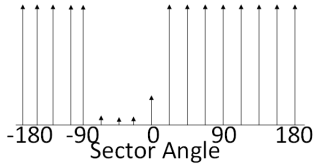
(b) Low-resolution FSFs for a CGM



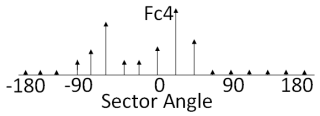
(c) Low-resolution shaped FSFs and Fc_1.



(d) Example for a FGM.



(e) Low-resolution FSFs for a FGM.



(f) Low resolution shaped FSFs and Fc_4.

Fig. 5.20 Instantaneous driving force component generation process for a particular time instance (T2).

the robot moved towards the goal. Finally, the robot reached its goal safely through the free space passages leading to the goal, labelled T4 (see Fig. 5.22). In this experiment, the robot travelled 370.4 cm in 84 s at an average speed of 4.4 cm/s.

The key findings of this experiment are as follows:

1. In this experiment, the robot avoided the obstacle by changing its direction of motion (not by changing its speed).

2. Initially, the robot changed its orientation towards the goal, as discussed in Experiment 1. (The force-shaping module is responsible for this decision.)

3. The algorithm predicted MO's velocity and its future locations with respect to the robot's future locations.

4. The free space concept influenced the robot to follow the free space leading to the goal.

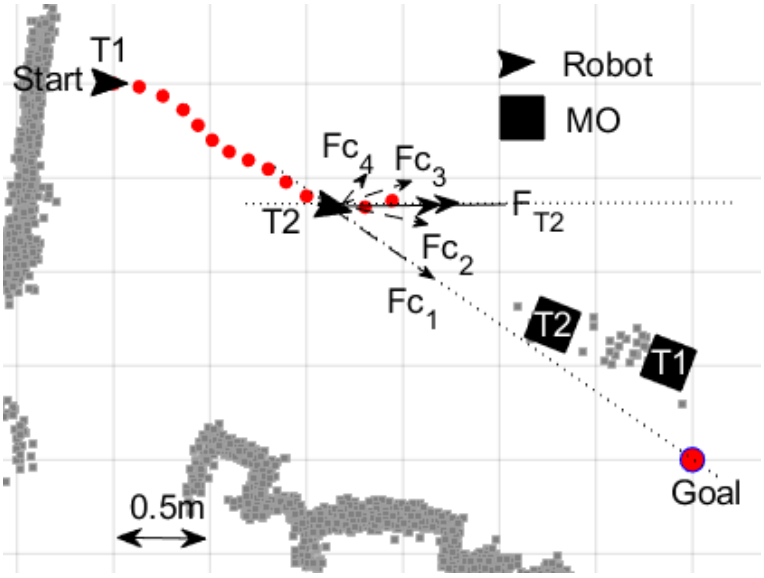


Fig. 5.21 Instantaneous Driving Force Components and the Final Instantaneous Driving Force at Time Instant T2 (F_{T2})

5.4.2.3 Experiment 3: One Moving Object Crosses the Robot's Path with Different Speeds (Robot Changes Its Speed to Avoid Collision).

The main purpose of this experiment was to identify the robot's driving force (magnitude) variation to avoid MOs. In this experiment, two tests were conducted. In both tests, the MO challenged the robot by crossing its path. However, the speed of the MO was different in each test. One experiment was conducted with a slow-moving obstacle that had an average speed of 2 cm/s. The second test was conducted with an object moving at an average speed of 3 cm/s. The robot's path and the driving forces were plotted for the experiment in Fig. 5.23 and Fig. 5.24, respectively.

In the conducted experiments, both the robot and the object started their motion at the instant labelled T1 (see Fig. 5.23). In both experiments, the robot started with a higher driving force (98% of the maximum driving force; see Fig. 5.24 [a] and [b]). Initially, the robot was placed facing the x-direction. However, the robot immediately changed its orientation towards the goal in both cases (see the second key finding in

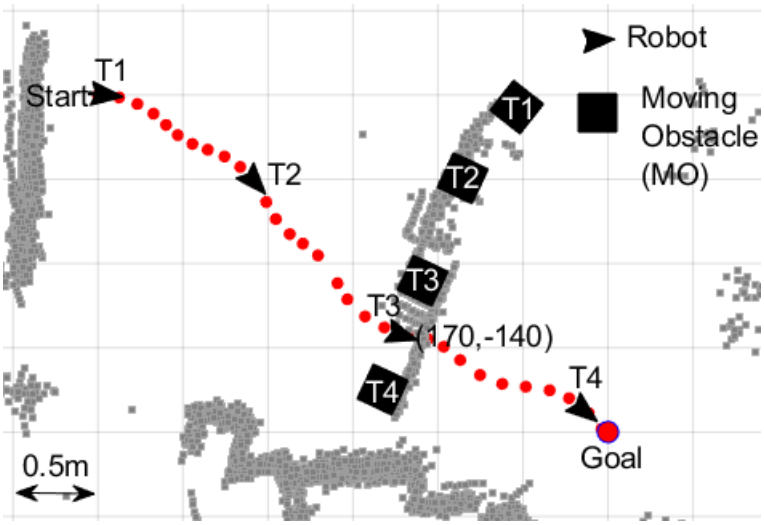


Fig. 5.22 Robot's and Obstacle's Path Observed in Experiment 2 with their Locations Shown at Time Instants T1 to T4

Experiment 1, Section 5.4.2.1).

At the instant labelled T2, the robot predicted the path and the velocity of the MO. In the test with the slow-moving obstacle, the robot changed its direction towards the negative y-direction with a higher driving force (see Figure 5.23 [a]; the intention was for the robot to pass the obstacle before it intersects the robot's path). Conversely, the robot in the test with the fast-moving object kept moving towards the goal (and towards the intersection point). However, the robot significantly reduced its driving force (see Fig. 5.24 [b]).

At the instant labelled T3, the robot in the test with the slow-moving obstacle passed the obstacle safely without any collision before the obstacle crossed the robot's path. After passing the MO, the robot moved towards the goal. During this process, the driving force did not vary significantly. In contrast, the robot in the test with the fast-moving obstacle reduced its speed and let the MO pass. Subsequently, the robot increased its driving force (Iterations 18–30) and moved towards the goal. Finally, the robot reached the goal safely in both cases (see Fig. 5.23, T4). A summary of the experimental results

Tab. 5.5 Summarised Information of Experiment 3

Parameter	With slow moving obstacle	With fast moving obstacle
Path length	370.4cm	365.4cm
Avg. speed	4.4cm/s	3.9cm/s
Travel time	84s	93s

is shown in Table 5.5.

The key findings of this experiment are as follows:

1. It was verified that the robot used the direction changes as well as the speed variations to avoid the obstacle.

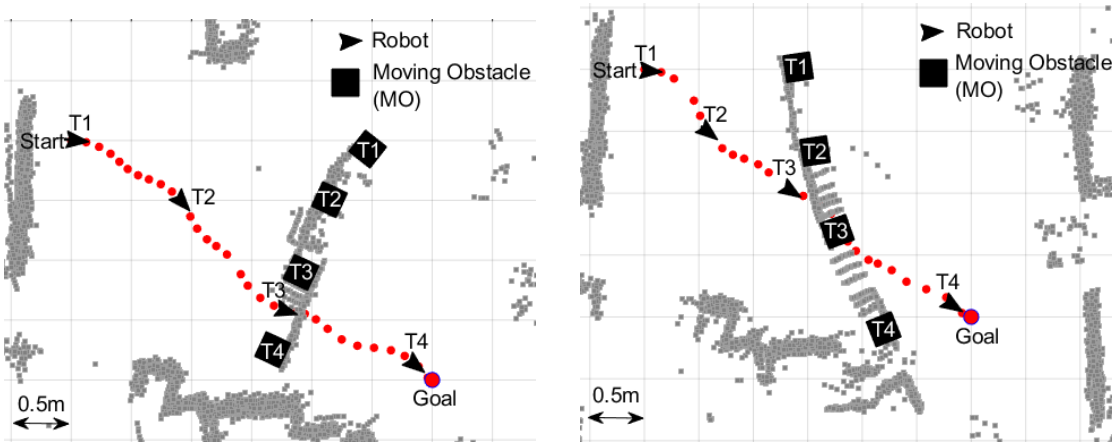
2. At the beginning, the robot immediately changed its direction towards the goal, as discussed in Experiment 1. (The force-shaping module is responsible for this decision).

3. The driving force variation plots show that the algorithm follows the FSF generation and the force-shaping concept as expected (the bigger the available free space, the bigger the robot's driving force).

4. The dynamic objects path prediction module allowed the robot to make prior decisions. (The robot can change its direction or speed even before the MO comes close to the robot).

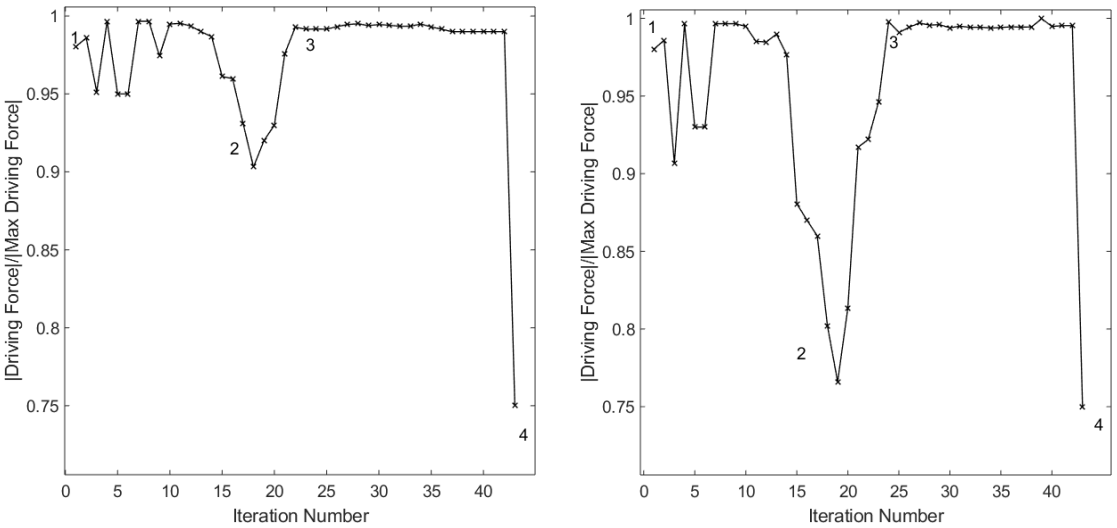
5. The algorithm predicted the MO's velocity and its future locations with respect to the robot's future locations.

6. The free space concept allowed the robot to follow the free space leading to the goal.



(a) Slow moving obstacle. **(b)** Fast moving obstacle.

Fig. 5.23 The Robot's Path with Slow-Moving and Fast-Moving Objects with Their Locations Shown at Time Instants T1 to T4



(a) Slow moving obstacle. **(b)** Fast moving obstacle.

Fig. 5.24 The Robot's Normalised Driving Force Variation over Fifty Iterations

5.4.2.4 Experiment 4: One Moving Object Challenging the Robot in a Trap with Random Motion

The main purpose of this experiment was to identify the robot's behaviour in a trapped situation. In this experiment, two SOs were placed to obstruct the robot's direct path to the goal. However, a new passage had been created in between these two obstacles (the new shortest path to the goal was through this passage). A MO was derived into this passage (to collide with the robot in the passage).

The robot and the MO started moving at the instant labelled T1. The robot immediately turned towards the goal (see the second key finding in Experiment 1, Section 5.4.2.1). At the instant labelled T2 (see Fig. 5.25), the robot successfully predicted the path and the speed of the obstacle and started moving away from the passage. Further, the robot decreased its speed (because the new direction was heading towards the moving object). From instants labelled T2 to T3, the robot moved with a slower speed, as shown in Fig. 5.25. This allowed the MO to pass the robot. After passing the obstacle, the robot started changing its direction towards the goal while increasing its speed (see the third key finding in Experiment 3, Section 5.4.2.3).

At the instant labelled T4, the MO changed its direction and started moving towards the robot to challenge it again. The robot kept its speed at a higher value and slightly changed its path with the intention of passing the obstacle before it crossed the robot's path (see the first key finding in Experiment 1, Section 5.4.2.1). Finally, the robot changed its direction slightly towards the goal and reached its goal safely (see Fig. 5.25, T5). In this experiment, the robot travelled 412.4 cm in 100 s at an average speed of 4.1 cm/s.

The key findings of this experiment are as follows:

1. It was verified that the dynamic objects prediction module helps the robot to avoid traps.

2. At the beginning, the robot immediately changed its orientation towards the goal, as discussed in Experiment 1. (The force-shaping module is responsible for this decision.)

3. The algorithm succeeded in predicting the MO's velocity and its future locations with respect to the robot's future locations.

4. The robot used a speed variation as well as a direction change at the same time to avoid collisions.

5. The free space concept allowed the robot to follow the future growing free space leading to the goal.

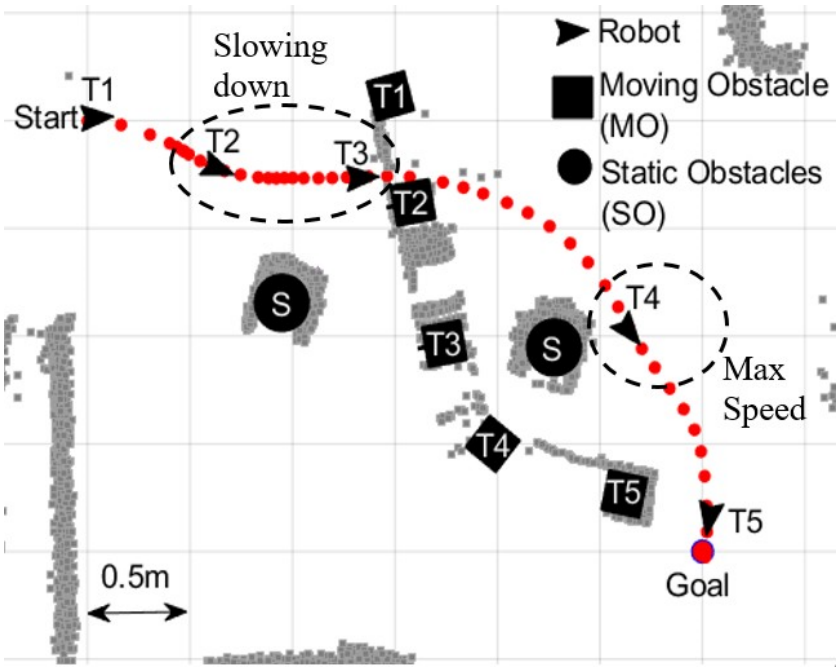


Fig. 5.25 The Robot's and Obstacle's Path Observed in Experiment 4 with their Locations Shown at Time Instants T1 to T5

5.4.2.5 Experiment 5: Multiple Moving Objects Challenging the Robot in a Random Environment (with Static Obstacles)

The main purpose of this experiment was to identify the robot's behaviour in a random environment with multiple MOs and SOs. In this experiment, multiple SOs were placed randomly. Two MOs were used to challenge the robot.

At the instant labelled T1 (see Fig. 5.26), the robot and MO1 started moving. The robot immediately turned towards the goal (see the second key finding in Experiment 1, Section 5.4.2.1).

At the instant labelled T2, the robot successfully predicted the path and the speed of the obstacle (MO1 was crossing the robot's path, see Fig. 5.26). However, the robot did not change its direction but decreased its speed (see the first key finding in Experiment 3, Section 5.4.2.3). From instants labelled T2 to T3, the robot moved at a slower speed. This allowed the MO to pass in front of the robot. At the same time, MO2 started moving towards the goal. After passing MO1, the robot slightly changed its direction towards the free space passage in between the two SOs. The robot did not choose P2—although it had more free space compared to P1—because P2 did not lead to the goal (the force-shaping module was the reason behind this decision). At the instant labelled T4, the robot changed its direction because of the influence of MO2. The robot passed the obstacle before it intersected the robot's path and the robot reached the goal safely (see Fig. 5.26, T5). In this experiment, the robot travelled 375.7 cm in 89 s at an average speed of 4.2 cm/s.

The key findings of this experiment are as follows:

1. The algorithm predicted the MO's velocity and future locations with respect to the robot's future locations.

2. The robot used speed variations as well as direction changes at the same time to avoid collisions.

3. The proposed algorithm successfully navigated the robot in an unknown dynamic environment with multiple moving objects.

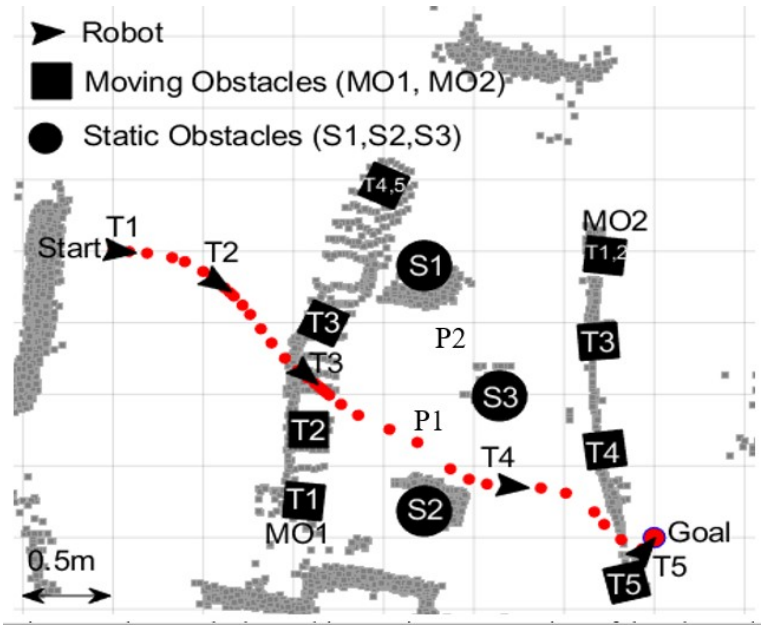


Fig. 5.26 The Robot's and Obstacle's Path Observed in Experiment 5 with their Locations Shown at Time Instants T1 to T5

5.4.2.6 Summarised Findings of Experimental Testing

The following summarises the findings of the experimental testing:

1. The chosen five experiments confirmed the importance of FSF generation and the force-shaping modules (see the first key finding in Experiments 1–5). This also verified the FSF concept.

2. The environment prediction module allowed the robot to make prior decisions. This was also proven by all the experiments.

3. The two basic experiments (i.e., Experiments 1 and 2) showed that the robot uses directional changes to avoid simple challenges created by one slow-moving object. The decisions made in those two experiments helped the robot to reach the goal safely and efficiently.

4. The results observed in Experiment 3 confirmed that the robot makes human-like intelligent decisions.

   When the slow-moving obstacle crossed the robot's path, the robot maintained a higher speed and passed the obstacle by changing its direction. When the fast-moving obstacle crossed the robot's path, the robot almost stopped and allowed the obstacle to pass the robot. This behaviour is similar to human decision-making. The robot gained this ability because of the dynamic object tracking and prediction modules.

5. Experiment 4 demonstrated a trapped situation for the robot. The robot could easily avoid the trap because of its early detection of the trap using the dynamic object prediction module. This mimics human-like behaviour and enforces the importance of the dynamic object prediction module.

6. Experiment 5 demonstrated the robot's behaviour in a random environment with multiple MOs. The robot successfully reached the goal in the unknown environment with multiple MOs.

5.5 Summary

The ANADE II is a local path planner to navigate mobile robots in unknown static and dynamic environments. The new algorithm takes an optimistic approach to solving the

navigation problem. It does not look for obstacles to avoid but for free space solutions to follow. Compared to the ANADE I, the algorithm discussed in this chapter uses a dynamic objects path prediction methodology to identify the robot's future environments. The ANADE II pulls the robot through the future growing free space passages leading to the goal, whereas the ANADE I discussed in Chapter 4 only considers the current free space. This new methodology has successfully eliminated some of the drawbacks of the ANADE I, which is limited to simple dynamic environments (multiple obstacles do not challenge the robot at the same time). The optimistic approach of the new algorithm combined with future dynamic objects path prediction makes the ANADE II superior to the ANADE I.

The primary aim of this chapter was to introduce the ANADE II and verify its success in dynamic environments. Therefore, two simulation tests and five real-world experiments were carefully designed and executed to demonstrate the algorithm's capabilities. The experimental test results clearly demonstrated that the ANADE II allows the robot to successfully negotiate its path among moving obstacles by changing its direction and speed. The robot can reach its goal even in challenging environments containing traps by using the object tracking and the dynamic objects path prediction modules. The objects path prediction helped the robot to make prior decisions, which was confirmed by the observed results. Further, the new algorithm showed its human-like behaviours in the real-world experiments.

However, the ANADE II uses a numerical weighting system to quantify the shaping factors by considering the angle difference between the sector angle and the robot-to-goal angle. Although the ANADE II performs well in dynamic environments, the algorithm's performance can be further improved by considering other parameters, such

as sector distances and the speeds of moving obstacles, when shaping the free space forces. Further, having increased flexibility in shaping forces yields advantages in refined decision-making. Therefore, an fuzzy logic controller-based force-shaping module was developed and embedded in the force-shaping module to further optimise the algorithm's decision-making. Chapter 6 discusses the development of this new controller and provides experimental validations of the further enhanced algorithm, the ANADE III.

6

Chapter 6: Artificial Intelligence Based-Agoraphilic Navigation Algorithm in Dynamic Environment (ANADE III)

6.1 Introduction

Dynamic environments are characterised by high levels of uncertainty, thereby presenting significant challenges to navigational algorithms [35]. Navigational techniques based on AI are frequently used to address this problem. Among these, FL is commonly used to overcome uncertainties at a low computational cost. Most alternative navigation algorithms fail to use the velocities of MOs as an input for decision-making. This is due to the difficulty of estimating the velocity vectors of MOs [36]. Consequently, most existing navigation algorithms are unable to navigate in uncertain dynamic environments, leading to suboptimal performance or system failure in complex situations [36]. However, as mentioned in Chapter 5, the tracking module used in the ANADE II estimates the velocities of MOs (used as an input for the DOPP module). The novel ANADE III algorithm presented in this chapter uses the estimated velocity vectors as an input for the new force-shaping module. Further, the ANADE III uses all advantages of the ANADE II and strengthens the algorithm further by introducing a new FL-based controller to the force-shaping module. This new AI-based component, along with the other modules discussed in Chapters 4 and 5, helps the algorithm to successfully over-

come the aforementioned common weakness of other navigation algorithms at a low computational cost.

6.2 The Architecture of the ANADE III

Additionally, in the ANADE III, a modular-based architecture was used. This architecture increases the adaptability and simplicity of the algorithm. The ANADE III comprises eight main modules:

1. Sensory Data Processing (SDP) module
2. Dynamic Obstacle Tracking (DOT) module
3. Dynamic Obstacle Position Prediction (DOPP) module
4. Current Global Map (CGM) generation module.
5. Future Global Map (FGM) generation module.
6. Free Space Attraction (FSA) module.
    (a) current and future FSH generation module
    (b) FSF generation module
    (c) force-shaping module
    (d) Instantaneous Driving Force Component (IDFC) generation module
7. IDFC weighting module.
8. Rvobot's Motion Command (RMC) generation module

The SDP module collects sensory data from the sensors mounted on the robot (RealSense^{TM} Camera and LiDAR, encoders and associated equipment) and transforms

all robot-centric data to a world reference frame (global positioning system). The tracking module then generates two outputs—a set of locations of moving obstacles and a set of velocity values of moving obstacles—within the map using the globalised sensory data. Subsequently, the algorithm is divided into two sections, as follows:

1. the IDFC (F_c) generation for the CGM (CGM-F_{c1})

2. the IDFC (F_c) generation for the FGMs (FGM-F_{c2}, -F_{c3}, ..., -F_{cN}; Prediction).

The IDFC generation for the CGM-F_{c1} section of the algorithm uses the current environment of the robot (the CGM) and generates the component force (F_{c1}) based on the current surroundings of the robot. In this process, the CGM becomes the input for the FSH generation module. The generated FSH is then converted to a set of FSFs by the FSF generation module. Subsequently, the force-shaping module regulates the FSFs such that the robot will move towards the goal. Finally, the IDFC generation module generates the final driving force component (F_{c1}) for the CGM using the shaped FSFs (see Fig. 6.1).

The second output of the tracking module (the current states of MOs) is used as the input for the DOPP module (see Fig. 6.1). This returns N-1 FGMs, where N is the final prediction of the DOPP module. Each FGM is input into the FSH generation module. The generated FSH is then converted to a set of predictive FSFs by the FSF generation module. Subsequently, the force-shaping module shapes the FSFs such that the robot will move towards the goal. Unlike in previous ANADEs, the new force-shaping module considers not only the sector angel but also the sector distance and sector velocity when shaping the forces (a detailed discussion is provided in Section 6.3.2). Finally, the IDFC generation module generates the final F_{c1} for the corresponding FGM. This process occurs for each FGM.

Finally, all the IDFCs (F_{c1}, F_{c2}, ..., F_{cN}) related to the CGMs and FGMs are input into the IDFC weighing module. This module generates the final driving force the robot's actual driving force for the current iteration. This process repeats for each iteration.

In the ANADE III, all the modules are mostly similar to the ANADE II's main modules except for the force-shaping module in the FSA module. The force-shaping module can be defined as the most powerful component in the FSA module (see Fig. 6.1). The force-shaping module has a direct influence on manipulating the FSFs. The development of these submodules is discussed in the following sections.

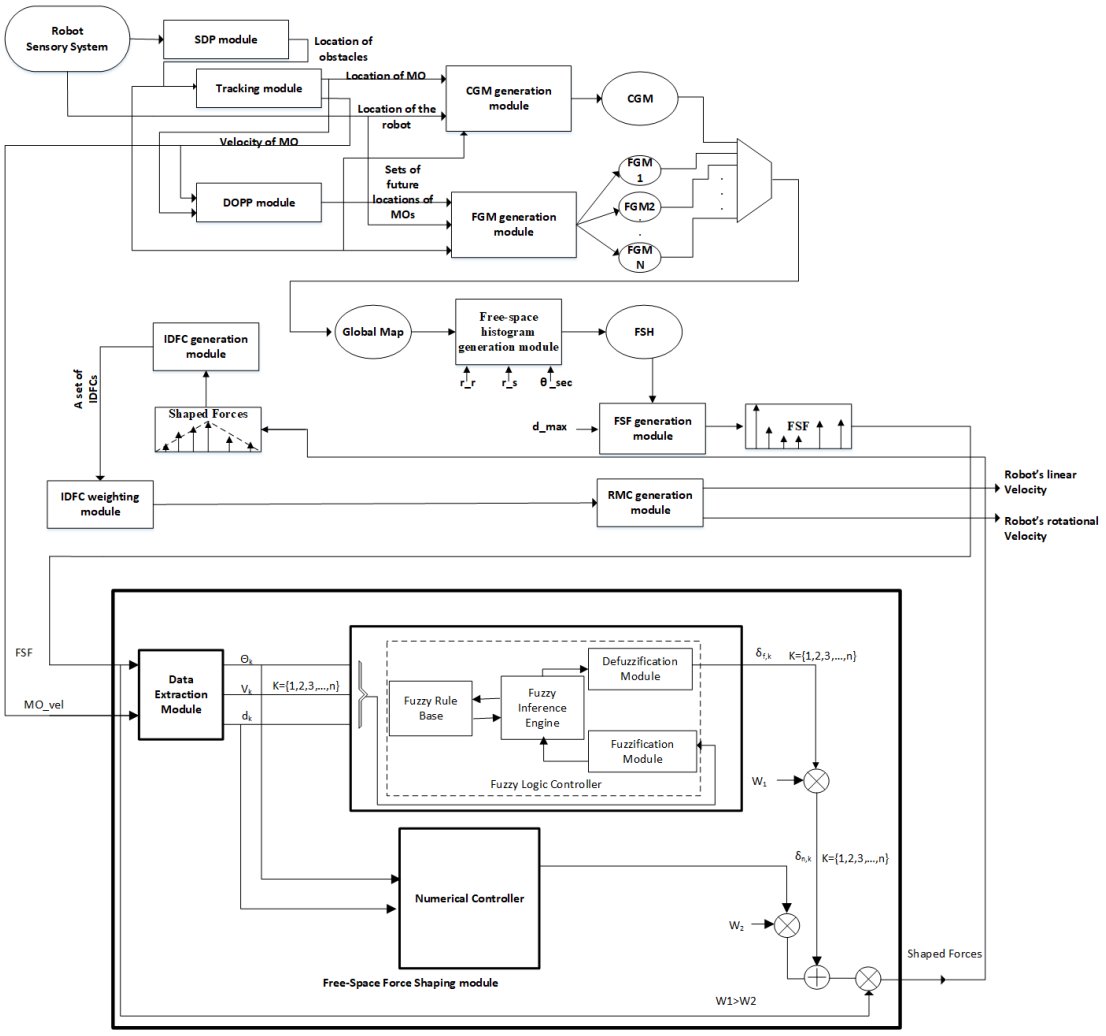


Fig. 6.1 Block diagram of ANADE III

6.3 New Main Modules Used in ANADE III to Optimise the Performance

6.3.1 Force-Shaping Module

A set of FSFs is taken as the input of the force-shaping module. The sets of FSFs generated for each global map need to be directed towards the goal. A weighing technique is used to perform this task. The force-shaping module is considered to be one of the most important parts of the algorithm, as it has a direct influence on selecting the robot's driving direction in the corresponding iteration.

To perform this task, a numerical weighing system and a FLC are used.

6.3.1.1 Numerical Function

During this process, FSFs pointing towards the goal are assigned a higher weighing factor compared to FSFs pointing away from the goal. Each of the sector forces (F_k) is modified by a numerical force-shaping coefficient (δ_N). The numerical force-shaping coefficients are determined by Eq. 6.1. If F_k points directly towards the goal, the corresponding numerical force-shaping coefficient is set at its maximum value of 1.

$$\delta(\theta) = \begin{cases} \frac{1}{90}\theta + 1, & \text{if } -90 \leq \theta \leq 0; \\ \frac{-1}{90}\theta + 1, & \text{if } 0 \leq \theta \leq 90; \\ \frac{1}{90}\theta - 3, & \text{if } 270 \leq \theta \leq 360; \\ \frac{-1}{90}\theta - 3, & \text{if } -270 \leq \theta \leq 360; \\ 0, & \text{otherwise.} \end{cases} \tag{6.1}$$

6.3.1.2 Fuzzy Logic Controller

The numerical weighing system quantifies the shaping factors by only considering the angle difference between the sector angle and robot-to-goal angle. In a cluttered dynamic environment, this single factor does not have sufficient spatial or temporal resolution. To perform more effectively, the system must consider other parameters such as sector distances and speeds of moving obstacles. Further, having increased flexibility in shaping forces yields advantages in refined decision-making. Therefore, a FLC-based force-shaping module is also embedded in the force-shaping module. The rest of this sub-section discusses this controller.

A linguistic fuzzy model, known as the 'Mandani Approach', was used to develop the fuzzy logic controller. Its basic components are shown in Fig. 6.2 and comprise four main modules:

1. a fuzzification module
2. a fuzzy rule bases
3. a fuzzy inference engine
4. a defuzzification module.

To develop the fuzzification module, four databases were designed. Three databases were used for the three inputs (θ_{diff},$\mathrm{d}_{k,n}$ and $\mathrm{v}_{k,n}$) (Fig. 6.4) and one for the output, the fuzzy shaping factor (δ_f) (Fig. 6.5).

The three input databases were modelled using multiple fuzzy data sets. For all membership functions over a given universe of discourse, the membership function of a fuzzy set S, denoted by μ_s, maps elements x $\in$ X into a numerical value in the closed unit interval, for example:

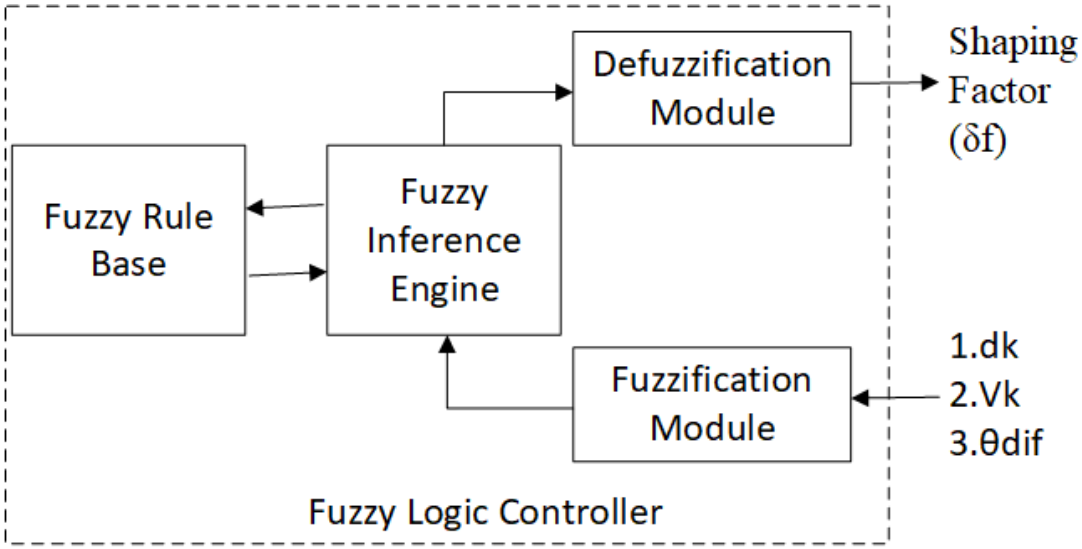


Fig. 6.2 A Block Diagram of the Fuzzy Logic Controller

μ_s (x) : $\rightarrow$ [0, 1]

The database for the first input θ_{diff} consists of nine fuzzy membership functions derived from Eq. 6.2. The corresponding linguistics are LRR: Left Rear, LR: Left Rear, SL: Side Left, LF: Left Front, F: Front, RF: Right Front, SR: Right Side, RR: Right Rear and RRR: Right Rear (Fig. 6.3).

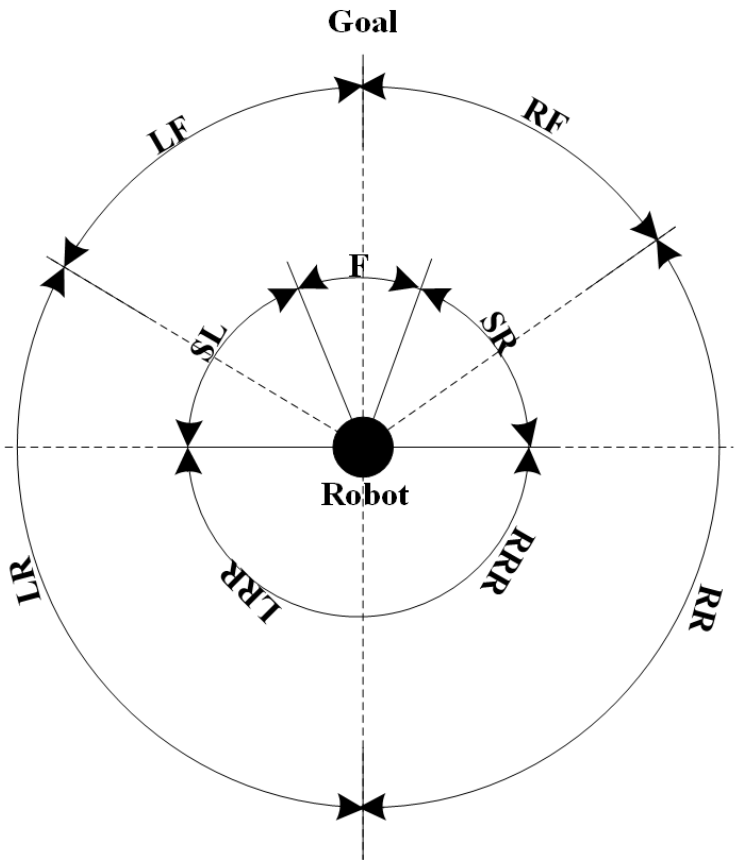


Fig. 6.3 FSH Used to Derive the Membership Function for Input θ_{diff}

$$
\begin{aligned}
\mu_{LRR}(\theta) &= max\{min\left(\frac{-90-\theta}{90}, 1\right), 0\} \\
\mu_{LR}(\theta) &= max\{min\left(\frac{\theta+180}{90}, \frac{-60-\theta}{30}\right), 0\} \\
\mu_{SL}(\theta) &= max\{min\left(\frac{\theta+90}{30}, \frac{-30-\theta}{30}\right), 0\} \\
\mu_{LF}(\theta) &= max\{min\left(\frac{\theta+60}{30}, \frac{-\theta}{30}\right), 0\} \\
\mu_{F}(\theta) &= max\{min\left(\frac{\theta+30}{30}, \frac{30-\theta}{30}\right), 0\} \\
\mu_{RF}(\theta) &= max\{min\left(\frac{\theta}{30}, \frac{60-\theta}{30}\right), 0\} \\
\mu_{SF}(\theta) &= max\{min\left(\frac{\theta-30}{30}, \frac{180-\theta}{90}\right), 0\} \\
\mu_{RR}(\theta) &= max\{min\left(\frac{\theta-60}{30}, \frac{180-\theta}{90}\right), 0\} \\
\mu_{RRR}(\theta) &= max\{min\left(\frac{\theta-90}{90}, 1\right), 0\}
\end{aligned}
\tag{6.2}
$$

Where: $\theta \in$ {-180,180}

The database for the second input, the normalised sector distance ($d_{k,n}$), consists of five fuzzy membership functions as shown in Eq. 8.7. The corresponding linguistics are VC: Very Close, C: Close, N: Near, F: Far, VF: Very Far.

$$\begin{aligned}
\mu_{VC}(d_{k,n}) &= max\{min\left(\frac{-d_{k,n}-0.3}{0.2}, 1\right), 0\} \\
\mu_{C}(d_{k,n}) &= max\{min\left(\frac{d_{k,n}-0.1}{0.2}, \frac{0.5-d_{k,n}}{0.2}\right), 0\} \\
\mu_{N}(d_{k,n}) &= max\{min\left(\frac{d_{k,n}-0.3}{0.2}, \frac{0.7-d_{k,n}}{0.2}\right), 0\} \\
\mu_{F}(d_{k,n}) &= max\{min\left(\frac{d_{k,n}-0.5}{0.2}, \frac{0.9-d_{k,n}}{0.2}\right), 0\} \\
\mu_{VF}(d_{k,n}) &= max\{min\left(\frac{d_{k,n}-0.7}{0.2}, 1\right), 0\}
\end{aligned} \tag{6.3}$$

The database for the third input, the normalised sector velocity ($v_{k,n}$), consists of seven fuzzy membership functions as shown in Eq. 8.8. The variable $v_{k,n}$ is positive if the moving obstacle is approaching the robot. The corresponding linguistics are NF: Negative Fast, NM: Negative Medium, NS: Negative Slow, Z: Zero, PS: Positive Slow, PM: Positive Medium and PF: Positive Fast.

$$
\begin{aligned}
\mu_{NF}(v_{k,n}) &= max\{min\left(\frac{-v_{k,n}-0.77}{0.33}, 1\right), 0\} \\
\mu_{NM}(v_{k,n}) &= max\{min\left(\frac{v_{k,n}+1}{0.33}, \frac{-0.33-v_{k,n}}{0.33}\right), 0\} \\
\mu_{NS}(v_{k,n}) &= max\{min\left(\frac{v_{k,n}+0.77}{0.33}, \frac{-v_{k,n}}{0.33}\right), 0\} \\
\mu_{Z}(v_{k,n}) &= max\{min\left(\frac{v_{k,n}+0.33}{0.33}, \frac{-0.33-v_{k,n}}{0.33}\right), 0\} \\
\mu_{PS}(v_{k,n}) &= max\{min\left(\frac{v_{k,n}}{0.33}, \frac{0.77-v_{k,n}}{0.33}\right), 0\} \\
\mu_{PM}(v_{k,n}) &= max\{min\left(\frac{v_{k,n}-0.33}{0.33}, \frac{1-v_{k,n}}{0.33}\right), 0\} \\
\mu_{PF}(v_{k,n}) &= max\{min\left(\frac{v_{k,n}-1}{0.33}, 1\right), 0\}
\end{aligned}
\tag{6.4}
$$

The simple output database is represented by eight fuzzy sets, as shown in Fig. 6.5.

The fuzzy rule base was developed with 315 rules, Table 6.1. The following structure was used to develop the rules.

if (condition1 is Lingustic1,i)AND (condition2 is Lingustic2,j) AND (condition3 is Linguistic3,k) THEN Rule base output is ROuti,j,k

Ex: *if (angDif is F)AND (dk,n is VF) AND (Vk,n is NF) THEN out is Z*

Tab. 6.1 Structure of the Fuzzy Rule Bases.

NF	VC	C	N	F	VF
LRR	Z	Z	Z	Z	Z
LR	Z	Z	Z	L	MH
SL	Z	Z	Z	HVH	HVH
LF	Z	MH	MH	VH	VH

F	Z	HVH	HVH	VH	VH
RF	Z	MH	MH	VH	VH
SR	Z	Z	Z	HVH	HVH
RR	Z	Z	Z	Z	MH
RRR	Z	Z	Z	Z	Z
NM	VC	C	N	F	VF
LRR	Z	Z	Z	Z	Z
LR	Z	Z	Z	Z	M
SL	Z	Z	Z	H	H
LF	Z	M	M	VH	VH
F	Z	H	H	VH	VH
RF	Z	M	M	VH	VH
SR	Z	Z	Z	H	H
RR	Z	Z	Z	Z	M
RRR	Z	Z	Z	Z	Z
NS	VC	C	N	F	VF
LRR	Z	Z	Z	Z	Z
LR	Z	Z	Z	Z	LM
SL	Z	Z	Z	MH	MH
LF	Z	LM	LM	HVH	VH
F	Z	MH	MH	VH	VH
RF	Z	LM	LM	HVH	VH
SR	Z	Z	Z	MH	MH

RR	Z	Z	Z	Z	LH
RRR	Z	Z	Z	Z	Z
Z	VC	C	N	F	VF
LRR	Z	Z	Z	Z	Z
LR	Z	Z	Z	Z	L
SL	Z	Z	Z	M	M
LF	Z	L	L	H	VH
F	Z	M	M	VH	VH
RF	Z	L	L	H	VH
SR	Z	Z	Z	M	M
RR	Z	Z	Z	Z	L
RRR	Z	Z	Z	Z	Z
PS	VC	C	N	F	VF
LRR	Z	Z	Z	Z	Z
LR	Z	Z	Z	Z	ZL
SL	Z	Z	Z	LM	LM
LF	Z	ZL	ZL	MH	HVH
F	Z	LM	LM	HVH	HVH
RF	Z	ZL	ZL	LM	LM
SR	Z	Z	Z	LM	LM
RR	Z	Z	Z	Z	ZL
RRR	Z	Z	Z	Z	Z
PM	VC	C	N	F	VF

LRR	Z	Z	Z	Z	Z
LR	Z	Z	Z	Z	Z
SL	Z	Z	Z	L	L
LF	Z	Z	Z	M	H
F	Z	L	L	H	H
RF	Z	Z	Z	M	H
SR	Z	Z	Z	L	L
RR	Z	Z	Z	Z	Z
RRR	Z	Z	Z	Z	Z
PF	VC	C	N	F	VF
LRR	Z	Z	Z	Z	Z
LR	Z	Z	Z	Z	Z
SL	Z	Z	Z	ZL	ZL
LF	Z	Z	Z	LM	MH
F	Z	ZL	ZL	MH	MH
RF	Z	Z	Z	LM	MH
SR	Z	Z	Z	ZL	ZL
RR	Z	Z	Z	Z	Z
RRR	Z	Z	Z	Z	Z

Z-Zero, ZL-Zero Low, L-Low, LM-Low Medium,

M-Medium, MH-Medium High, H-High,

HVH-High Very High, VH-Very High

Based on the given inputs, the fuzzy inference engine chooses eight firing rules out

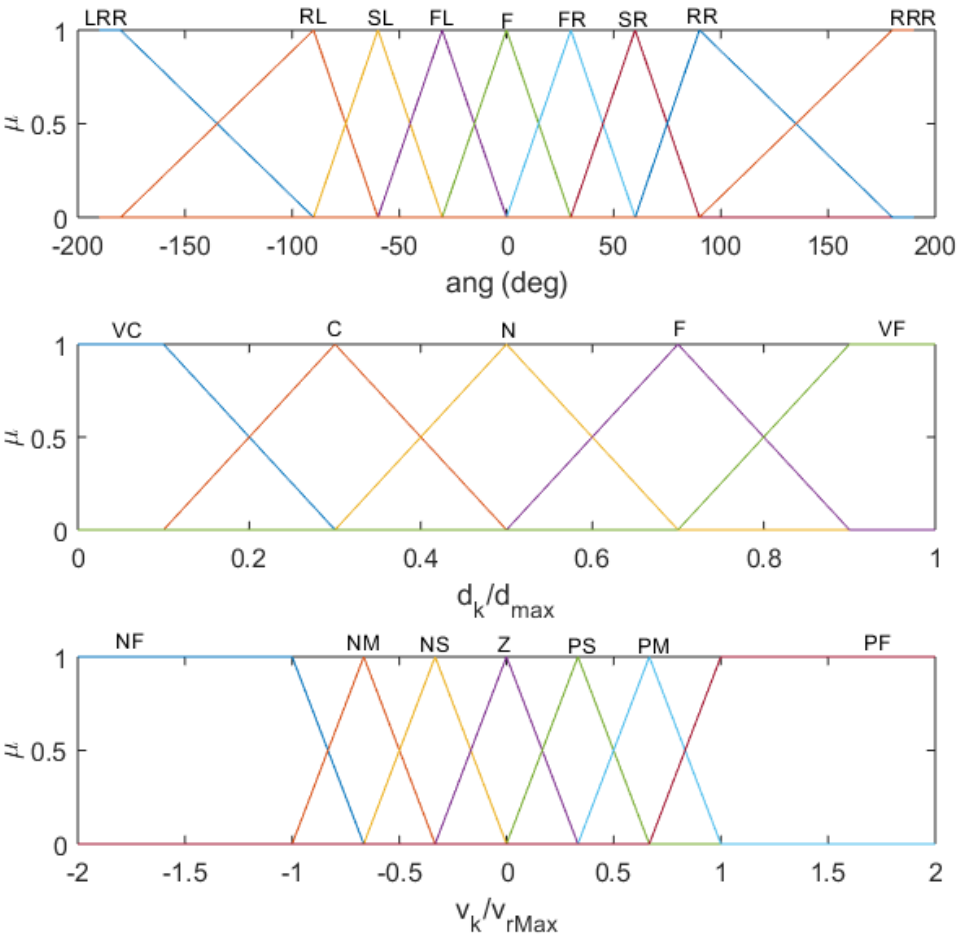


Fig. 6.4 Input Membership Functions of the Fuzzy Controller

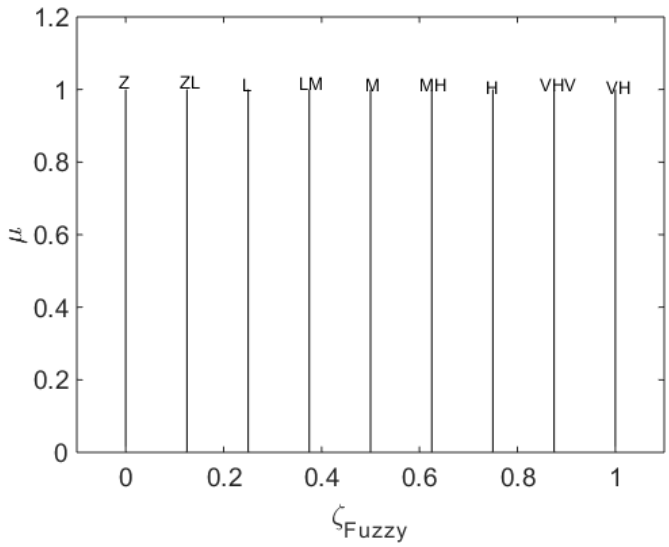


Fig. 6.5 Output Membership Functions of the Fuzzy Controller

of 315 from the rule base. It then calculates the firing power of each selected rule (α_i), Eq. 6.5

$$\alpha_i = min\{\mu_{L1}(\theta), \mu_{L2}(d_{k,n}), \mu_{L3}(v_{k,n})\} \tag{6.5}$$

Where;

i=1,2,3,4

L1 $\in$ {LRR, LR, SL, LF, F, RF, SR, RR, RRR}

L2 $\in$ {VC, C, N, F, VF}

L3 $\in$ {NF, NM, NS, Z, PS, PM, PF}

The defuzzification module uses the fuzzy rule with the maximum firing strength to find the final fuzzy shaping factor for the corresponding sector ($\delta_{f,k}$).

6.4 Results and Discussion

The experimental work was conducted to demonstrate and validate the performance of the new algorithm. The experiments were also designed to test the performance of the new fuzzy logic controller as well as the DOPP module. From the range of investigations performed, a series of three basic experiments and one random experiment were specifically selected for presentation in this chapter:

1. an obstacle moving towards the goal from the start point
2. an obstacle moving towards the start point from the goal
3. an obstacle moving perpendicularly across the robot's path

4. three moving obstacles simultaneously challenging the robot and pushing the robot towards a trap.

As mentioned in Chapter 5, the TB3 waffle pi research platform and developed experimental setup was used to conduct the real-world experiments precent in this section.

6.4.1 Experiment 1: An Obstacle Moving Towards the Goal from the Start Point

The primary purpose of this experiment was to identify the robot's behaviour when a MO moved in front of the robot towards the goal (the MO's speed being greater than the maximum speed of the robot). This experiment also showed the improvements of the algorithm after introducing the FLC. In this experiment, two tests were conducted. In both tests, the robot's environment and allowed maximum speed were kept constant. However, in one experiment, the FLC was disabled.

For both tests the robot's starting point was recorded as (0,0), the goal location was (300,0), the MO's start point was (27,0) and the MO's velocity was maintained at 6.3 cm/s. The robot's maximum allowed velocity was limited to 4.5 cm/s.

Case1: Basic force shaping module without the FLC

Referring to Fig. 6.6, the robot and the moving obstacle started moving at the instant labelled 'T1'. At this moment, the distance between the robot and the MO was recorded as 27 cm (the MO was close to the robot). As the MO was in close proximity to the robot, the robot immediately started moving away from the MO.

At the instant labelled 'T2', the robot kept moving away from the MO and followed a long path, outlined in red and blue, to reach the goal.

At the instant labelled 'T3', the robot changed its direction back towards the goal. Once the MO passed the goal, the robot reached its destination safely without any col-

Tab. 6.2 Summarised Test Results of Experiment 1

Parameter	Without FLC	With FLC	20cm Performance comparison with respect to case 2
Travel time	85s	74s	14%
Path length	368cm	318cm	14%
Avg Speed	4.3cm/s	4.3cm/s	0%

lision at the instant label 'T4'. In this experiment, the robot travelled 368 cm in 85.6 s with an average speed of 4.3 cm/s.

Case2: New force-shaping module with the FLC

Referring to Fig. 6.7, the robot and the moving obstacle started moving at the instant labelled 'T1'. At this moment, the distance between the robot and the MO was recorded as 27 cm (the MO being close to the robot). However, the FLC considers the speed and the direction of the MO's motions. Consequently, the algorithm allowed the robot to move towards the goal without changing its direction.

At the instant labelled 'T2', the robot kept moving in the same direction towards the goal behind the MO. With the influence of the new FLC, the robot did not change its direction unnecessarily.

At the instant labelled 'T3', the robot showed a slight deviation from the shortest path. However, the robot redirected its path towards the goal and successfully reached the goal, at the instant label 'T4', without collision. In this experiment, the robot travelled 318 cm in 73.9 s with an average speed of 4.3 cm/s.

Table 6.2 provides a numerical summary of these data.

Key findings of this experiment:

1. The MO was moving in front of the robot towards the goal, but the MO had a higher speed. In an optimised system, the robot would have identified that the MO

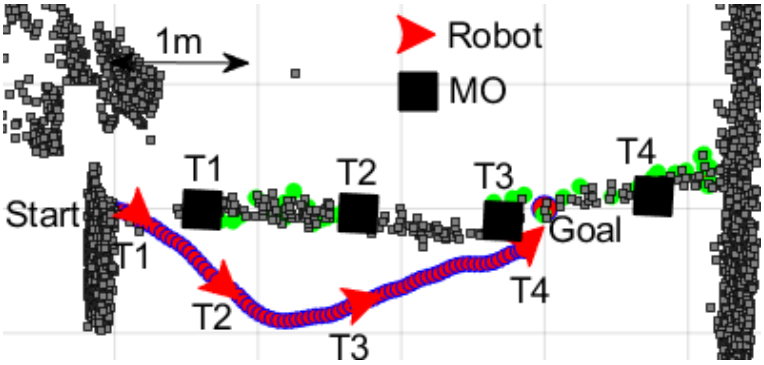


Fig. 6.6 Robot's Path Observed in Experiment 1 without the New FLC. The Locations of the Robot and Moving Obstacle are Shown at Four Different Time Instants (T1-T4)

did not impede its pathway towards the goal. However, in the absence of the FLC, the basic force-shaping module gave unnecessarily more weight to existing free space and directed the robot away from the MO. Conversely, the force-shaping module directed by the new FLC allowed the robot to move towards the goal behind the MO while maintaining a safe distance.

2. The basic force-shaping module had not considered the velocity of the moving object when it was shaping the FSFs. (which led to a path similar to that when avoiding a static obstacle, see Fig. 6.8).

3. The FLC optimised the efficiency of the algorithm by reducing the time taken to reach the goal in addition to reducing the path length.

4. Initially, the robot avoided the obstacle by changing its direction of motion (the basic force-shaping module). Subsequently, the force-shaping module with the FLC enabled correctly identified that the MO was not a challenge to the robot.

5. The object tracking module allowed the algorithm to accurately determine the states (position and velocity) of the MO. This optimised the efficiency of the algorithm in successfully completing the navigation task.

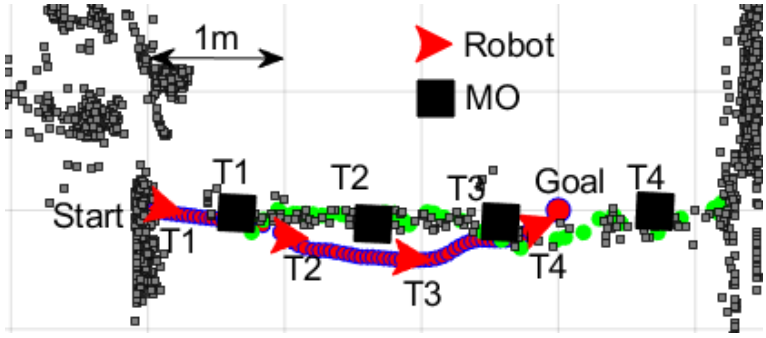


Fig. 6.7 The Robot's Path Observed in Experiment 1 with the New FLC Operating. The Locations of the Robot and Moving Obstacle are Shown at Four Different Time Instants (T1-T4)

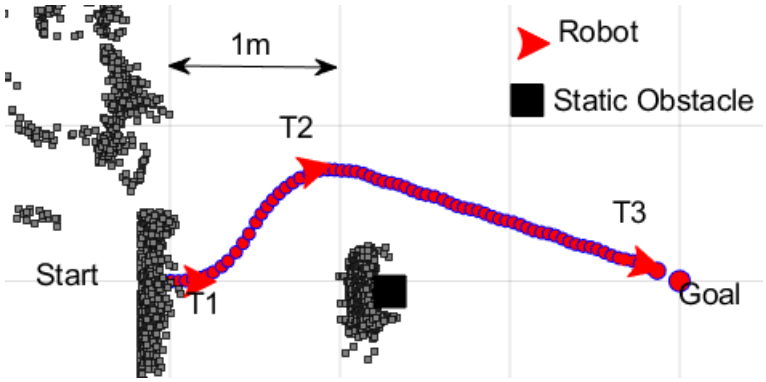


Fig. 6.8 The Robot's Path Observed in Experiment 1 with a Static Obstacle (This Configuration Creates a Local Minimum for the APF Method). The robot's Locations at Three Different Time Instances are Shown as T1-T3

6.4.2 Experiment 2: An Obstacle Moving Towards the Start Point from the Goal

The main purpose of this experiment was to identify the robot's behaviour when a fast-moving obstacle continuously challenges the robot by moving towards the robot from the goal. This experiment also demonstrated the improvements of the algorithm after introducing the new FLC. The robot's behaviour was tested with and without the new FLC within the same environment. The speed of the MO was increased to the point of collision with the robot. In all cases, the robot could reach the goal without any collision when the MO's speed was less than 10 cm/s. When the MO's speed was more than 10 cm/s the algorithm without the new FLC failed to avoid the collision. The MO's speed was maintained at 10 cm/s in the two tests presented in this chapter.

In this experiment, the robot's starting point was recorded as (0,0), the goal's location was (300,0), the MO's starting point was (370,0) and the MO's velocity was maintained at 10 cm/s.

Case 1: Basic force-shaping module without the FLC

Referring to Fig. 6.9,the robot and the moving obstacle started moving at the instant labelled 'T1'. At this moment, the distance between the robot and the MO was recorded as 295cm. As the MO is far from the robot, the robot started moving towards the goal with its maximum speed.

At the instant labelled 'T2', the robot identified that the MO is approaching the robot and started changing its direction. However, the decision was taken too late, because of the MO's higher speed (more than twice the robot's maximum speed). As a result of this delayed decision-making, the robot collided with the MO at the instant labelled 'T3' In this test, the robot did not successfully reach the goal.

Case2: New force-shaping module with the FLC

Referring to Fig. 6.10, the robot and the moving obstacle started moving at the instant labelled 'T1'. At this moment, the distance between the robot and the MO was recorded as 295 cm. Albeit at a distance, the MO was approaching the robot with a relatively high velocity, requiring the FLC to identify and quantify the speed and direction of the MO's movement. In response, the robot started moving away from the MO at time 'T1' as it identified the challenge from the MO.

At the instant labelled 'T2' the robot kept moving along the same trajectory, maintaining a distance from the MO. The robot's prior decision allowed the robot to successfully avoid the MO at the instant labelled 'T3'. The robot redirected its path towards the goal and reached the goal successfully without any collision at the instant labelled

Tab. 6.3 Summarised test results of experiment 2

Parameter	Without FLC	With FLC
Travel time	Did not reach the goal	52s
Path length	Did not reach the goal	367cm
Avg Speed	4.3cm/s	7.0cm/s

'T5'. In this experiment the robot travelled 367 cm in 52 s with an average speed of 7 cm/s, Table 6.3 presents the summarised results of these data.

Key findings of this experiment:

1. The MO was approaching the robot from the goal at high speed. Consequently, the robot was challenged by the MO, although it was not close to the robot. While the basic force-shaping module allowed the robot to move towards the goal, implementation of the new FLC immediately changed the robot's direction allowing it to deviate and avoid the path of the MO.

2. The basic force-shaping module did not consider the object's velocity in calculating the FSFs. Consequently, the challenge from the fast-moving MO was not identified sufficiently early to avoid a collision.

3. The FLC took the velocity of the MO into account, thereby allowing the robot to refine its decision-making and to avoid a collision.

4. The importance of the FLC is clearly demonstrated in assisting the robot to optimise its decisions and avoid collisions when faced with obstacles moving at high speed.

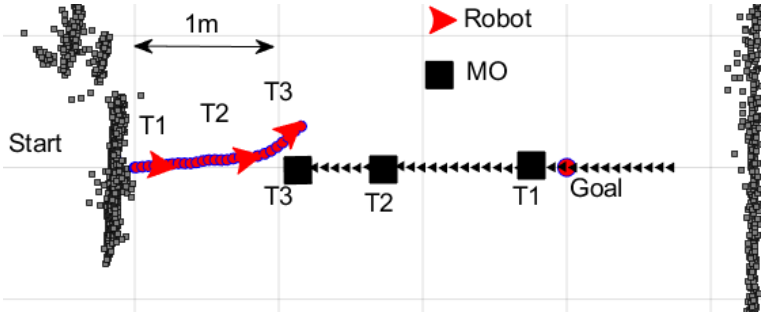


Fig. 6.9 The robot's path observed in experiment 2 without the new FCL. The locations of the robot and moving obstacle (MO) are shown at three different time-instants (T1-T3)

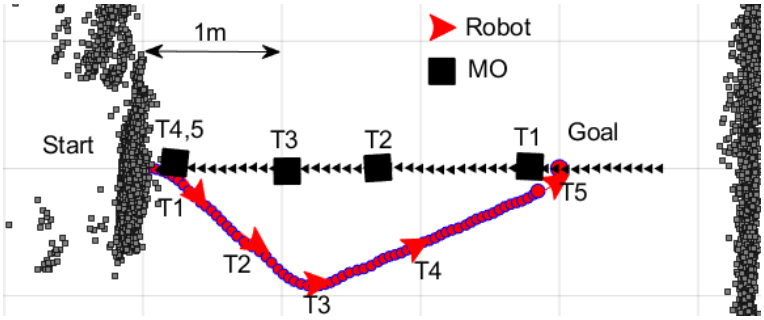


Fig. 6.10 The robot's path observed in experiment 2 with the new FLC. The locations of the robot and moving obstacle (MO) are shown at five different time-instants (T1-T5)

6.4.3 Experiment 3: An Obstacle Moving Perpendicularly Across the Robot's Path

The focus of this experiment was to identify the robot's behaviour when a moving obstacle challenged the robot by bisecting the robot's direct path in a perpendicular (-y) direction, as shown in Fig. 6.11. This experiment also showed the importance of the prediction module. The robot's behaviour was tested with and without the prediction module operating, but with all other variables held constant.

Case1: Navigation without the prediction module

Referring to Fig. 6.11, the robot and the moving obstacle started moving at the instant labelled 'T1'. At this moment, the MO was moving in the –y direction. Simultaneously the robot immediately started moving towards the goal with its maximum speed.

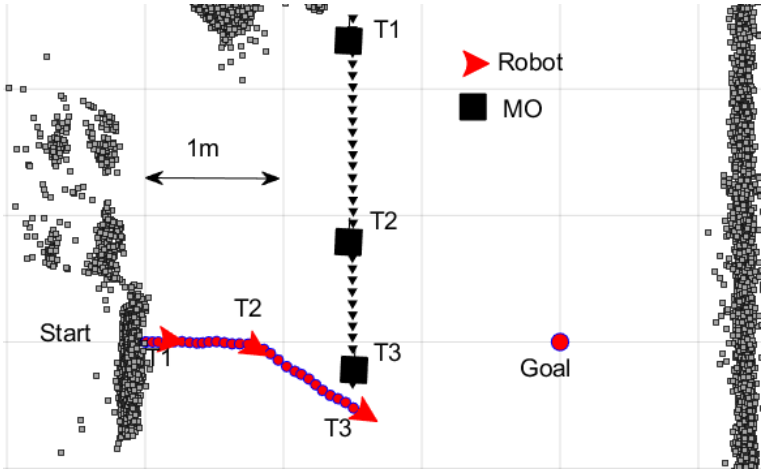


Fig. 6.11 The Robot's Path Observed in Experiment 3 without either the New FLC or the DOPP Module Operating. The Locations of the Robot and Moving Obstacle are shown at Three Different Time Instants (T1-T3)

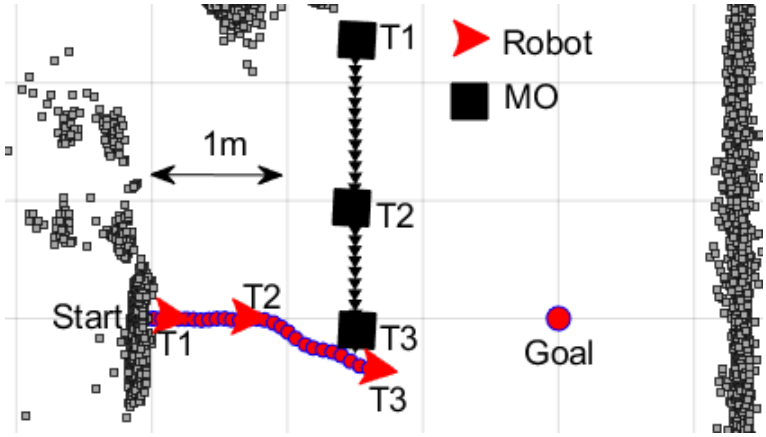


Fig. 6.12 The Robot's Path Observed in Experiment 3 with the New FLC but without the DOPP Module Operating. The Locations of the Robot and Moving Obstacle are shown at Three Sifferent Time Instants (T1-T3)

At the instant labelled 'T2' the algorithm identified that the MO was in close proximity to the robot, forcing the robot to change its direction. However, the alteration was not made early enough, which, combined with the speed of the MO, lead to a collision at the time labelled 'T3'. In this test, the robot could not reach the goal safely. This was repeated even when the new FLC was operational, as shown in Fig. 6.12.

Case2: Navigation with the prediction module

The robot and the moving obstacle started moving at the instant labelled 'T1' (Fig. 6.13). At T1, the MO was moving in the –y direction. Although the MO was far from

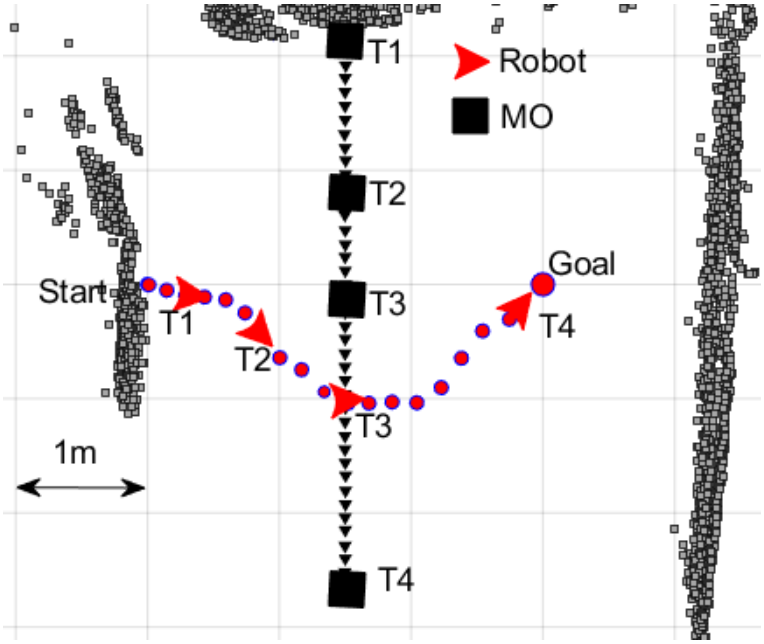


Fig. 6.13 The Robot's Path Observed in Experiment 3 with Both the New FLC and the DOPP Modules Enabled. The Locations of the Robot and the Moving Obstacle are shown at Four different Time-Instants (T1-T4)

the robot and had zero velocity component towards the robot's direction, the prediction module identified that the free space in the robot-to-goal direction was diminishing. The DOPP module, constantly checking future growing and diminishing free spaces, assisted the algorithm in optimising its decisions. Under the influence of the DOPP module, the robot started moving away from the robot to the goal path soon after starting the journey.

At the instant labelled 'T2', the robot started moving further in the –y direction as the prediction module influenced the robot with an increased weighing. The navigation system's prediction of the future environment allowed the robot to successfully avoid the MO at the instant labelled 'T3'. The robot redirected its path towards the goal and reached the goal successfully without collision (moment 'T4'). In this experiment, the robot travelled 372 cm in 88 s with an average speed of 4.2 cm/s, as shown in Table 6.4.

Key findings of this experiment:

Tab. 6.4 Summarised tTest Results of Experiment 3

Parameter	Without FLC and DOPP	With FLC and without DOPP	20cm With FCL and DOPP
Travel time	Did not reach the goal	Did not reach the goal	88s
Path length	Did not reach the goal	Did not reach the goal	372cm
Avg Speed	4.3cm/s	4.3cm/s	4.2cm/s

1. The MO was moving fast in a direction perpendicular to the robot's initial motion (the direct start-to-goal direction). Therefore, the force-shaping module identified neither any conflicting sector velocity component from the MO (as the velocity was perpendicular to the robot's initial direction) nor any obstacle between the robot and its destination.

2. Without the prediction module, the algorithm made decisions based only on the existing circumstances of the robot's surroundings. Consequently, the robot could not change its direction and kept moving towards the goal as identified at the beginning of the journey, not accounting for the spatial changes soon to be imposed by the MO. This failure ultimately led to a collision.

3. When enhanced by the prediction module, the algorithm made more accurate decisions based not only on the robot's current surroundings but also on predicted future environments. This allowed the robot to identify future growing and diminishing free-space, thereby optimising its path-making decisions and minimising the risk of collision.

4. The importance of the prediction module was confirmed, and its performance verified. It enabled the robot to make early decisions when the robot deals with increasingly challenging cases.

6.4.4 Experiment 4: Three Moving Obstacles Simultaneously Challenging the Robot and Pushing the Robot Towards a Trap

The main purpose of this experiment was to characterise the robot's behaviour when placed in a random environment with multiple moving and static obstacles incorporating traps. The experimental scenario included one static obstacle and three moving obstacles designed to challenge the robot. The first moving obstacle, MO1, changed its speed and direction rapidly to test the algorithm's dynamic capabilities. The second, MO2, was programmed to maintain a constant velocity during the experiment. The third, MO3, slightly changed its speed and direction during the experiment. For this test, the maximally functional algorithm, including prediction and FLC, was used.

Fig. 6.14 illustrates the trajectories that resulted from this scenario. The robot, MO1 and MO2 started moving at the instant labelled 'T1'. Moving Object 3 started moving 3s before the time instant T1. At this moment, MO1 and MO3 were coming towards the robot. The robot moved in the +x direction to avoid any collision with MO1 and MO3.

At the time instant T2, the robot had four possible choices:

1. Move in the (+x, -y) direction to pass in front of MO2 and reach the goal in the shortest path.
2. Reduce its speed and wait for MO2 to pass.
3. Go forward towards the goal.
4. Move in the (+x, +y) direction (towards the converging MO1 and MO2).

The new force-shaping module with the FLC operating identified the challenge from MO3 (although it was distant from the robot it was rapidly approaching the robot in the

Tab. 6.5 Summarised Test Results of Experiment 4

Parameter	Results
Travel time	133s
Path length	557cm
Avg Speed	4.2cm/s

(+x, –y) direction) and stopped the robot from moving towards the (+x, –y) direction. The prediction module also supported this decision. The second option was eliminated by the prediction module as it identified that the robot would be hit by MO1 and MO2 if the robot slowed down. The force-shaping module did not choose the third option as there remained a static obstacle between the robot and the goal. The new Agoraphilic algorithm picked the 4th option, although it is a more complicated solution. The algorithm successfully manoeuvred the robot through the time-varying free-space passage in between MO1 and MO2 during time interval T2–T4. During this time interval, the force-shaping module gave accurate directions and avoided any incorrect decisions that may have driven the robot to a trap or resulted in a collision.

At time instant T4, the robot avoided the major challenges from MO1 and MO2 and started changing its direction towards the goal. However, MO1 started crossing the robot's path again between T4 and T6. The robot slightly changed its path at time instant T5 as a result of this influence from MO1. Finally, the robot reached the goal safely at time instant T6.

In this experiment, the robot travelled 557.0 cm in 133 s with an average speed of 4.2 cm/s (Table 6.5).

Key findings of this experiment:

1. The importance of the new FLC, even in the presence of the prediction module,

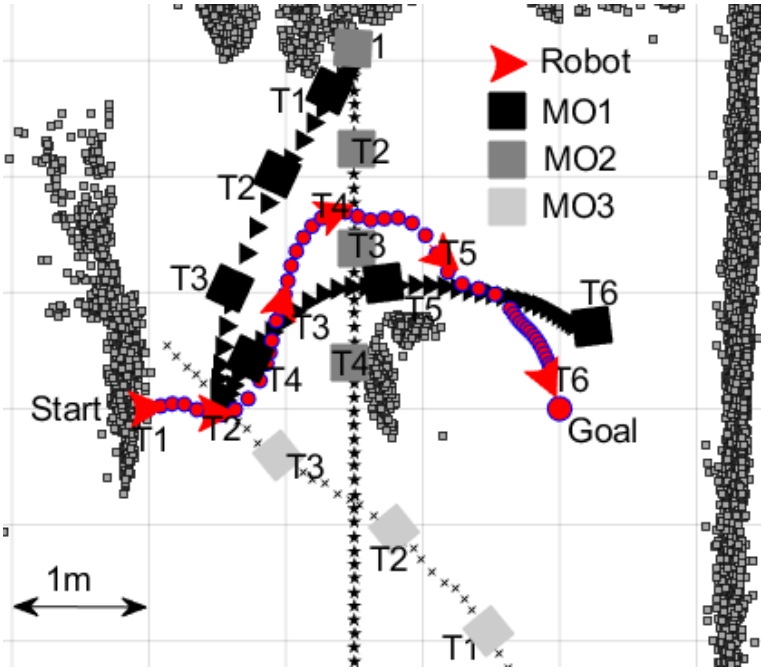


Fig. 6.14 The Robot's Observed Path in Experiment 4. The Locations of the Robot and Moving Obstacles (MO1-MO3) are shown at Six Different Time instants (T1-T6)

was clearly shown in this experiment.

2. The algorithm was able to identify and implement successful decisions in complex environments.

3. The force-shaping module could select the robot's pathway precisely to navigate through narrow time-varying free-space corridors.

4. The new Agoraphilic algorithm could successfully navigate robots in unknown dynamic environments with traps.

6.4.5 Experimental Comparison of the ANADE III with Other Recent Approaches

This subsection presents three experiments which were conducted to compare the ANADE III with three other recent navigation algorithms.

6.4.5.1 Comparison of ANADE III with Matrix-Binary Codes-Based Genetic Algorithm (MBCGA)

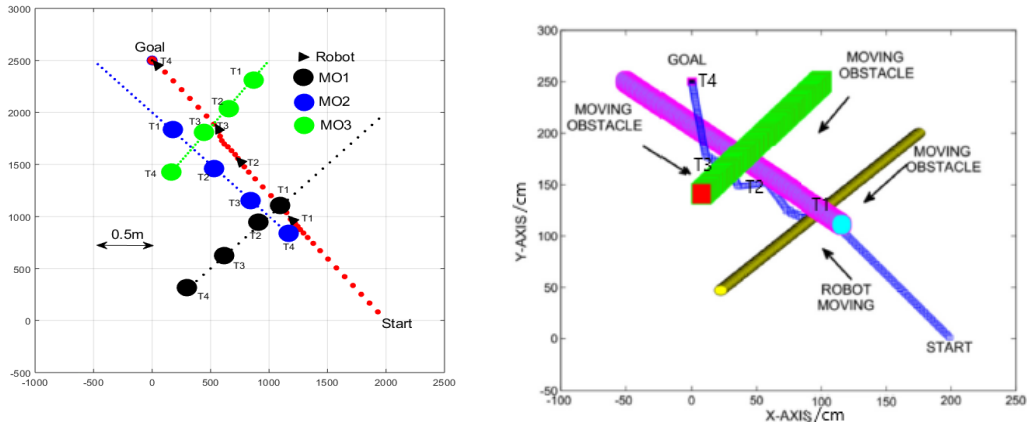
In this experiment, the environmental setup was developed to match the environment that was used in [57]. This is to compare the results under the same conditions. Three moving obstacles were used in this setup. By comparing the results of ANADE and MBCGA we notice that there are two deferent decisions at time instant T1 (Fig. 6.15):

1. ANADE III - Slowed down the robot and MO1 was enabled to pass the robot.
2. MBCGA - Changed the robot's moving direction and pass the MO1.

In this case, ANADE III reduced the robot's speed; enabling the MO1 to pass as the free space was diminishing, which was enabled through the embedded prediction algorithm. Consequently, the robot with ANADE easily avoided the challenge from MO2 at time instant T2. On the other hand, the robot with the MBCGA method had to move further in x-direction to avoid MO2. In addition, the two algorithms made different decisions at time instant T3 as well, Fig. 6.15. Overall, the decisions made by ANADE allowed the robot to use a shorter path with 66% fewer direction changes to reach the goal compared to MBCGA, Fig. 6.15.

6.4.5.2 Comparison of ANADE III with Fuzzy-Wind Driven Optimization Algorithm (FWDOP)

In this experiment, the environmental setup was developed to match the experimental setup used in [117]. Hence the experiment was conducted with two moving obstacles under similar conditions. It can be seen in this experiment that the robot with ANADE and the robot with FWDOA have taken two deferent decisions at time instant T2 (Fig.



(a) The Robot's Path with ANADE III **(b)** The Robot's Path with MBCGA [57]

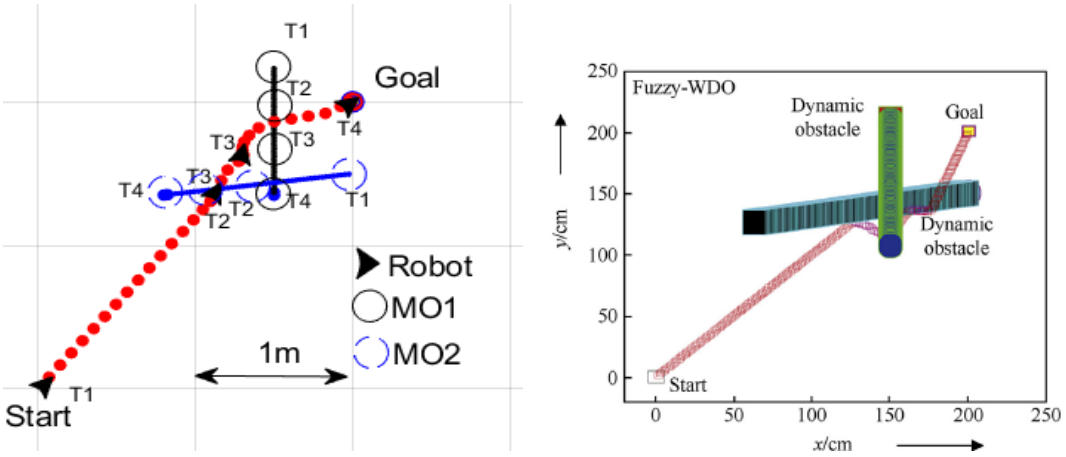
Fig. 6.15 ANADE III and MBCGA. The Robot's and Obstacles' Locations at Four Different Time-Instants, T1-T4.

6.16):

1. ANADE III - Increased the speed of the robot and passed MO2 at its front.

2. FWDOA -Turned towards (+x, -y) direction.

In this case, ANADE III increased the robot's speed; enabling the robot to pass the MO2, as the robot's free space on robot's left side was not stationary and DO2 is moving slowly compared to the robot. Therefore, the robot with ANADE algorithm managed to easily avoid the challenge imposed by MO2 at time instant T3, without any direction change. On the other hand, the robot with FWDOA [117] had to change its direction towards +x direction to avoid DO2. Also, at time instant T3, the two algorithms made different decisions.Overall the decisions made by the ANADE allowed the robot to use a shorter path with 25% fewer direction changes to reach the goal compared to the FWDOA, Fig. 6.16.



(a) The Robot's Path with ANADE III **(b)** The Robot's Path with FWDOA [117]

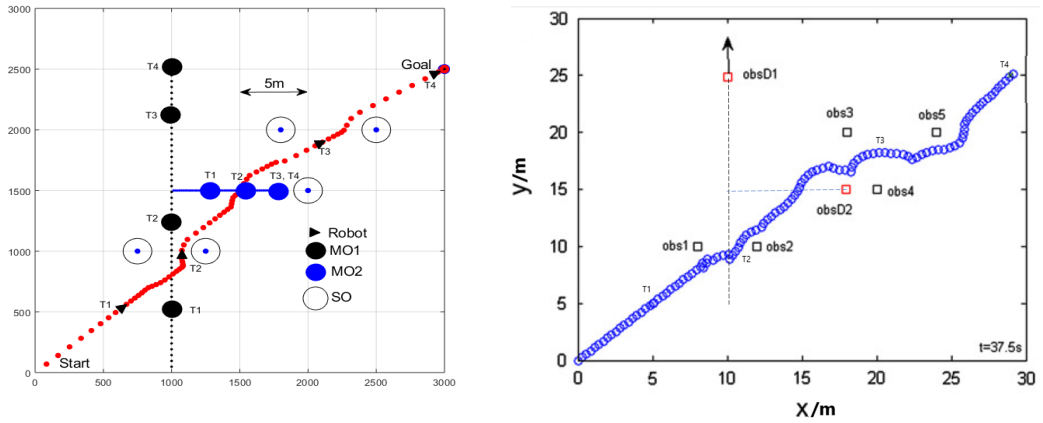
Fig. 6.16 ANADE and FWDOA. The Robot's and Obstacles' Locations at Four Different Time-Instants, T1-T4.

6.4.5.3 Comparison of ANADE III with an Improved APF Algorithm Developed Based on Transferable Belief Model (APF-TBM)

In this experiment, the environmental setup was developed according to an experimental setup that was used in [39] to compare the results under the same conditions. Two moving obstacles and five static obstacles were used in this experiment. It was observed from this benchmarking that the ANADE algorithm used a shorter path and 33% fewer direction changes to reach the goal compared to the improved APF algorithm (APF-TBM), Fig. 6.17.

6.4.6 A Qualitative Comparison Between ANADE III with Recent Navigation Approaches

A qualitative comparison between some of the recently published methods is shown in the Table 6.6. Six fundamentally important parameters for mobile robot navigation in dynamic environment were used for this comparison.



(a) The Robot's Path with ANADE III **(b)** The Robot's Path with APF-TBM [39]

Fig. 6.17 ANADE III and APF-TBM. The Robot's and Obstacles' Locations at Four Different Time-Instants T1-T4

Tab. 6.6 A Qualitative Comparison Between ANADE III with Other Approaches

Algorithm [Ref.]	Base-Concept	Tracking	Prediction	Velocity for decision making	Unknown Dynamic Environments	Experimental Validation
[57]	Genetic Algorithm	No	No	No	Yes	Yes
[117]	Fuzzy Logic	No	No	No	Yes	Yes
[39]	APF	Yes	No	Yes	Yes	Yes
[76]	Bacterial Foraging	No	No	No	Yes	No
[118]	Dynamic Window Approach	No	No	No	Yes	Yes
[52]	APF	No	No	No	Yes	No
[54]	APF	No	Yes	Yes	Yes	No
[55]	APF	No	Yes	Yes	Yes	No
ANADE III	Agoraphlic (Free-Space Attraction)	Yes	Yes	Yes	Yes	Yes

6.5 Summary

A novel controller based on fuzzy logic was developed to enhance the performances of the force-shaping module. The effectiveness of the algorithm was further improved by incorporating the velocity vectors of MOs in the decision-making. The tracking module was able to estimate the position and velocities of unknown moving obstacles accurately. Further, the algorithm uses a dynamic objects position prediction methodology to describe the robot's future environments accurately.

Four experiments and three simulations were designed and executed to demonstrate the effectiveness of the algorithm and benchmark the algorithm against other recently published methods. The experimental test results clearly demonstrated the effective-

ness of the novel fuzzy-based force-shaping module. The experiments also showed that the robot could reach its goal even in challenging environments that may include traps by means of the object tracking and dynamic object prediction modules. The inclusion of moving obstacle velocity vectors within the force-shaping methodology refined the algorithm and significantly improved its effectiveness in dynamic environments. The comparison tests of ANADE III with other methods proved ANADE III makes human-like decisions to keep minimal direction changes and shorter path to reach the goal. Further, the ANADE III, enhanced by the novel fuzzy logic controller, showed human-like intelligent decision-making behaviour in real-world experiments. The new algorithm demonstrates enhanced capabilities to successfully navigate through complicated dynamic environments without collisions and at a level of performance superior to that of the ANADE II.

However, the performances of the algorithm can be further improved in static and dynamic environments if the fuzzy logic controller can be tuned according to the changing environmental conditions. Chapter 7 introduces a self-tuning methodology for the fuzzy logic controller used in the ANADE III. This improvement has the potential to further increase the flexibility of the decision-making of the ANADE III. Further, this allows the algorithm to effectively enhance the flexibility of the robot's motion command generation module discussed in Chapter 5.

7

Chapter 7: Self-Tuning Artificial Intelligence–Based Agoraphilic Navigation Algorithm in Dynamic Environment (ANADE IV)

7.1 Introduction

The FL-based ANADE III discussed in Chapter 6 uses a fuzzy knowledge–based controller with general and unchanging membership functions. This gives a fixed Fuzzy Logic Controller (FLC) with limited flexibility to the algorithm when making navigation decisions. In mobile robot navigation in dynamic environments, robots face new or different circumstances regularly. Therefore, if the navigation algorithm can identify new circumstances and change the decision-making behaviour, the robot can adapt to new situations effectively. This adaptation increases the competence and safety of the navigation process. Therefore, a new self-tuning FLC has been introduced to further improve the performances of the algorithm and to maximise its decision-making capacity.

Based on the changed controller parameters, the adaptation methods are classified. Gain scheduling is one such adaptation method, and it is generally used to change the controller parameters (e.g., fuzzy membership functions). There are three types of gain scheduling [119, 120].

1. Changing operating conditions are used to tune the control parameters.

2. Control parameters are changed based on the control error.

3. Sugeno type rule-based controllers are used to change the controller parameters [121].

The self-tuning FLC proposed in this chapter changes the controller parameters of the FLC discussed in Chapter 6. The controller parameters are changed based on the robot's operating conditions, which are highly dependent on the robot's surrounding environment. The identified different types of the robot's environments and suitable Agoraphilic behaviours are discussed in Section 7.2.

7.2 Different Types of Agoraphilic Behaviours-Needed to Optimise the Navigation Algorithm

7.2.1 Goal Seeking

During the navigation process, there will always be situations in which the goal is in the robot's line of sight (see Fig. 7.1). In these cases, there are no obstacles between the robot and the goal. Also, there are no obstacles close to the robot and the goal line. In this kind of situation, the algorithm can give less consideration to the free space around the robot and focus more on reaching the goal quickly. However, to achieve this, the weighing system in the force-shaping module should be changed accordingly. The best possible option is changing the knowledge base of the FLC by changing the database for the first input diff discussed in Chapter 6. The proposed new θ_{diff} input database specially developed for *'goal seeking'* behaviour consists of nine fuzzy membership functions (see Fig. 7.2) derived from Equation 7.1. This compresses the membership

functions and focuses the robot more towards the goal. Further, as the robot's path is clear, the algorithm can maximise the robot's speed.

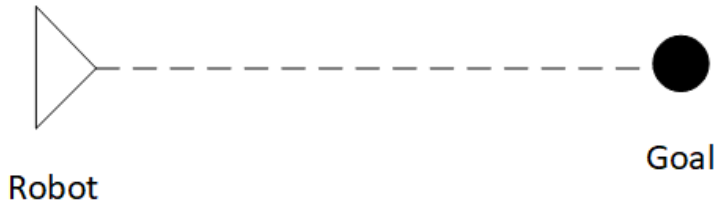


Fig. 7.1 Goal Seeking

$$
\begin{aligned}
\mu_{LRR}(\theta) &= max\{min\left(\frac{-\theta-60}{120}, 1\right), 0\} \\
\mu_{LR}(\theta) &= max\{min\left(\frac{\theta+180}{120}, \frac{-45-\theta}{15}\right), 0\} \\
\mu_{SL}(\theta) &= max\{min\left(\frac{\theta+60}{15}, \frac{-\theta-15}{30}\right), 0\} \\
\mu_{LF}(\theta) &= max\{min\left(\frac{\theta+45}{45}, \frac{-\theta}{30}\right), 0\} \\
\mu_{F}(\theta) &= max\{min\left(\frac{\theta+15}{15}, \frac{15-\theta}{15}\right), 0\} \\
\mu_{RF}(\theta) &= max\{min\left(\frac{\theta}{15}, \frac{45-\theta}{30}\right), 0\} \\
\mu_{SF}(\theta) &= max\{min\left(\frac{\theta-15}{15}, \frac{60-\theta}{15}\right), 0\} \\
\mu_{RR}(\theta) &= max\{min\left(\frac{\theta-45}{15}, \frac{180-\theta}{120}\right), 0\} \\
\mu_{RRR}(\theta) &= max\{min\left(\frac{\theta-60}{120}, 1\right), 0\}
\end{aligned}
\tag{7.1}
$$

where: $\theta \in$ {-180,180}

The characteristics of goal-seeking behaviour are as follows:

1. The robot has less ability to change its moving direction away from the goal.

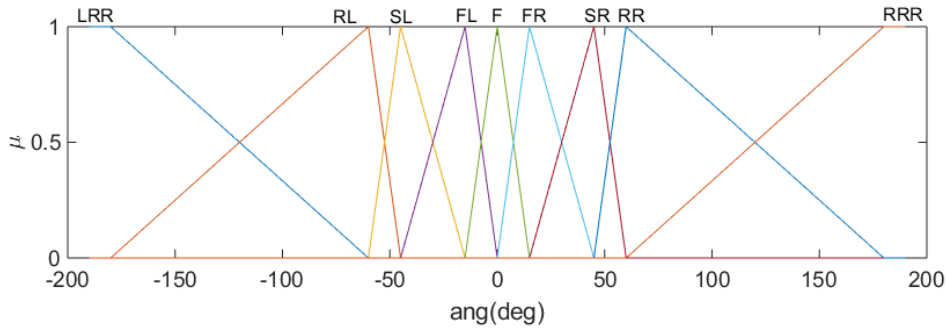


Fig. 7.2 Input Membership Function for the First Input θ_{diff} of the Fuzzy Controller for Goal-Seeking Behaviour

2. The robot has fewer direction changes

3. The robot moves faster

7.2.2 Safe Travel

During the navigation process, there will be situations in which the robot is in a cluttered environment with very limited free space (see Fig. 7.3). In this kind of situation, the algorithm can give less priority in heading towards the goal and give more importance to free spaces around the robot. This allows the robot to focus more on free space passages to move from within the cluttered area safely. However, the weighing system in the force-shaping module should be changed accordingly to achieve this. As mentioned previously, the database for the first input diff is changed. The proposed new θ_{diff} input database specially designed for *'safe travel'* behaviour comprises nine fuzzy membership functions (see Fig. 7.4) derived from Eq. 7.2. This expands the membership functions and gives more freedom to find free space.

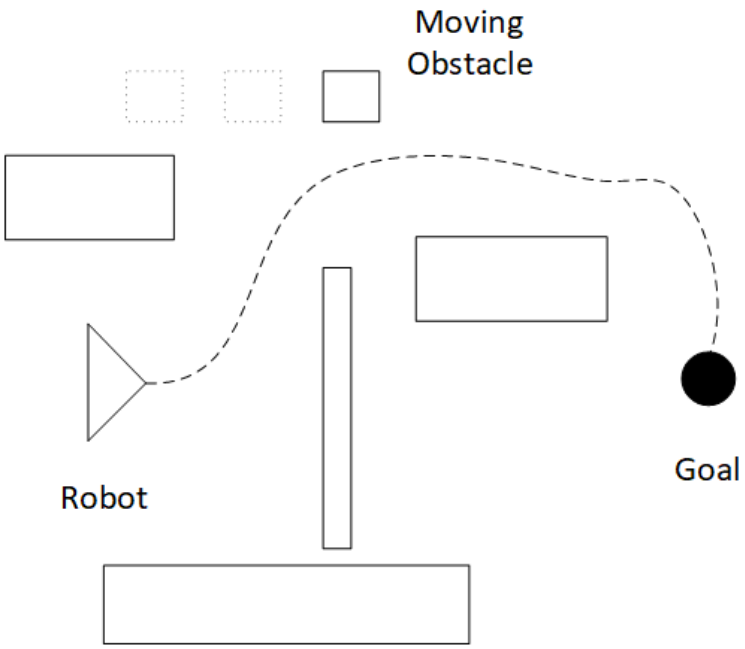


Fig. 7.3 Safe Travel

$$
\begin{aligned}
\mu_{LRR}(\theta) &= max\{min\left(\frac{-\theta-135}{45}, 1\right), 0\} \\
\mu_{LR}(\theta) &= max\{min\left(\frac{\theta+180}{45}, \frac{-90-\theta}{45}\right), 0\} \\
\mu_{SL}(\theta) &= max\{min\left(\frac{\theta+135}{45}, \frac{-\theta-45}{45}\right), 0\} \\
\mu_{LF}(\theta) &= max\{min\left(\frac{\theta+90}{45}, \frac{-\theta}{45}\right), 0\} \\
\mu_{F}(\theta) &= max\{min\left(\frac{\theta+45}{45}, \frac{45-\theta}{45}\right), 0\} \\
\mu_{RF}(\theta) &= max\{min\left(\frac{\theta}{45}, \frac{90-\theta}{45}\right), 0\} \\
\mu_{SF}(\theta) &= max\{min\left(\frac{\theta-45}{45}, \frac{135-\theta}{45}\right), 0\} \\
\mu_{RR}(\theta) &= max\{min\left(\frac{\theta-90}{45}, \frac{180-\theta}{45}\right), 0\} \\
\mu_{RRR}(\theta) &= max\{min\left(\frac{\theta-135}{45}, 1\right), 0\}
\end{aligned}
\tag{7.2}
$$

where: $\theta \in \{\text{-}180,180\}$

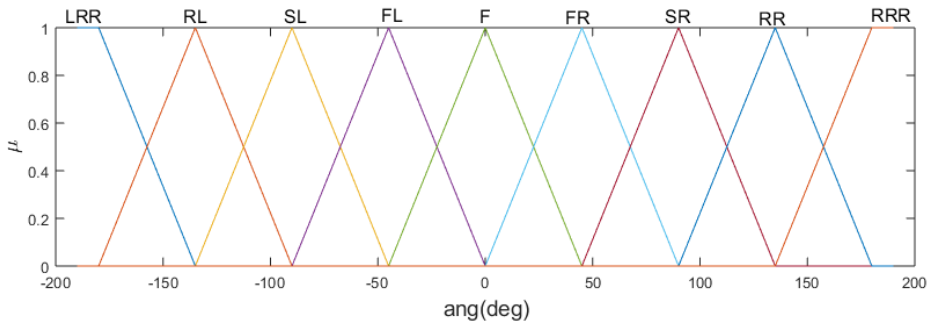


Fig. 7.4 Input Membership Function for the First Input θ_{diff} of the Fuzzy Controller for Safe Travel Behaviour

The characteristics of safe travel behaviour are as follows:

1. The robot has maximum ability to change its moving direction based on the available free space (this increases safety).
2. The robot has sudden directional changes. This may increase oscillations
3. The robot has a low linear velocity
4. The robot has a high rotational velocity

7.2.3 Normal Travel

During the navigation process, there will be situations in which the robot is in an environment with dynamic and static obstacles with a reasonable amount of free space (see Fig. 7.5). In this kind of situation, the algorithm should still consider heading towards the goal but also consider free spaces around the robot at the same time. This allows the robot to focus more on free space passages leading to the goal safely. The weighing system in the force-shaping module is changed accordingly to achieve this. The database for the first input θ_{diff} in the FLC discussed in Chapter 6 is changed accordingly. The

proposed new θ_{diff} input database specially dedicated for *'normal travel'* behaviour comprises nine fuzzy membership functions (see Fig. 7.6) derived from Eq. 7.3.

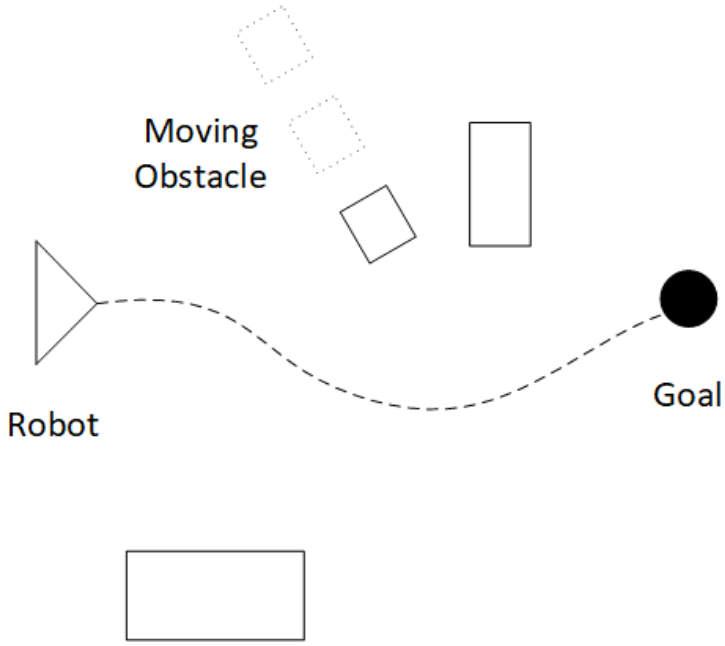


Fig. 7.5 Normal Travel

$$
\begin{aligned}
\mu_{LRR}(\theta) &= max\{min\left(\frac{-90-\theta}{90}, 1\right), 0\} \\
\mu_{LR}(\theta) &= max\{min\left(\frac{\theta+180}{90}, \frac{-60-\theta}{30}\right), 0\} \\
\mu_{SL}(\theta) &= max\{min\left(\frac{\theta+90}{30}, \frac{-30-\theta}{30}\right), 0\} \\
\mu_{LF}(\theta) &= max\{min\left(\frac{\theta+60}{30}, \frac{-\theta}{30}\right), 0\} \\
\mu_{F}(\theta) &= max\{min\left(\frac{\theta+30}{30}, \frac{30-\theta}{30}\right), 0\} \\
\mu_{RF}(\theta) &= max\{min\left(\frac{\theta}{30}, \frac{60-\theta}{30}\right), 0\} \\
\mu_{SF}(\theta) &= max\{min\left(\frac{\theta-30}{30}, \frac{180-\theta}{90}\right), 0\} \\
\mu_{RR}(\theta) &= max\{min\left(\frac{\theta-60}{30}, \frac{180-\theta}{90}\right), 0\} \\
\mu_{RRR}(\theta) &= max\{min\left(\frac{\theta-90}{90}, 1\right), 0\}
\end{aligned}
\tag{7.3}
$$

Where: $\theta \in$ {-180,180}

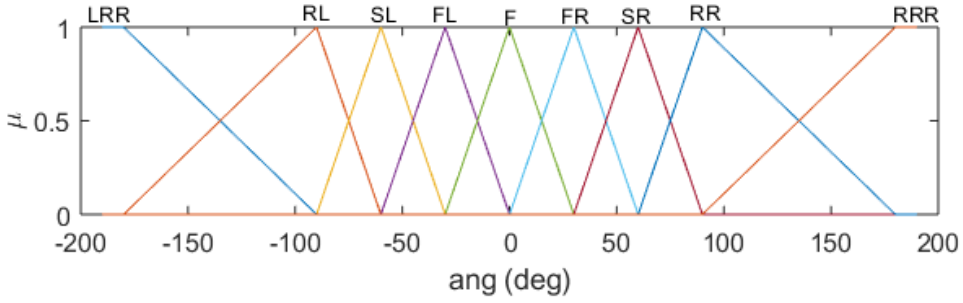


Fig. 7.6 Input Membership Function for the First Input θ_{diff} of the Fuzzy Controller for Normal Travel Behaviour

The characteristics of normal travel behaviour are as follows:

1. The robot has a medium ability to change its moving direction based on the avail-

able free space.

2. The robot has a medium linear velocity

3. The robot has a medium angular velocity

7.2.4 Safe Right Side

During the navigation process, there will be situations in which the robot is in a cluttered environment with limited free space on its left side (see Fig. 7.7). In this kind of situation, the algorithm can give less consideration to the limited free space passages on the robot's left side (as the robot may become stuck in the cluttered environment) and focus more on the free spaces around the robot's right side. This approach tends to create fewer challenging situations in future iterations. This can also be achieved by changing the knowledge base of the FLC by altering database of the input θ_{diff} discussed in Chapter 6. The proposed new θ_{diff} input database specially dedicated for *'Safe Right-Side'* behaviour comprises nine fuzzy membership functions (see Fig. 7.8) derived from Eq. 7.4. This expands the right side of the membership functions and gives more freedom to find free space on the right side.

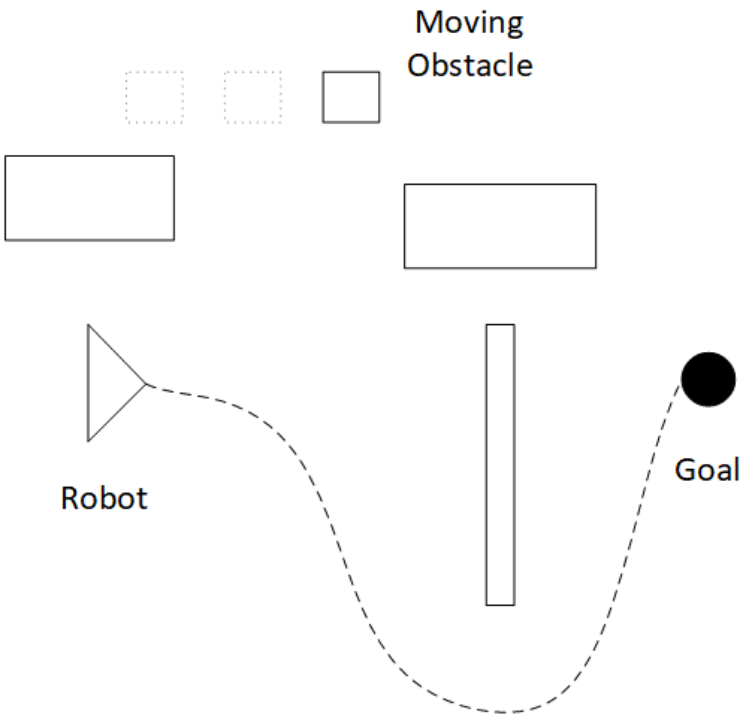


Fig. 7.7 Safe Right Side

$$
\begin{aligned}
\mu_{LRR}(\theta) &= max\{min\left(\frac{-60-\theta}{120}, 1\right), 0\} \\
\mu_{LR}(\theta) &= max\{min\left(\frac{\theta+180}{120}, \frac{-45-\theta}{15}\right), 0\} \\
\mu_{SL}(\theta) &= max\{min\left(\frac{\theta+60}{15}, \frac{-15-\theta}{30}\right), 0\} \\
\mu_{LF}(\theta) &= max\{min\left(\frac{\theta+45}{45}, \frac{-\theta}{30}\right), 0\} \\
\mu_{F}(\theta) &= max\{min\left(\frac{\theta+15}{15}, \frac{45-\theta}{45}\right), 0\} \\
\mu_{RF}(\theta) &= max\{min\left(\frac{\theta}{45}, \frac{90-\theta}{45}\right), 0\} \\
\mu_{SF}(\theta) &= max\{min\left(\frac{\theta-45}{45}, \frac{135-\theta}{45}\right), 0\} \\
\mu_{RR}(\theta) &= max\{min\left(\frac{\theta-90}{45}, \frac{180-\theta}{45}\right), 0\} \\
\mu_{RRR}(\theta) &= max\{min\left(\frac{\theta-135}{45}, 1\right), 0\}
\end{aligned}
\tag{7.4}
$$

where: $\theta \in$ {-180,180}

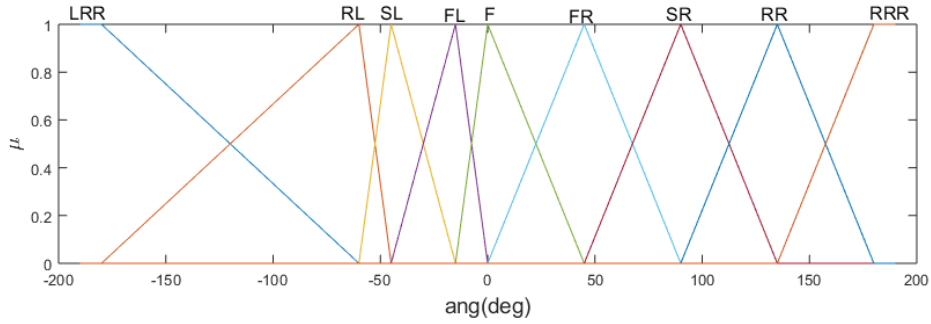


Fig. 7.8 Input Membership Function for the First Input θ_{diff} of the Fuzzy Controller for Safe Right Side Behaviour

The characteristics of safe right side behaviour are as follows:

1. The robot has maximum ability to change its moving direction based on the available free space on its right side.
2. The robot can have sudden directional changes.
3. The robot has a low linear velocity
4. The robot has a high rotational velocity

7.2.5 Safe Left Side

During the navigation process, there will be situations in which the robot is in a cluttered environment with limited free space on its right side (see Fig. 7.9). In this kind of situation, the algorithm can give less consideration to the limited free space passages on the right side (as the robot may become stuck in the cluttered environment) and focus more on the free spaces around the robot's left side. This approach tends to create fewer challenging situations in future iterations. The database of input θ_{diff} discussed in Chapter 6 is altered accordingly. The proposed new θ_{diff} input database specially

dedicated for ‘safe left side’ behaviour comprises nine fuzzy membership functions (see Fig. 7.10) derived from Eq. 7.5. This expands the left side of the membership functions and gives more freedom to find free space on the left side of the robot.

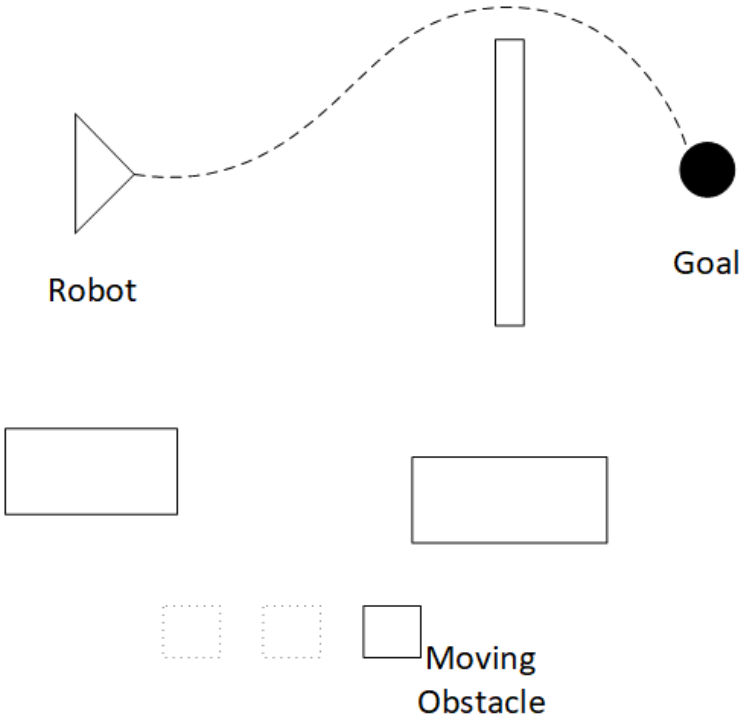


Fig. 7.9 Safe Left Side

$$
\begin{aligned}
\mu_{LRR}(\theta) &= max\{min\left(\frac{-135-\theta}{45}, 1\right), 0\} \\
\mu_{LR}(\theta) &= max\{min\left(\frac{\theta+180}{45}, \frac{-90-\theta}{45}\right), 0\} \\
\mu_{SL}(\theta) &= max\{min\left(\frac{\theta+135}{45}, \frac{-45-\theta}{45}\right), 0\} \\
\mu_{LF}(\theta) &= max\{min\left(\frac{\theta+90}{45}, \frac{-\theta}{45}\right), 0\} \\
\mu_{F}(\theta) &= max\{min\left(\frac{\theta+45}{45}, \frac{15-\theta}{15}\right), 0\} \\
\mu_{RF}(\theta) &= max\{min\left(\frac{\theta}{15}, \frac{45-\theta}{30}\right), 0\} \\
\mu_{SF}(\theta) &= max\{min\left(\frac{\theta-15}{15}, \frac{60-\theta}{15}\right), 0\} \\
\mu_{RR}(\theta) &= max\{min\left(\frac{\theta-40}{15}, \frac{180-\theta}{120}\right), 0\} \\
\mu_{RRR}(\theta) &= max\{min\left(\frac{\theta-60}{120}, 1\right), 0\}
\end{aligned}
\tag{7.5}
$$

where: $\theta \in$ {-180,180}

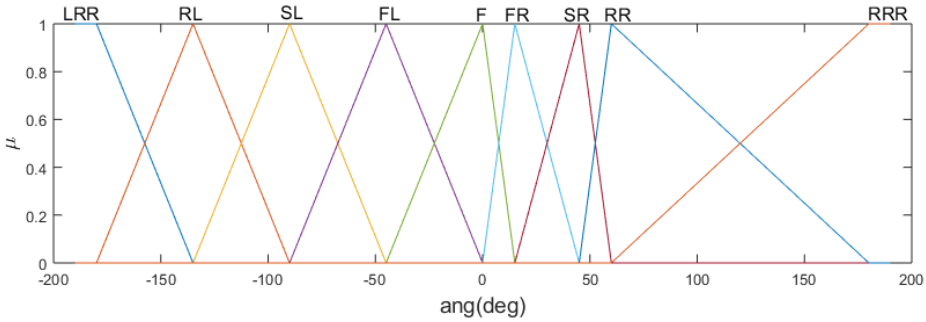


Fig. 7.10 Input Membership Function for the First Input θ_{diff} of the Fuzzy Controller for Safe Left Side Behaviour

The characteristics of safe lift side behaviour are as follows:

1. The robot has maximum ability to change its moving direction based on the avail-

able free space on its left side.

2. The robot can have sudden directional changes.

3. The robot has a low linear velocity

4. The robot has a high rotational velocity

7.3 Supervisory Controller

As discussed in Section 7.2, the robot's behaviour should be altered according to different environmental conditions. A new supervisory controller was developed to achieve this. The supervisory controller comprises three main modules:

1. Low-Resolution Space and Velocity Histogram generation module

2. Sector States Generation Fuzzy Logic Controller (SSGFLC) module

3. Motion Behaviour Selection (MBS) module

7.3.1 Low Resolution Space and Velocity Histogram

A robot-centred global map (a CGM or FGM) and estimated velocities of moving objects are taken as the inputs to this module. The robot-centred global map is divided into four sectors, as shown in Fig. 7.11. For each sector, an average sector distance ($d_{LRH,k}$) and a sector velocity $v_{LRH,k}$ are generated.

7.3.2 Sector States Generation Fuzzy Logic Controller

The SSGFLC is used to generate the sector status for all four sectors in the low-resolution histogram (see Fig. 7.12). The SSGFLC takes two sets of main inputs:

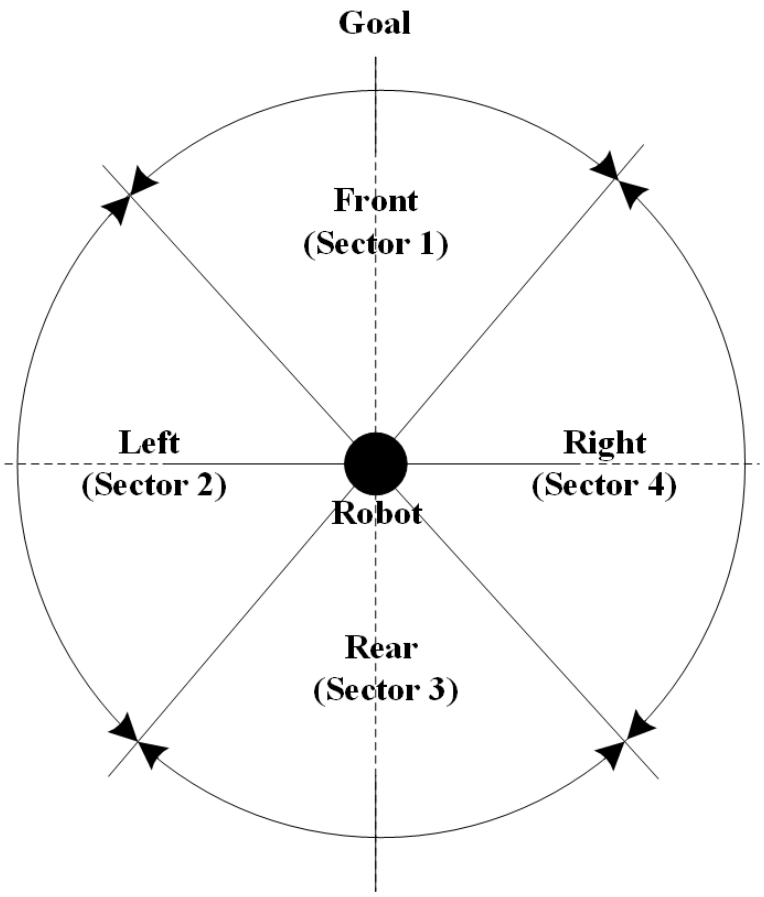


Fig. 7.11 Low Resolution Histogram

1. average sector distance ($d_{LRH,k}$ where, k={1,2,3,4})

2. sector velocity ($v_{LRH,k}$ where, k={1,2,3,4})

Three databases were designed to develop the fuzzification module. Two databases were used for the two inputs ($d_{LRH,k}$ and $v_{LRH,k}$; see Figures 7.13 and 7.14) and one for the output (the sector status SSk; see Fig. 7.16).

The database for the first input $d_{LRH,k}$ comprises three fuzzy membership functions derived from Eq. 7.6. The corresponding linguistics are C = close, M = medium and F = far.

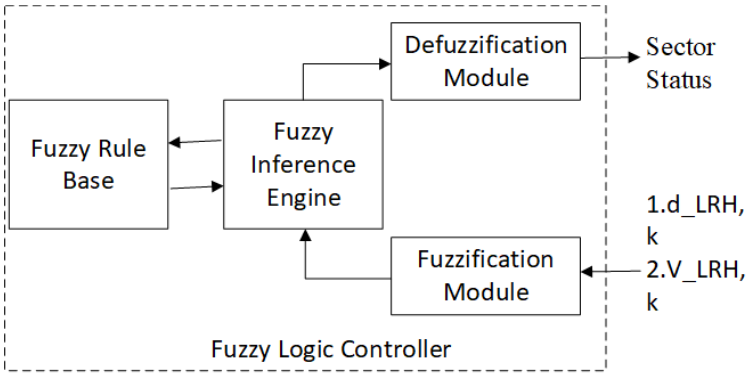


Fig. 7.12 The Block Diagram of the Sector States Generation Fuzzy Logic Controller

$$\begin{aligned}
\mu_C(d_{LRH,k}) &= max\{min\left(\frac{-d_{LRH,k}+129}{106},1\right),0\} \\
\mu_M(d_{LRH,k}) &= max\{min\left(\frac{d_{LRH,k}-23}{106},\frac{232-d_{LRH,k}}{103}\right),0\} \\
\mu_F(d_{LRH,k}) &= max\{min\left(\frac{d_{LRH,k}-129}{103},1\right),0\}
\end{aligned} \tag{7.6}$$

The database for the second input—the normalised low-resolution sector velocity ($v_{LRH,k}$)—comprises three fuzzy membership functions, as shown in Eq. 7.7. The variable $v_{LRH,k}$ is positive if the closest MO is approaching the robot. The corresponding linguistics are N = negative, Z = zero, P = positive. The simple output database is represented by three fuzzy sets, as shown in Fig. 7.15.

$$\mu_N(v_{LRH,k}) = max\{min\left(-v_{LRH,k}, 1\right), 0\}$$

$$\mu_Z(v_{LRH,k}) = max\{min\left(v_{LRH,k} + 1, 1 - v_{LRH,k}\right), 0\}$$

$$\mu_P(v_{LRH,k}) = max\{min\left(v_{LRH,k}, 1\right), 0\} \tag{7.7}$$

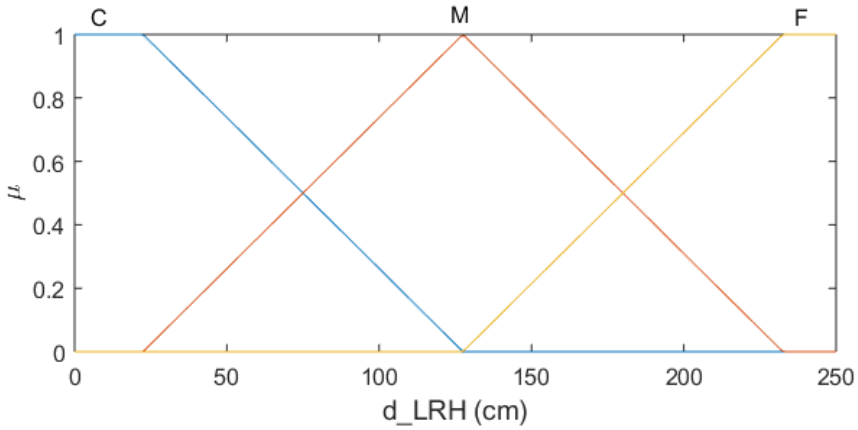


Fig. 7.13 Membership Functions of $d_{LRH,k}$

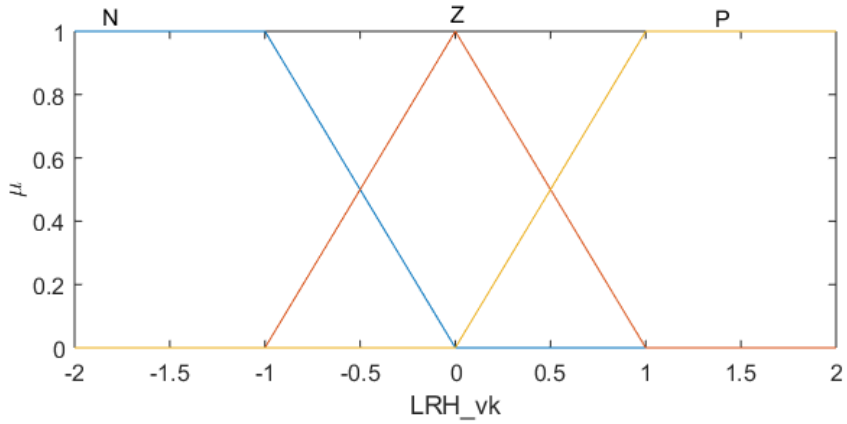


Fig. 7.14 Membership Functions of v_{LRH_k}

The fuzzy rule base was developed with nine rules (see Table 7.1). The following

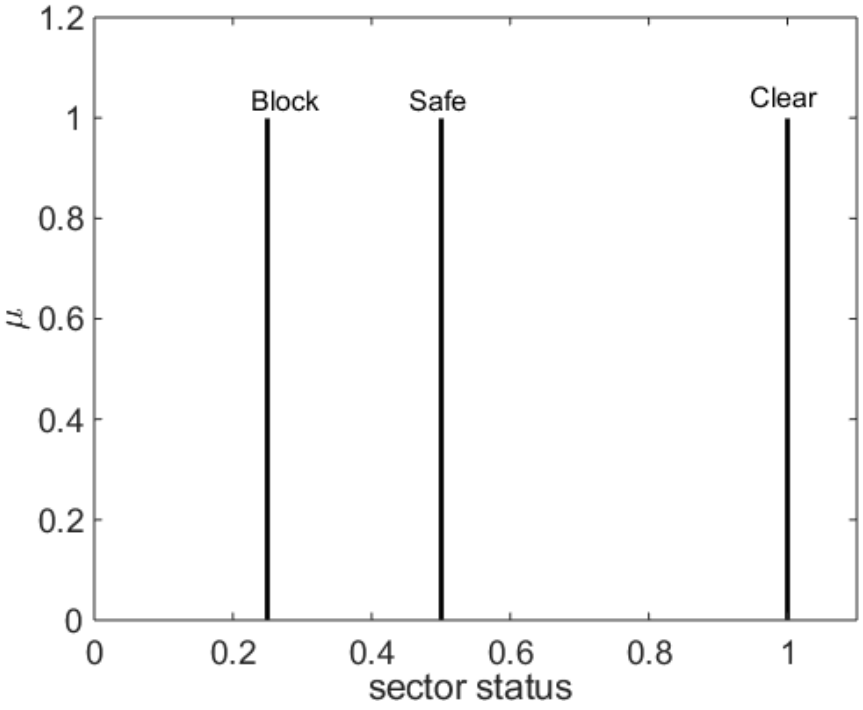


Fig. 7.15 Output Membership Functions of the Fuzzy Controller

Tab. 7.1 Structure of the Fuzzy Rule Bases Used in the Supervisory Controller

SS	C	M	F
P	Block	Block	Safe
Z	Block	Safe	Clear
N	Safe	Clear	Clear

structure was used to develop the rules.

Based on the given inputs, the fuzzy inference engine chooses four out of nine firing rules from the rule base. Then the firing power of each selected rule (α_i) is calculated via Eq. 7.8:

$$\alpha_i = min\{\mu_{L1}(d_{LRH,k}), \mu_{L2}(v_{LRH,k})\} \tag{7.8}$$

The defuzzification module uses the fuzzy rule with the maximum firing strength to

find the final sector status.

The generated sector statuses are fed into the MBS module to select the most suitable behaviour for the robot.

7.4 Motion Behaviour Selection Module

The MBS module takes the four sector status as its input and decides the robot's motion behaviour according to the robot's surroundings. Five different cases are defined in this module.

Case 1- Goal seeking: In Case 1, the main FLC is organised according to the parameters discussed in Section 7.2.1 to optimise the robot's behaviour when the goal is in the robot's line of sight.

Case 2- Safe travel: In Case 2, the main FLC is organised according to the parameters discussed in Section 7.2.2 to fully use the available free space in highly cluttered environments.

Case 3- Normal travel: In Case 3, the main FLC is organised according to the parameters discussed in Section 7.2.3 to navigate robots in an environment with dynamic and static obstacles with a reasonable amount of free space

Case 4- Right side safe: In Case 4, the main FLC is organised according to the parameters discussed in Section 7.2.4 to drive the robot more towards the right side to avoid future possibilities of becoming trapped in the cluttered left side of the robot.

Case 5- Left side safe: In Case 5, the main FLC is organised according to the parameters discussed in Section 7.2.5 to drive the robot more towards the left side to

avoid future possibilities of becoming trapped in the cluttered right side of the robot.

The appropriate case is selected according to the flowchart shown in Fig. 7.16. Case 1 is triggered if the sector status of Sector 1 (SS1) is ‘clear’. If SS1 is ‘safe’, Case 3 is triggered.

If SS1 is ‘block’, then the algorithm checks SS2 and SS4. If SS2 is ‘clear’, Case 4 is triggered. If SS2 is ‘safe’ and SS4 is either ‘safe’ or ‘block’, Case 4 is triggered. If SS2 is not ‘block’ and SS4 is ‘clear’ or ‘safe’, Case 5 is triggered. However, when SS2 and SS4 are the same but not ‘block’, and if the algorithm has triggered Case 4 or 5 in previous iteration, the same case is triggered for the next iteration (see Fig. 7.16).

If SS1, SS2 and SS4 are ‘block’, then Case 2 is triggered.

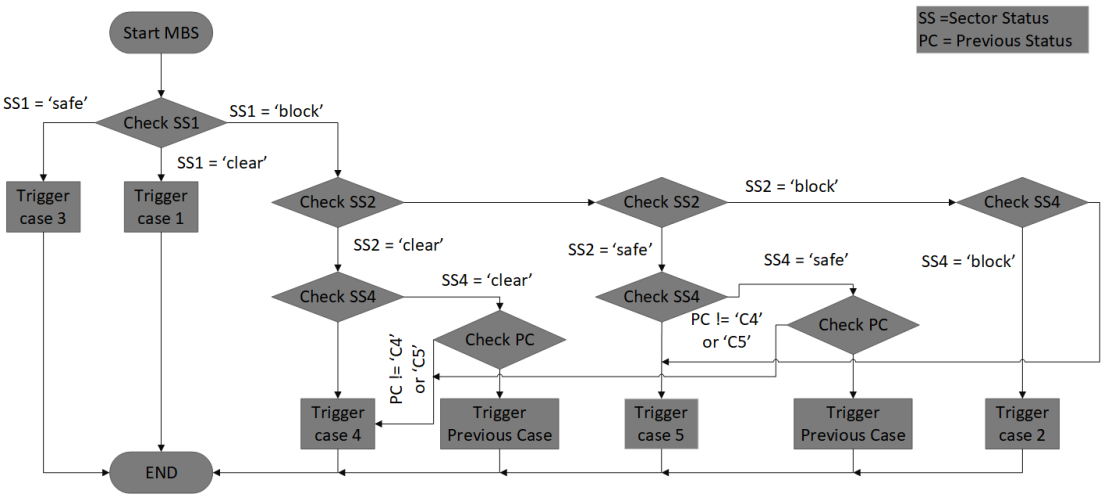


Fig. 7.16 Flowchart of the Algorithm Used in the Motion Behaviour Selection Module
Note. MBS = Motion Behaviour Selection; SS1 = sector status of Sector 1; SS2 = sector status of Sector 2; SS4 = sector status of Sector 4; PC = previous case.

7.5 Force-Shaping Module with Self-Tuning Fuzzy Logic Controller

A set of FSFs, the velocities of obstacles and a global map are taken as the inputs of the new force-shaping module. As mentioned in previous chapters, the task of the force-

shaping module is to focus the FSFs towards the goal. However, this module also has a direct influence on selecting the robot's driving direction in the corresponding iteration. Therefore, the force-shaping module is considered one of the most important parts of the algorithm.

The new force-shaping module introduced in this section also has the numerical weighing system discussed in Chapter 6. The FLC used in Chapter 6 was updated by the new self-tuning FLC, which comprises two main modules (see Fig. 7.17), as follows:

1. *Main FLC:* The main FLC used in the new adaptive FLC is mostly similar to the FLC discussed in Chapter 6. However, the knowledge base of the FLC is altered by changing the database for the first input (θ_{diff}) discussed in Chapter 6. The fuzzy membership functions representing the database of θ_{diff} is replaced by one of the fuzzy membership functions discussed in Section 7.2. The most suitable fuzzy membership function is chosen according to the command given by the supervisory controller.

2. *Supervisory controller:* The supervisory controller discussed in Section 7.3 is used as the supervisory controller of the new self-tuning FLC. The supervisory controller assesses the robot's surrounding environment and chooses the most suitable motion behaviour for the robot's navigation. Further, this module provides scaling parameters (v_i, w_i)) to scale up and down the robot's linear and angular velocities according to the selected motion behaviour.

The blocked diagram of the new force shaping module is shown in Fig. 7.17.

The following section shows some selected test results related to the theoretical improvements discussed in this chapter.

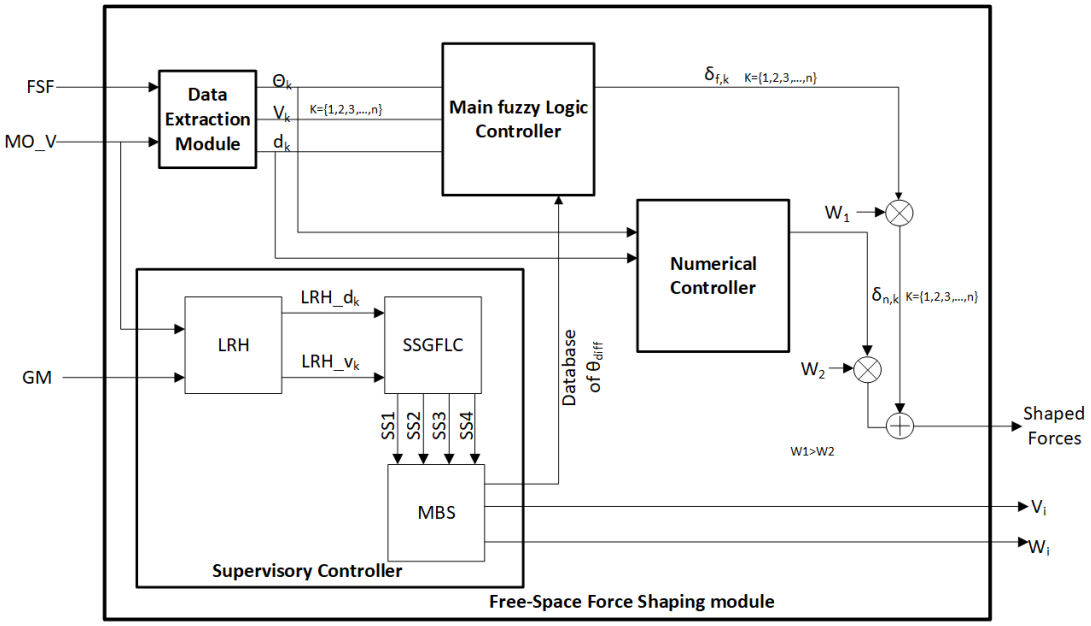


Fig. 7.17 Block Diagram of the New Force-Shaping Module

Note. FSF = Free Space Force; MO_V = velocity of moving obstacle; GM = Global Map; LRH = ; SSGFLC = Sector States Generation Fuzzy Logic Controller; SS1 = sector status of Sector 1; SS2 = sector status of Sector 2; SS3 = sector status of Sector 3; SS4 = sector status of Sector 4; MBS = Motion Behaviour Selection.

7.6 Results and Discussion

The experimental work was conducted to demonstrate and validate the performance of the algorithm with the new force-shaping module. The experiments were also designed to test the performance of the new adaptive FLC. From the range of investigations performed, a series of six basic experiments with different set-ups of FLC are presented to show the importance of the self-tuning FLC. Further, a random experiment was specifically selected to present in this chapter to validate the algorithm with the proposed new force-shaping module in this chapter:

1. experiment to show the effects of different knowledge bases in navigation

2. one MO in a cluttered environment with normal FLC and with self-tuning FLC

A safety index (SI) was introduced to quantitatively compare the safety of naviga-

tion under different knowledge bases (see Eq. 7.9).

$$SI = \frac{d_{RO}}{r_R + Safety} \tag{7.9}$$

where:

d_{RO} = robot to obstacle distance when passing

r_R = radius of the robot

$Safety = 0.5 \times r_R$

if,

SI = 0; Collision

SI $<$1; unsafe

SI $\leq$ 0.7; high danger

SI $\geq$ 1; safe

SI = 1; ideal

7.6.1 Effects of Different Knowledge Bases in Navigation

7.6.1.1 Experiment 1

The primary purpose of this experiment was to identify the robot's behaviour when an MO moved in front of the robot towards the goal (the MO's speed being less than the maximum speed of the robot). This experiment also showed the importance of the new adaptive FLC. In this experiment, four tests were conducted. In all the tests, the robot's environment was kept constant. However, in the first three experiments, the knowledge base of the main FLC was tuned to three different modes, as discussed in Section 7.2:

1. goal seeking

2. safe travel

3. normal travel

The fourth experiment shows the robot's behaviour with the new adaptive FLC.

For all the tests, the robot's starting point was recorded as (0, 0), the goal location was (300, 0), the MO's starting point was (25, 0), and the MO's velocity was maintained at 2 cm/s towards the positive x-axis.

Case 1: Navigation with 'Goal Seeking' Knowledge Base Setup

The robot started moving 10 iteration after the MO started moving (see Figure 7.18). The locations of the robot and the MO when the robot began is shown by the instant labelled T1 (see Fig. 7.18). At that moment, the distance between the robot and the MO was recorded as 47 cm (i.e., the MO was not very close to the robot). Therefore, the robot kept moving towards the goal without changing its direction.

At the instant labelled T2 (Fig. 7.18), the robot kept moving away from the MO and started overtaking the MO. At that instant, the robot's deviation from its original path was recorded as 15°. This deviation is the needed minimum deviation to overtake the MO without a collision (SI = 0.96).

At the instant labelled T3 (see Fig. 7.18), the robot successfully passed the MO and changed its direction back towards the goal. The robot kept moving towards the goal while increasing its speed (see Fig. 7.19) with maximum focus on the goal (generally, a longer distance between red dots indicates a higher velocity of the robot; see Fig. 7.18). The robot reached its destination at the instant label T4 without any collisions. In this experiment, the robot travelled 308 cm in 75 s at an average speed of 4.1 cm/s.

In this experiment, the robot did not present any oscillation. Also, the robot had minimal direction changes.

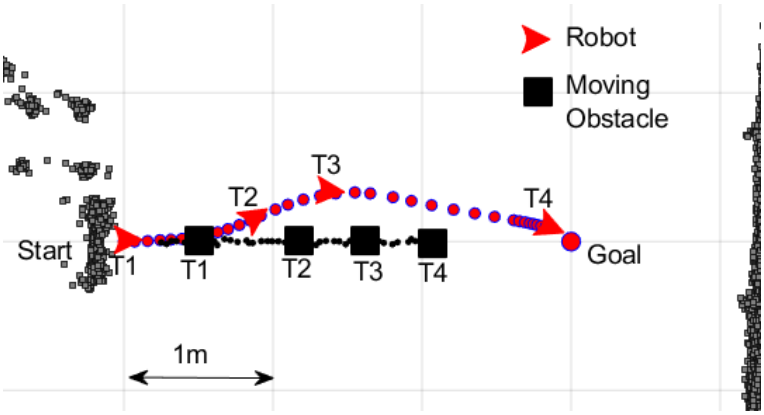


Fig. 7.18 The Robot's Path Observed in Experiment 1, Case1
Note. The locations of the robot and moving obstacle are shown at four different time instants (T1–T4).

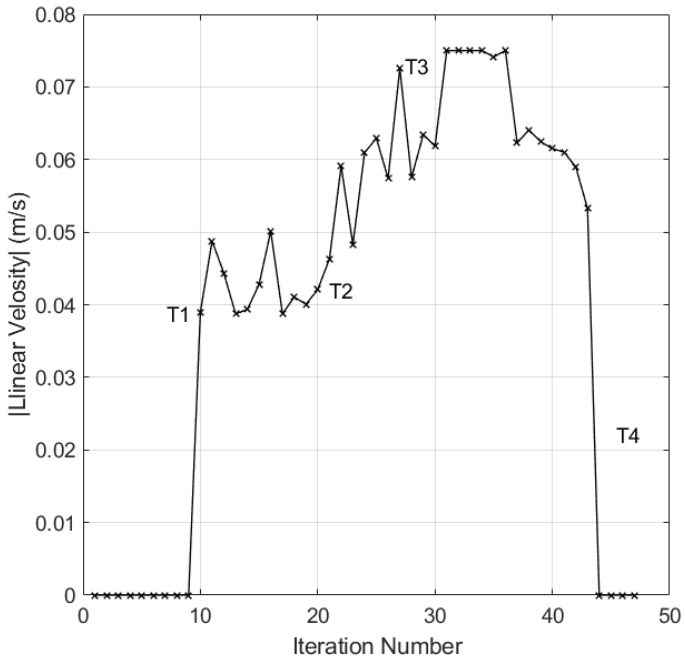


Fig. 7.19 The Robot's Speed Observed in Experiment 1, Case1

Case 2: Navigation with 'Safe Travel' Knowledge Base Setup

The robot started moving under conditions similar to those discussed in Case 1 (see Fig. 7.20). The locations of the robot and the MO when the robot began is shown by the instant labelled T1 (see Fig. 7.20). At that moment, the distance between the robot and the MO was recorded as 47 cm (i.e., the MO was not very close to the robot). However, with 'safe travel' mode, the robot immediately started moving away from the MO while reducing its speed. At that instant, the robot's deviation from its original path

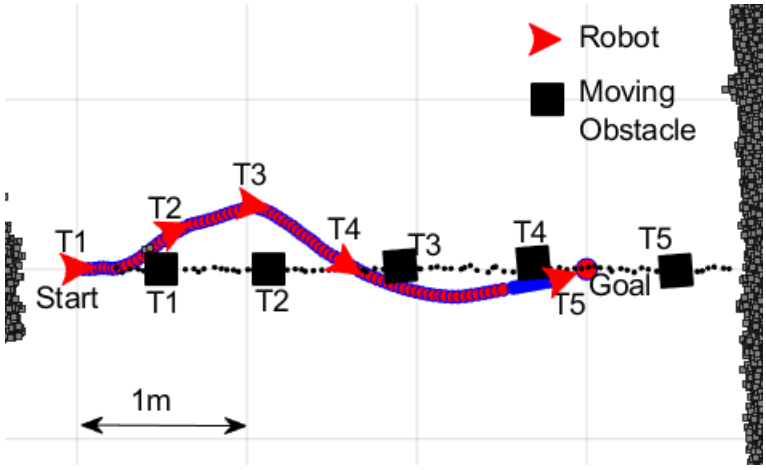


Fig. 7.20 The Robot's Path Observed in Experiment 1, Case 2
Note. The locations of the robot and moving obstacle are shown at five different time instants (T1–T5).

was recorded as 36^0.

At the instant labelled T2 (see Fig. 7.20), the robot unnecessarily kept moving away from both the MO and the goal. As discussed in Section 7.2, with 'safe travel' mode, the algorithm's focus towards the goal is minimal. The robot gets pointless freedom to go towards the free space.

At the instant labelled T3 (see Fig. 7.20), the robot changed its direction towards the goal. However, this is an overcorrection. The algorithm gave unnecessary weight to the challenge from the MO (although the MO was far away from the robot).

At the time instant T5 (see Fig. 7.20), the robot reached its destination safely without any collisions. In this experiment, the robot travelled 324 cm in 216 s at an average speed of 1.5 cm/s. In this experiment, the robot presented oscillation. Also, the robot had maximal direction changes. Also, the robot always maintained a very safe speed (see Fig. 7.21).

Case3: Navigation with 'Normal Travel' Knowledge Base setup

The initial conditions were kept the same as before (see Fig. 7.22). The start location of the robot and the corresponding location of the MO is shown by the instant labelled

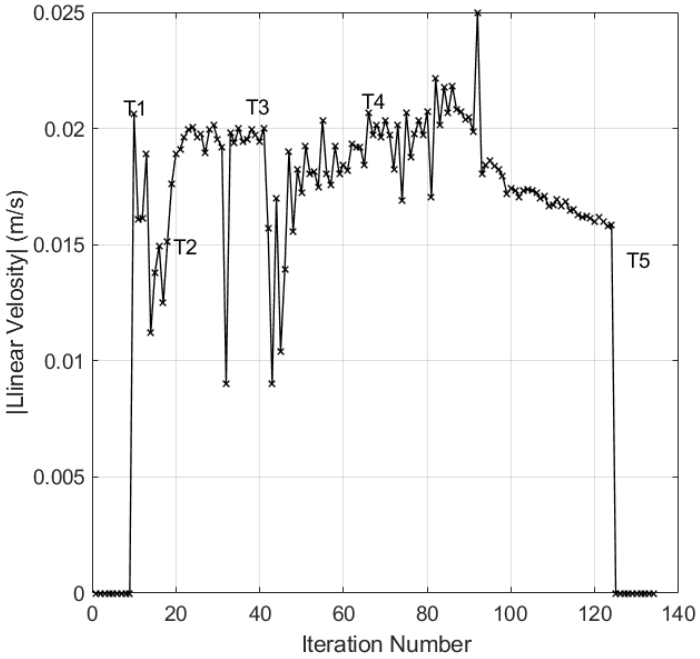


Fig. 7.21 The Robot's Speed Observed in Experiment 1, Case 2

T1. At that moment, the distance between the robot and the MO was recorded as 47 cm (i.e., the MO was not very close to the robot). In this case, the robot did not immediately change its direction but also did not wait long to change its direction away from the MO, as in Case 1 (see Fig. 7.23).

At the instant labelled T2 (see Fig. 7.22), the robot kept moving away from the MO. At that instant, the robot's deviation from its original path was recorded as 20^o. This extra 5^o was compromised to ensure that the robot overtook the MO safely.

At the instant labelled T3, the robot successfully overtook the MO and then changed its direction back towards the goal. The robot reached its destination safely at the instant label T4 without any collisions. In this experiment, the robot travelled 324 cm in 183 s at an average speed of 1.77 cm/s.

In this experiment, the robot did not present any oscillation. Also, the robot did not have unnecessary direction changes.

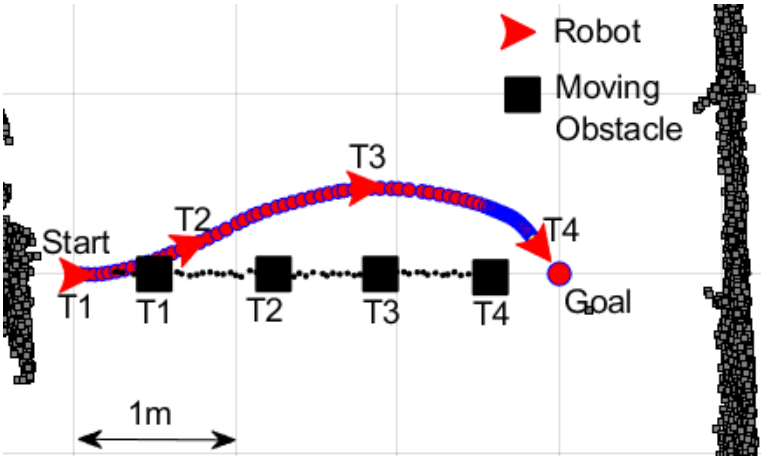


Fig. 7.22 The Robot's Path Observed in Experiment 1, Case 3
Note. The locations of the robot and moving obstacle are shown at four different time instants (T1–T4).

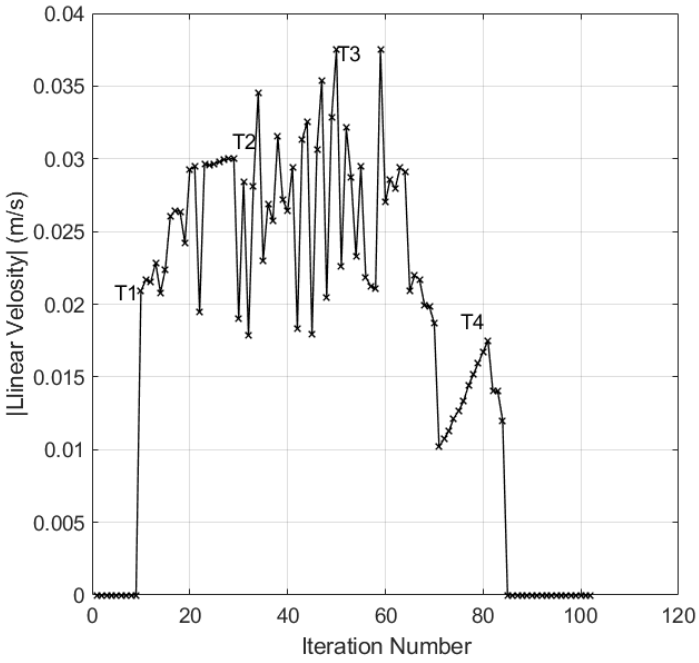


Fig. 7.23 The Robot's Speed Observed in Experiment 1, Case 3

Case 4: Navigation with Adaptive Fuzzy Logic Controller

Under the same conditions, the locations of the robot and the MO when the robot began is shown by the instant labelled T1 (see Fig. 7.24). At that moment, the distance between the robot and the MO was recorded as 47 cm (i.e., the MO was not very close to the robot). In this case, the robot did not immediately change its direction but also did not wait as long as it did in Experiment 1 to change its direction away from the MO.

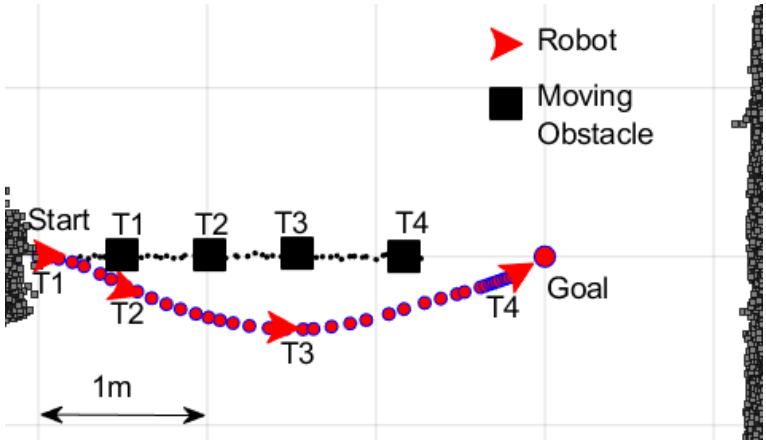


Fig. 7.24 The Robot's Path Observed in Experiment 1, Case 4
Note. The locations of the robot and moving obstacle are shown at four different time instants (T1–T4).

At the instant labelled T2 (see Fig. 7.24), the robot kept moving away from the MO. At that instant, the robot's deviation from its original path was recorded as 20^o. In this case, the algorithm compromised an extra 5^o to ensure the robot's safety.

At the instant labelled T3, the robot successfully overtook the MO and then changed its direction back towards the goal. From the start to the time instant T3, the robot mostly followed 'normal travel' behaviour.

The robot reached its destination safely at the instant T4 without any collisions. During T3 to T4, the robot followed the 'goal-seeking' behaviour with extra speed (Fig. 7.25). In this experiment, the robot travelled 314 cm in 92 s at an average speed of 3.4 cm/s.

In this experiment, the robot did not present any oscillation. Also, the robot reached the goal safely.

A summary of the experimental test results is shown in Table 7.2

The key findings of this experiment are as follows:

1. The goal-seeking mode gave the closest path, and it took the minimum time to reach the goal. However, the SI of the goal-seeking mode was less than 1 (unsafe).

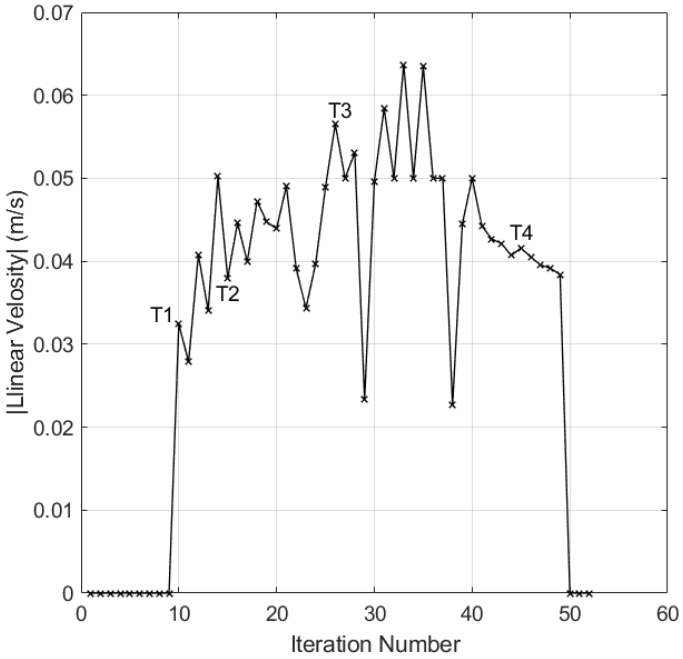


Fig. 7.25 Figure 7.25: The Robot's Speed Observed in Experiment 1, Case 4

Tab. 7.2 Summarised test results of experiment 1.

Parameter	Goal Seeking	Safe travel	Normal travel	With self-tunning FLC
Travel time	75s	216s	183s	92s
Path length	308cm	324cm	323cm	314cm
Avg. speed	4.1cm/s	1.5cm/s	1.8cm/s	3.4cm/s
Safety index	0.96	N/A (did not pass the obstacle)	1.41	1.30

2. In Case 1, it was identified that the goal-seeking mode is ideal during the period from T3 to T4.

3. The safe travel mode was the most ineffective mode in this experiment. This mode did not allow the robot to pass the slow-moving object. Also, this mode recorded the longest path and the highest time to reach the goal.

4. The safe travel mode showed oscillatory movements. The robot kept moving unnecessarily away from the MO and the goal. As discussed in Section 7.2,

with safe travel mode, the algorithm's focus towards the goal is minimal. In this experimental set-up, the robot gets pointless freedom to go towards the free space.

5. However, the safety was high with the safe travel mode.

6. The performance of the normal travel mode was between the goal-seeking and safe travel modes.

7. The five discussed behaviours were used according to the situation by the self-tuning FLC to give the best navigation path. The new self-tuning fuzzy controller balanced the safety and efficiency well to achieve the best performance.

7.6.1.2 Experiment 2

The primary purpose of this experiment was to identify the robot's behaviour when an MO moves towards the robot from the direction of the goal. This experiment also showed the importance of the new adaptive FLC. In this experiment, four tests were conducted. In all the tests, the robot's environment was kept constant. However, in the first three cases, the knowledge base of the main FLC was tuned to three different modes, as discussed in Section 7.2:

1. goal seeking

2. safe travel

3. normal travel

The fourth case shows the robot's behaviour with the new adaptive FLC.

For all the tests, the robot's starting point was recorded as (0, 0), the goal location was (300, 0), the MO's starting point was (25, 0), and the MO's velocity was maintained at 2 cm/s towards the negative x-direction.

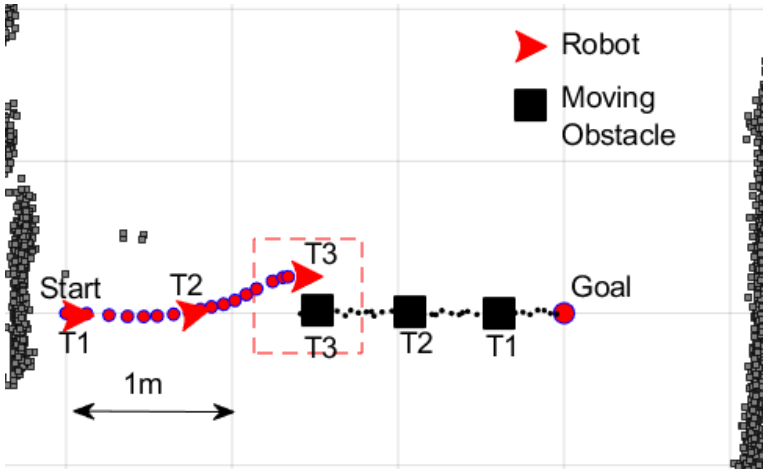


Fig. 7.26 The Robot's Path Observed in Experiment 2, Case 1
Note. The locations of the robot and moving obstacle are shown at three different time instants (T1–T3).

Case1: Navigation with Goal-Seeking Knowledge Base Setup

The robot started moving 10 iteration after the MO started moving (see Fig. 7.26). The locations of the robot and the MO when the robot began is shown by the instant labelled T1.

At the instant labelled T2, the robot started changing its direction (12^o of deviation). However, the robot did not make this decision early enough and did not turn enough to avoid the MO. The robot collided with the MO at time instant T3.

From T1 to T2, the robot kept moving unnecessarily quickly towards both the goal and the MO. This increased the relative velocity of the MO. As discussed in Section 7.2, with goal-seeking mode, the robot's driving force towards the goal is maximal. The robot gets minimal freedom to go towards the free space. This experiment proves goal-seeking behaviour is not suitable in this challenging situation.

Case 2: Navigation with Safe Travel Knowledge Base Setup

The initial environmental conditions were kept the same as they were in Case 1 (see Fig. 7.27). The locations of the robot and the MO when the robot began is shown by the instant labelled T1. At that moment, the distance between the robot and the MO

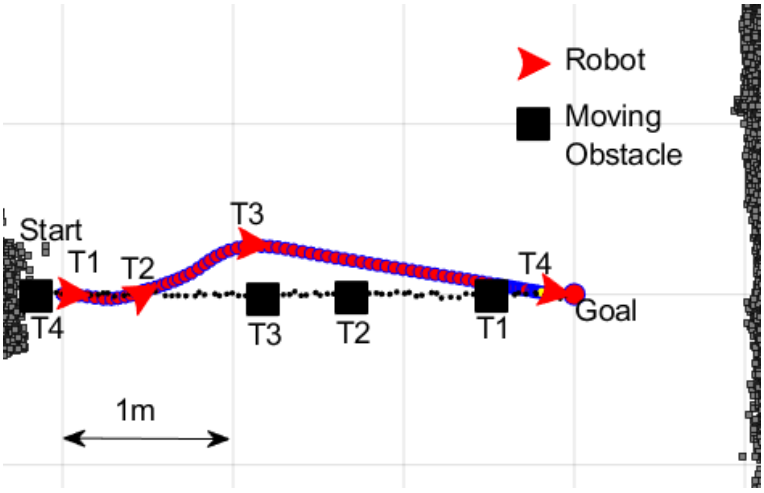


Fig. 7.27 The Robot's Path Observed in Experiment 2, Case 1
Note. The locations of the robot and moving obstacle are shown at three different time instants (T1–T3).

was recorded as 263 cm (i.e., the MO was very far from the robot). However, with safe travel mode, the robot started moving slowly.

At the instant labelled T2, the robot started moving away from the MO with a high deviation from its original path, recorded as 23^o. This allowed the robot to pass the MO at time instant T3 without any collisions (SI = 0.85).

After time instant T3, the robot did not have any obstacle in front of it. Therefore, it started moving towards the goal using the shortest path. However, safe travel behaviour limited the robot's velocity unnecessarily.

The robot kept moving unnecessarily away from both the MO and the goal. As discussed in Section 7.2, with safe travel mode, the algorithm's forces towards the goal are minimal. The robot gets pointless freedom to go towards the free space.

At the time instant T4, the robot reached its destination safely without any collisions. In this experiment, the robot travelled 308 cm in 162 s at an average speed of 1.9 cm/s. In this experiment, the robot behaved well according to the situation from time instant T1 to T3. However, during this period, the robot was driven unreasonably slowly.

Case 3: Navigation with Normal Travel Knowledge Base Setup

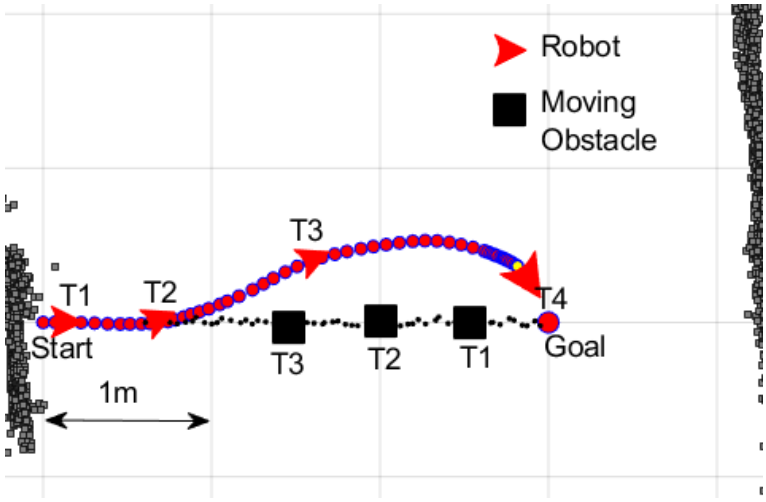


Fig. 7.28 The Robot's Path Observed in Experiment 2, Case 3
Note. The locations of the robot and moving obstacle are shown at four different time instants (T1–T4).

Under similar environmental conditions, the starting location of the robot and the corresponding MO's location is shown by the instant labelled T1 (see Fig. 7.28). In this case, the robot did not immediately change its direction but also did not wait to change its direction away from the MO for as long as it did in Case 1.

At the instant labelled T2, the robot started moving away from the MO with a high deviation from its original path, recorded as 13^o. This allowed the robot to pass the MO safely at time instant T3. In this case, the robot reached the goal safely at time instant T4. The robot travelled 311 cm in 124 s at an average speed of 2.5 cm/s.

Case 4: Navigation with Adaptive Fuzzy Logic Controller

The robot started moving 10 iteration after the MO started moving (see Fig. 7.29). The locations of the robot and the MO when the robot began is shown by the instant labelled T1. At this moment, the distance between the robot and the MO was recorded as 263 cm (i.e., the MO was very far from the robot). However, the MO was moving directly towards the robot; therefore, the adaptive fuzzy controller changed the behaviour mode to safe travel. Consequently, with safe travel mode, the robot started moving slowly.

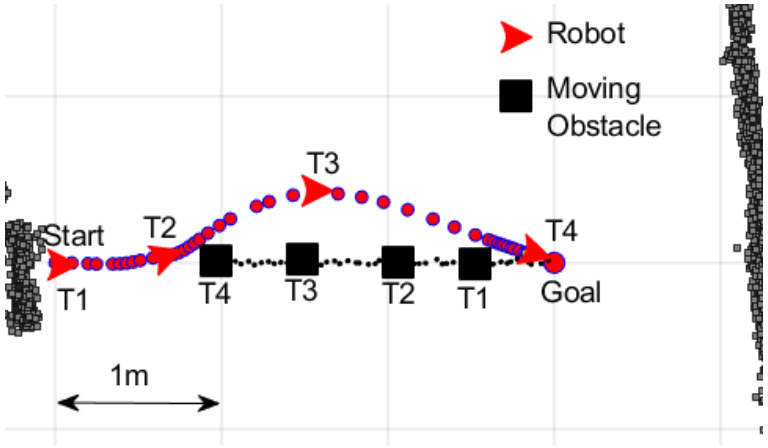


Fig. 7.29 The Robot's Path Observed in Experiment 2, Case 4
Note. The locations of the robot and moving obstacle are shown at four different time instants (T1–T4).

At the instant labelled T2, the robot kept moving away from the MO. At that instant, the robot's deviation from its original path was recorded as 23^o. This deviation and acceleration from T2 to T3 allowed the robot to pass the MO with an SI of 1.25 (safe). At the instant labelled T3, the robot successfully passed the MO and then changed its direction back towards the goal. From T2 to T3, the robot mostly followed normal travel behaviour.

The robot reached its destination safely at the instant label T4 without any collisions. From T3 to T4, the robot followed the goal-seeking behaviour. In this experiment, the robot travelled 315 cm in 90 s at an average speed of 3.5 cm/s.

In this experiment, the robot did not present any oscillation. Also, the robot had minimal direction changes.

A summary of the test results is shown in Table 7.3

The key findings of this experiment are as follows:

1. In Case 2, from T1 to T2, it was identified that the safe travel mode was ideal during that period.

2. The goal-seeking mode was the most ineffective mode in this experiment. This

Tab. 7.3 Summarised Test Results of Experiment 2

Parameter	Goal Seeking	Safe travel	Normal travel	With the self-tunning FLC
Travel time	Did not reach the gaol	162s	124s	90s
Path length	Did not reach the gaol	308cm	311cm	315cm
Avg. Speed	4cm/s	1.9cm/s	2.5cm/s	3.5cm/s
Safety Index (SI)	0	0.85	0.67	1.25

mode did not allow the robot to reach the goal.

3. In goal-seeking mode, the algorithm's focus towards the goal is maximal. The robot gets minimal freedom to go towards the free space. This experiment proves goal-seeking behaviour is not suitable in this type of challenging situation.

4. The performance of the normal travel mode was between that of the goal-seeking and safe travel modes.

5. The five discussed behaviours were used according to the situation by the self-tuning FLC to give the best navigation path. The new adaptive fuzzy controller balanced the safety and efficiency well to achieve the best performance.

6. The new self-tuning system also allowed the algorithm to control the robot's speed more effectively. At time instant T2, the FLC with safe travel mode and the self-tuning FLC influenced the algorithm to make the same decision to deviate the robot 220 away from the original path. However, the algorithm with the self-tuning FLC increased the robot's speed after time instant T2. This allowed the robot to move away from the MO quickly and pass the MO with a better SI.

7.6.1.3 Experiment 3: One Moving Obstacle in a Cluttered Environment with a Normal Fuzzy Logic Controller and with an Adaptive Fuzzy Logic Controller

The main purpose of this experiment was to characterise the robot's behaviour when placed in a cluttered environment with multiple obstacles and traps created by the obstacles. The experimental scenario included multiple SOs and one MO designed to challenge the robot. The robot's starting location was an extreme condition where the robot had to be guided carefully to move from the trapped environment. In this experiment, two tests were conducted:

1. In the first test, the algorithm used the force-shaping module with only the FLC (i.e., the force-shaping module discussed in Chapter 6).

2. In the second test, the algorithm used the force-shaping module with an adaptive FLC (i.e., the force-shaping module discussed in Chapter 7).

The robot and the MO started moving at the time instant labelled T1 (see Fig. 7.30). At that moment, the robot was in a trapped situation. At that time instant, the algorithm with only the FLC had to stick with the single behaviour mode it had. This mode did not give the extra freedom that the robot needed to avoid this trap safely. As a result, the robot ended up going further into the trap and colliding with the obstacle at time instant T3.

Conversely, the algorithm with the new self-tuning FLC had the privilege to choose a behavioural mode suitable for this situation from the following modes:

1. goal seeking

2. safe travel

3. normal travel

4. safe right side

5. safe left side

The self-tuning FLC selected the safe left side mode at the time instant T1 based on the algorithm discussed in Section 7.3. This allowed the robot to take a big safe turn towards the robot's left side, which had more free space. This allowed the robot to move to a much better place at the time instant T2.

From T2 to T4, the robot changed its behaviour to normal travel mode. However, the prediction module identified that the MO would cross the robot's path. Therefore, it did not allow the robot to move with its maximum speed from T2 to T3 (see Fig. 7.31). At time instant T3, the robot allowed the MO to cross its path. Then it started increasing its speed (see Fig. 7.31). At time instant T4, the robot changed its behaviour mode to goal seeking, as there were no obstacles in between the robot and the goal. The robot increased its velocity to its maximum and reached the goal safely at the time instant T5.

7.7 Summary

This chapter demonstrates the efficacious application of the self-tuning fuzzy logic control system for the ANADE III. The new control system allows the algorithm to identify the robot's different contexts and tune the algorithm's control parameters accordingly. This allows the algorithm to change its decision-making behaviour effectively. The new self-tuning fuzzy logic controller has a direct influence on the force-shaping module. The improved force-shaping module comprises a supervisory controller, a force-shaping fuzzy logic controller and a numerical force-shaping module. The newly added

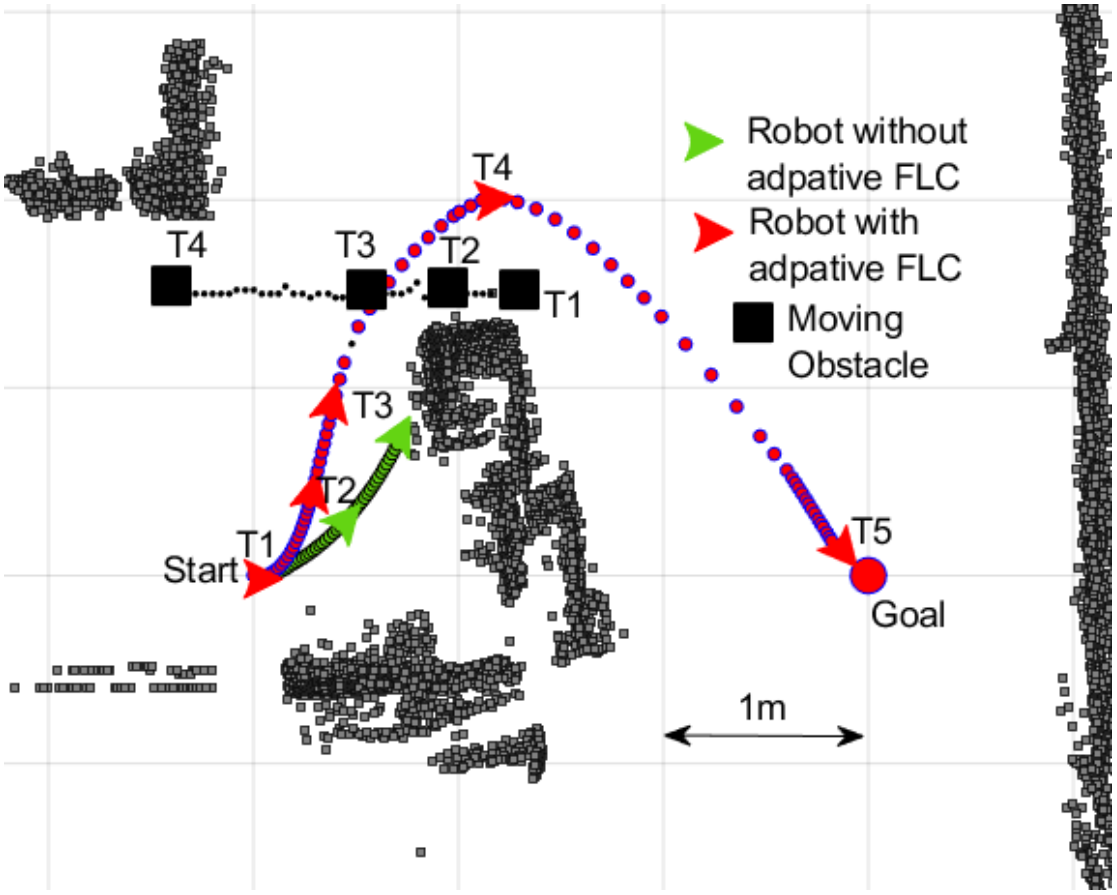


Fig. 7.30 The Robot's Path Observed in Experiment 3 with the Self-Tuning Fuzzy Logic Controller and the Normal Fuzzy Logic Controller, Discussed in Chapter 6
Note. The locations of the robot and moving obstacle are shown at five different time instants (T1–T5).

supervisory controller has its low-resolution histogram generation module, sector states generation fuzzy logic controller and motion behaviour selection module.

Three main experiments were designed and executed to demonstrate the importance of the new self-tuning fuzzy logic control system in the algorithm. In the first experiment set, four tests were conducted. In all four tests, the robot's environment was kept constant. However, in the first three cases, the knowledge base of the main fuzzy logic controller was tuned to three different modes (i.e., goal seeking, safe travel and normal travel). The fourth experiment was conducted with the self-tuning fuzzy logic controller . In the second experiment, the MO's moving direction was reversed. Finally, a random experiment was conducted with the ANADE III (as discussed in Chapter 6)

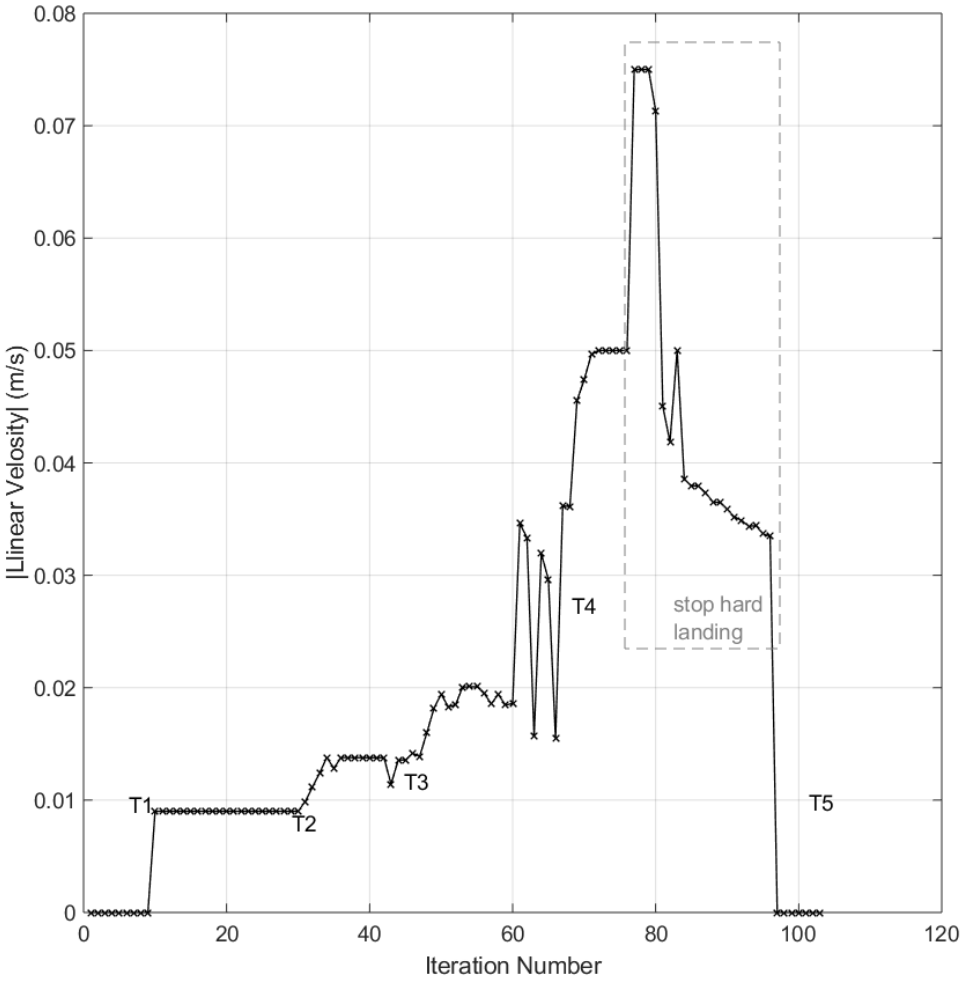


Fig. 7.31 The Robot's Speed Observed in Experiment 3 with the Self-Tuning Fuzzy Logic Controller

and the ANADE IV. The experimental test results clearly demonstrate the effectiveness of the novel self-tuning fuzzy logic controller-based force-shaping module. The experiments also showed that the new system has successfully improved the efficiency of the algorithm while increasing the robot's safety. The inclusion of the supervisory controller within the force-shaping methodology refined the algorithm and significantly improved its effectiveness under static and dynamic environments. Further, the ANADE IV algorithm, enhanced by the novel self-tuning fuzzy logic controller, showed intelligent decision-making behaviour in real-world experiments. The new al-

gorithm demonstrated enhanced capabilities to successfully navigate through complicated environments without collisions and at a level of performance superior to that of the ANADE III.

As mentioned in Chapter 2, most navigation algorithms are incapable of navigating robots in dynamic environments with a moving target. The new free space attraction -based ANADEs discussed in this and previous chapters also suffer from this problem. However, this is an important problem to address. Therefore, the ANADE IV was improved further and used to hunt moving targets under dynamic environments. Chapter 8 demonstrates the development and application of the free space attraction concept in dynamic environments with a moving goal.

8

Chapter 8: Agoraphilic Navigation Algorithm in Dynamic Environment with a Moving Goal (ANADE V)

8.1 Introduction

As mentioned in Chapter 2, only a few navigation algorithms can track and hunt a moving goal in an unknown dynamic environment. However, there is practical application where the robot has to follow or hunt a moving goal. Therefore, in this research, we further improved the ANADE to address this issue. This chapter presents the improved algorithm (ANADE V), which incorporates the FSA concept to track and hunt a moving goal in an unknown dynamic environment. It is essential to track and estimate the location of the moving goal to successfully follow and hunt it. Further, having a short-term path prediction of the moving goal will increase the efficiency of the task. Also, involving the velocity of the moving goal in the force-shaping module will increase the smart decision-making ability. However, adding another input parameter to the FLCs discussed in previous chapters will increase the size of the rule base by 500%. Also, this will reduce the flexibility and increase the computational cost of the FLC. An ML-based force-shaping module was developed to address all these issues while incorporating the velocity of the moving goal for decision-making.

8.2 The Architecture of the ANADE V

The architecture of the ANADE V is an advanced version of the architecture of the ANADE III and IV. This is also a modular-based architecture that gives the freedom to alter the module individually without altering the algorithm. The structure of this architecture increases the adaptability of the algorithm in future applications. The improved algorithm (ANADE V) comprises nine main modules:

1. Sensory Data Processing (SDP) module
2. Dynamic Obstacle Tracking (DOT) module
3. Dynamic Obstacle Position Prediction (DOPP) module
4. moving goal tracking module
5. moving goal path prediction module
6. Current Global Map (CGM) generation module
7. Future Global Map (FGM) generation module
8. Free Space Attraction (FSA) module
    (a) FSH generation module
    (b) FSFs generation module
    (c) force shaping module
    (d) Instantaneous Driving Force Component (IDFC) generation module
9. IDFC weighting module
10. Rvobot's Motion Commands (RMC) weighting module

These modules are iteratively used in the proposed algorithm to navigate the robot towards a moving goal in a dynamic environment (see Fig. 8.1). In this improved algorithm, two new modules are introduced, and a major modification was made to the force-shaping module:

1. moving goal tracking module

2. moving goal path prediction module

Some slight changes were also made to the CGM and the FGM generation modules. All the other modules are similar to those used in the ANADE IV. The development of those modules was discussed in Chapters 4–7.

The development of the ML-based force-shaping module and the two new modules and the changes made to the CGM and FGM generation modules are discussed in the following sections.

8.3 The New Modules Used in the ANADE V to Track and Hunt a Moving Goal

8.3.1 Moving Goal Tracking Module

The main task of this module is to track the moving target and goal and estimate its velocity and location. The goal tracking module takes the data from the sensory system as its input (see Fig. 8.2). This module also runs a KF-based algorithm to produce its outputs (i.e., the positions and velocity of the moving goal).

These outputs are fed into the CGM generation module and the goal path prediction module. In the tracking module, the kinematic model of the moving goal or target is defined in Eq. 8.1.

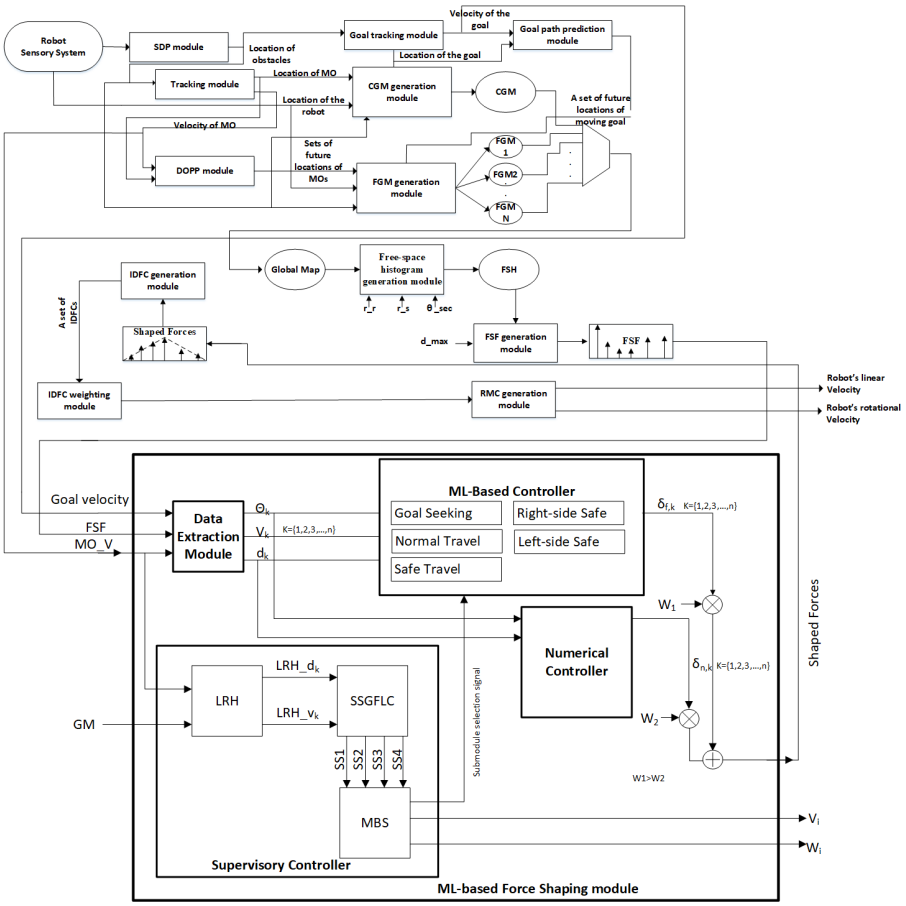


Fig. 8.1 Block Diagram of the ANADE V

Note. SDP = Sensory Data Processing; MO = moving obstacle; CGM = current global map; DOPP = Dynamic Obstacle Position Prediction; FGM = future global map; IDFCs = instantaneous driving force components; FSH = Free Space Histogram; FSF = Free Space Force; RMC = robot's motion command; ML = machine learning; GM = ; LRH = ; SSGFLC = Sector States Generation Fuzzy Logic Controller; SS1 = sector status of Sector 1; SS2 = sector status of Sector 2; SS3 = sector status of Sector 3; SS4 = sector status of Sector 4; MBS = Motion Behaviour Selection.

$$\begin{bmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \dot{x}_{k-1} \\ \dot{y}_{k-1} \end{bmatrix} + \begin{bmatrix} a_x \times \frac{dt^2}{2} \\ a_y \times \frac{dt^2}{2} \\ a_x \times dt \\ a_y \times dt \end{bmatrix} \times u(t) + w_k \tag{8.1}$$

$$\begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} + v_k$$

The prior estimation (state estimation based on the previous state estimation) and globalised sensory data (measurements ($x_{k,m}$,$y_{k,m}$) are combined using the filter shown in Eq. 8.2 to derive the optimal state estimations for the moving goal in each iteration.

$$\begin{aligned} \begin{bmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{bmatrix} = & \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \dot{x}_{k-1} \\ \dot{y}_{k-1} \end{bmatrix} + \\ & \begin{bmatrix} a_x \times \frac{dt^2}{2} \\ a_y \times \frac{dt^2}{2} \\ a_x \times dt \\ a_y \times dt \end{bmatrix} + k_k \left\{ \begin{bmatrix} x_{k,m} \\ y_{k,m} \end{bmatrix} - \right. \\ & \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}^T \left(\left(\begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \dot{x}_{k-1} \\ \dot{y}_{k-1} \end{bmatrix} \right. \right. \\ & \left. \left. + \begin{bmatrix} a_x \times \frac{dt^2}{2} \\ a_y \times \frac{dt^2}{2} \\ a_x \times dt \\ a_y \times dt \end{bmatrix} \right) \right\} \end{aligned} \tag{8.2}$$

In this expression k_k (Kalman Gain) is found by using Eq. 8.3;

$$k_k = p_k \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}^T \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}^T p_k \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \right)^{-1} \tag{8.3}$$

Where:

$$p_k = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} p_{k-1} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ dt & 0 & 1 & 0 \\ 0 & dt & 0 & 1 \end{bmatrix} + Q \tag{8.4}$$

At each iteration following the optimal state estimation of every moving goal, p_k is updated as shown in Eq.8.5.

$$p_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - k_k \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ dt & 0 & 1 & 0 \\ 0 & dt & 0 & 1 \end{bmatrix} \times p_{k-1} \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ dt & 0 & 1 & 0 \\ 0 & dt & 0 & 1 \end{bmatrix} + Q \tag{8.5}$$

The updated p_k is used as $p_{(k-1)}$ in Eq.8.4 to re-calculate the new p_k in the next iteration.

where:

$v_k \sim$ N(0,R) (R=measurement error covariance)

$w_k \sim$ N(0,Q) (Q=process noise)

$p \stackrel{\text{def}}{=}$ error covariance

$(x_k, y_k) \stackrel{\text{def}}{=}$ estimated position of the moving goal

$(x_{km}, y_{km}) \stackrel{\text{def}}{=}$ measured position of the moving goal

$(a_x, a_y) \stackrel{\text{def}}{=}$ acceleration of the moving obstacle

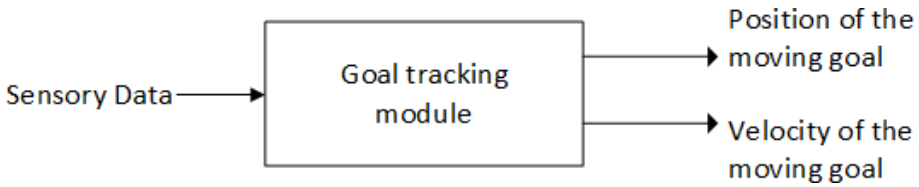


Fig. 8.2 The input and outputs of the goal tracking module.

8.3.2 Moving Goal Path Prediction Module

The main functionality of the goal path prediction module is to estimate the future locations of the moving target/ goal. This helps the algorithms to adjust the robot's trajectory to reach the moving goal effectively.

The goal path prediction module takes the current estimated states (position and velocity) of the moving goal from the goal tracking module. This creates a set of future locations of the moving goal (see Fig. 8.3). These forecasted locations are used to generate the FGMs.

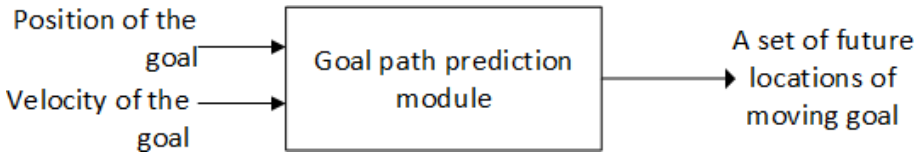


Fig. 8.3 The Inputs and Output of the Goal Path Prediction Module

Future states (positions and velocities) of the moving goal are predicted using the

prediction model described in Eq. 8.6. The predictions are updated at each iteration, and new predictions are made according to the updated state data.

$$\begin{bmatrix} x(t+n) \\ y(t+n) \\ dx(t+n)/dt \\ dy(t+n)/dt \end{bmatrix} = \begin{bmatrix} 1 & 0 & nT & 0 \\ 0 & 1 & 0 & nT \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x(t) \\ y(t) \\ dx(t)/dt \\ dy(t)/dt \end{bmatrix} + \begin{bmatrix} \frac{(nT)^2}{2} \times \frac{d^2x(t)}{dt^2} \\ \frac{(nT)^2}{2} \times \frac{d^2y(t)}{dt^2} \\ nT \times \frac{d^2x(t)}{dt^2} \\ nT \times \frac{d^2y(t)}{dt^2} \end{bmatrix} \tag{8.6}$$

where:

$x(t) \stackrel{\text{def}}{=}$ position in x direction

$y(t) \stackrel{\text{def}}{=}$ position in y direction

$T \stackrel{\text{def}}{=}$ sample period

$n \stackrel{\text{def}}{=}$ sample number

8.3.2.1 Current Global Map Generation Module

The CGM generation module generates a map of the robot's environment with respect to the world axis system. This modified module takes four main inputs:

1. locations of SO with respect to the robot's axis system from the sensory system
2. current locations of MO from the MO tracking module
3. current location of the goal/target estimated from the goal tracking module
4. current location of the robot from the sensory system

The CGM generation module processes these four inputs and creates a map of the

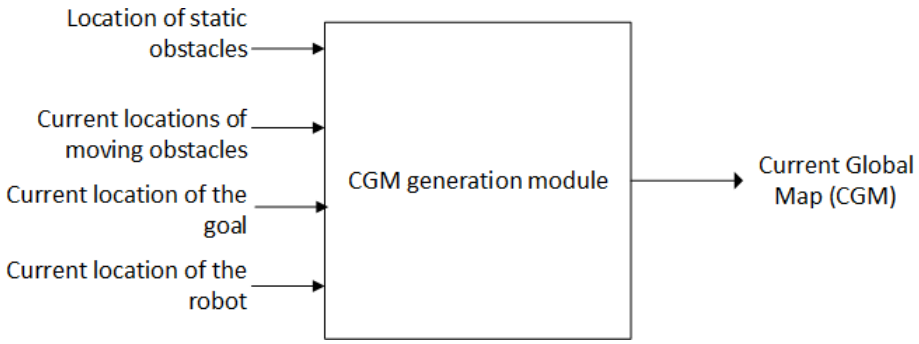


Fig. 8.4 The Inputs and Output of the Current Global Map Generation Module

environment with respect to the global axis system (see Fig. 8.4). This map (i.e., the CGM) is fed into the FSA module, as discussed in Chapter 5.

8.3.2.2 Future Global Map Generation Module

As discussed in previous chapters, the main task of the FGM generation module is to forecast the robot's future environment set-ups and generate a set of maps called FGMs. This module takes three main inputs (see Fig. 8.5),

1. the current location of the robot (taken from the sensory system)
2. sets of future locations of MOs (taken from the obstacle path prediction module)
3. a set of future locations of the goal (taken from the goal path prediction module)

The FGM at iteration 'N' is generated by combining the estimated location of the robot at the nth iteration with forecasted locations of MOs and SOs and the goal at the nth iteration. If the FGM generation module develops predicted maps until the t + n iteration, there will be n-1 FGMs. The FGMs are updated at each iteration, with corrections made to position estimations.

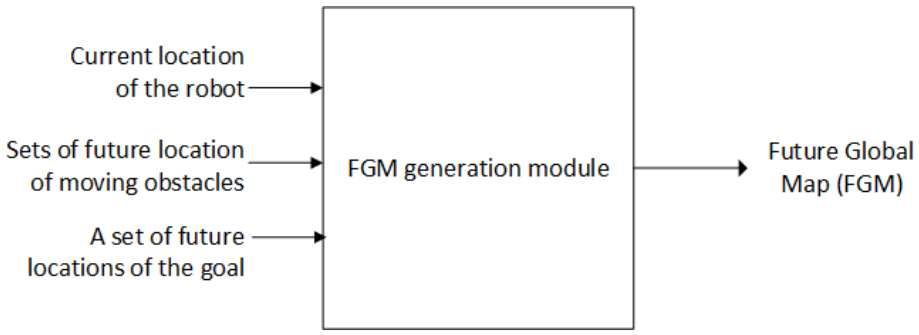


Fig. 8.5 The Inputs and Output of the Future Global Map Generation Module

8.3.3 Machine Learning–Based Force-Shaping Module

The force-shaping modules used in Chapters 6 and 7 do not consider the velocity of the moving goal when shaping the FSFs. The novel ML-based force-shaping module was developed to address this problem.

A set of FSFs, the velocity of the moving goal and of the obstacles, and a global map are taken as the inputs of the new ML-based force-shaping module. As mentioned in previous chapters, the task of the force-shaping module is to focus the FSFs towards the goal. However, this module also has a direct influence on selecting the robot's driving direction in the corresponding iteration. Therefore, the force-shaping module is considered one of the most important parts of the algorithm. The new ML-based force-shaping module introduced in this chapter replaces the main FLC in the self-tuning force-shaping module discussed in Chapter 7 by an ML-based controller. The new self-tuning ML-based force-shaping module comprises two main modules (see Fig. 8.6).

1. supervisory controller: The supervisory controller discussed in Section 7.3 is used as the supervisory controller of the new self-tuning ML-based force-shaping module. The supervisory controller assesses the robot's surrounding environment and chooses the most suitable motion behaviour for the robot's navigation. This

module works similarly to the supervisory controller discussed in Chapter 7.

2. ML-based controller: The ML-based controller used in the new self-tuning force-shaping module is developed to improve the performance of the force-shaping module while incorporating the velocity of the moving goal. This module comprises five ML-based submodules. The development process of the ML-based controller is discussed in the following section.

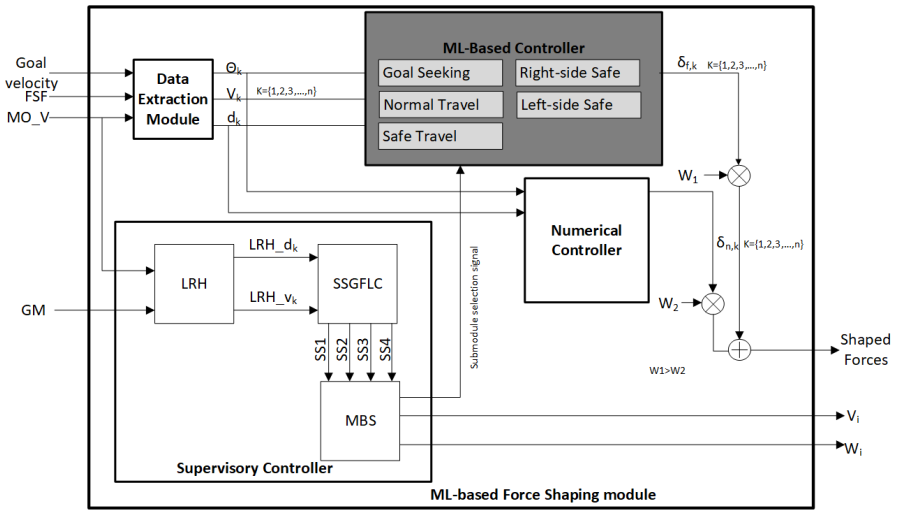


Fig. 8.6 Block Diagram of the New Machine Learning–Based Force-Shaping Module
Note. ML = machine learning; GM = ; LRH = ; SSGFLC = Sector States Generation Fuzzy Logic Controller; SS1 = sector status of Sector 1; SS2 = sector status of Sector 2; SS3 = sector status of Sector 3; SS4 = sector status of Sector 4; MBS = Motion Behaviour Selection.

8.4 Machine Learning–Based Controller for Force-Shaping Module

The main challenge associated with ML was generating good training and testing datasets. The trained module with the predetermined datasets should be able to successfully replace the FLC in the force-shaping module while incorporating the velocity of the moving goal for force shaping. Training and testing datasets were developed for the five main Agoraphilic behaviours discussed in Chapter 7:

1. goal seeking

2. normal travel

3. safe travel

4. right side safe

5. left side safe

The development process of these datasets is discussed in the following sections.

8.4.1 Training and Test Data Generation

Five dataset generation engines were developed for the training and testing dataset generation process. These datasets cover the five main Agoraphilic behaviours. The results received from the dataset generation engines were modified based on the domain knowledge to obtain the final training and testing datasets (see Fig. 8.7). The FLCs discussed in Chapters 6 and 7 were used to develop the dataset generation engines.

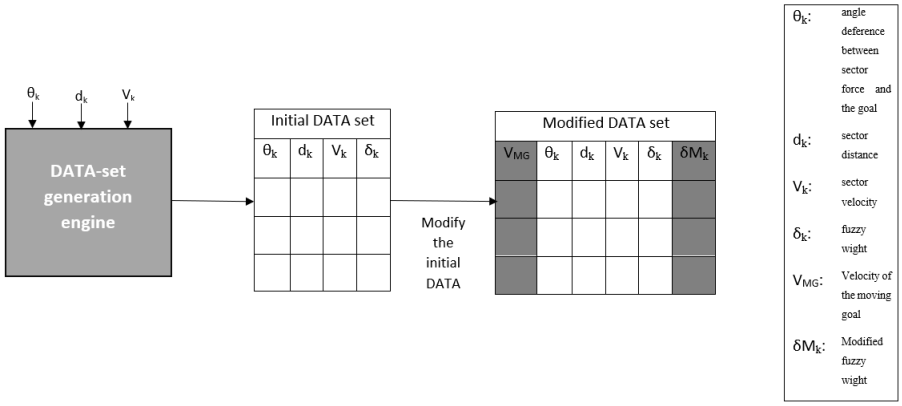


Fig. 8.7 Dataset Generation Procedure.

8.4.1.1 Datasets for Different Agoraphilic Behaviours

Five dataset generation engines were developed using five different FLCs to generate datasets. Those are also linguistic FLCs like the FLC discussed in Chapter 6. Four

databases were designed to develop the fuzzification module. Three databases were used for the three inputs:

1. angle difference between sector forces and the goal (θ_k))—five different databases were used for the five engines
2. normalised sector distance (d_k)—this is common for all five engines (databases are developed according to Eq. 8.7)
3. normalised sector velocity (v_k)—this is common for all five engines (databases are developed according to Eq. 8.8)

One database was used for the output:

1. the fuzzy shaping factor (δ_k) (see Fig. 8.9).

Similar to the database suggested to θ_k in Chapter 6, the database for θ_k also comprises nine fuzzy membership functions. The corresponding linguistics are LRR = left rear rear, LR = left rear, SL = side left, LF = left front, F = front, RF = right front, SR = side right, RR = right rear and RRR = right rear rear. As discussed in Chapter 7, five different databases were used for θ_k in five dataset generation engines:

1. for goal seeking Eq. 7.1
2. for normal travel Eq. 7.3
3. for safe travel Eq. 7.2
4. for right side safe Eq.7.4
5. for left side safe Eq. 7.5

Similar to the fuzzification module used in Chapter 6, the database for $d_{k,n}$ also comprises five fuzzy membership functions, as shown in Eq. 8.7. The corresponding linguistics are VC = very close, C = close, N = near, F = far, VF = very far (see Fig. 8.8).

$$\begin{aligned}
\mu_{VC}(d_{k,n}) &= max\{min\left(\frac{-d_{k,n}-0.3}{0.2}, 1\right), 0\} \\
\mu_{C}(d_{k,n}) &= max\{min\left(\frac{d_{k,n}-0.1}{0.2}, \frac{0.5-d_{k,n}}{0.2}\right), 0\} \\
\mu_{N}(d_{k,n}) &= max\{min\left(\frac{d_{k,n}-0.3}{0.2}, \frac{0.7-d_{k,n}}{0.2}\right), 0\} \\
\mu_{F}(d_{k,n}) &= max\{min\left(\frac{d_{k,n}-0.5}{0.2}, \frac{0.9-d_{k,n}}{0.2}\right), 0\} \\
\mu_{VF}(d_{k,n}) &= max\{min\left(\frac{d_{k,n}-0.7}{0.2}, 1\right), 0\}
\end{aligned} \tag{8.7}$$

Similar to the fuzzification module discussed in Chapter 6, the database for $v_{k,n}$ also comprises seven fuzzy membership functions, as shown in Eq. 8.8. The variable $v_{k,n}$ is positive if the MO is approaching the robot. The corresponding linguistics are NF = negative fast, NM = negative medium, NS = negative slow, Z = zero, PS = positive slow, PM = positive medium and PF = positive fast (see Fig. 8.8)

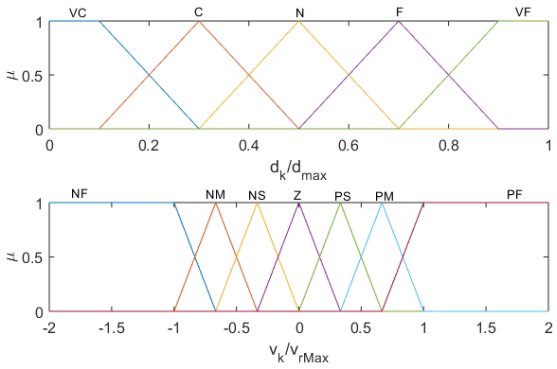


Fig. 8.8 Input Membership Functions of the Fuzzy Controller

$$
\begin{aligned}
\mu_{NF}(v_{k,n}) &= max\{min\left(\frac{-v_{k,n}-0.77}{0.33}, 1\right), 0\} \\
\mu_{NM}(v_{k,n}) &= max\{min\left(\frac{v_{k,n}+1}{0.33}, \frac{-0.33-v_{k,n}}{0.33}\right), 0\} \\
\mu_{NS}(v_{k,n}) &= max\{min\left(\frac{v_{k,n}+0.77}{0.33}, \frac{-v_{k,n}}{0.33}\right), 0\} \\
\mu_{Z}(v_{k,n}) &= max\{min\left(\frac{v_{k,n}+0.33}{0.33}, \frac{-0.33-v_{k,n}}{0.33}\right), 0\} \\
\mu_{PS}(v_{k,n}) &= max\{min\left(\frac{v_{k,n}}{0.33}, \frac{0.77-v_{k,n}}{0.33}\right), 0\} \\
\mu_{PM}(v_{k,n}) &= max\{min\left(\frac{v_{k,n}-0.33}{0.33}, \frac{1-v_{k,n}}{0.33}\right), 0\} \\
\mu_{PF}(v_{k,n}) &= max\{min\left(\frac{v_{k,n}-1}{0.33}, 1\right), 0\}
\end{aligned}
\tag{8.8}
$$

The simple output database is represented by eight fuzzy sets, as shown in Fig. 8.9. The fuzzy rule bases used for these data set generation engines consists of 189 rules.

In the initial dataset generation process (see Fig. 8.7), different combinations of input data (θ_k, d_k and v_k) are fed into the five dataset generation engines, and corresponding fuzzy shaping factors (δ_k) were recorded. This information is tabulated to produce five different initial datasets for different Agoraphilic behaviours.

However, in these initial datasets, there is no involvement of the velocity of the

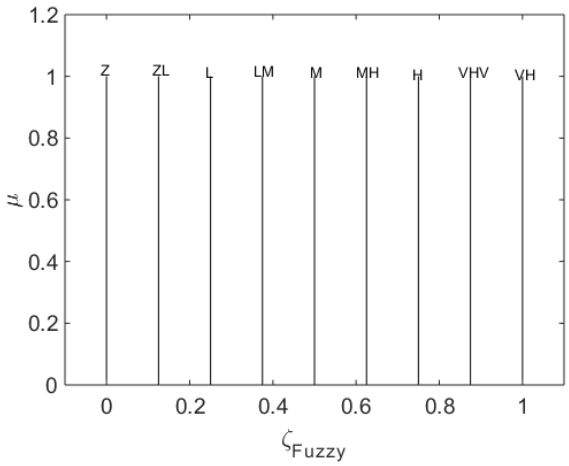


Fig. 8.9 Output Membership Functions of the Fuzzy Controller

moving goal. Therefore, these initial datasets were manually modified according to the domain knowledge to integrate the velocity of the moving goal into the dataset. Of the data from the new modified datasets, 60% was used to train the five new ML-based modules (see Fig. 8.7). These modules mimic the five different Agoraphilic behaviours:

1. ML module for goal seeking behaviour
2. ML module for goal normal travel behaviour
3. ML module for goal safe travel behaviour
4. ML module for goal right side safe behaviour
5. ML module for goal left side safe behaviour

The k-nearest [122–124] neighbours algorithm was used to build the ML-based modules.

Of the data from each modified dataset, 40% was used to evaluate the corresponding trained module. In the evaluation, it was identified that the mismatch of each module was as follows:

1. ML module for goal seeking behaviour, 23%

2. ML module for normal travel behaviour, 7%

3. ML module for safe travel behaviour, 8%

4. ML module for right side safe behaviour, 9%

5. ML module for left side safe behaviour, 10%.

8.5 Results and Discussion

8.5.1 Simulation Results

8.5.1.1 Experiment 1: The Robot Tries to Chase a Moving Goal in the Open Space

This experiment was conducted to test the algorithm's capability of following and reaching a moving goal in an open space. Start locations of the moving goal and the robot were (5, 20) m and (–20, –10) m, respectively. The goal kept moving towards the positive x-direction with a speed of 20 cm/s (see Fig. 8.10).

The observations of this experiment are as follows:

1. At time instant T1, the goal was 30m away from the robot. The robot started moving directly towards the goal.

2. The algorithm identified that the goal was moving towards the positive x-direction. Consequently, the robot increased its velocity component in the positive x-direction.

3. During the instants labelled T1 to T2, the robot could successfully chase the goal while reducing the distance between the goal and the robot.

4. At time instant T2, the distance between the robot and the goal was recorded as 24 m. (The robot had reduced the distance to the goal by 20%.)

Tab. 8.1 Experimental Conditions of Experiment 2.

Parameter	Value
Robot's path length	41.28m
Goal's path length	24m
Robot's avg. speed	34.4cm/s
Avg. speed of the goal	20cm/s
Total time	120s

5. At time instant T3, the robot changed its direction to the positive x-direction and reached the goal at (0, 10) m.

6. During this test, the robot kept increasing its velocity towards the goal until it hunted the target.

The robot took 120 s to finish the task successfully with a path length of 41.28 m. During the process, the algorithm pulled the robot towards the goal with the highest force magnitude. This allowed the robot to maintain its highest velocity. The summarised test results are shown in Table 8.1.

8.5.1.2 Experiment 2: The Robot Tries to Reach a Moving Goal in the Presence of a Static Obstacle

This experiment was conducted to test the algorithm's capability of reaching and following a moving goal in an open space. Start locations of the moving goal and the robot were (20, 20) m and (–20, –20) m, respectively. The goal kept moving towards the positive x-direction with a speed of 20 cm/s (see Fig. 8.11).

This experiment was conducted to test the algorithm's capability of reaching a moving goal under a special condition (local minima problem for the APF method). In this experiment, the robot, the goal and an SO were placed on a straight line. The

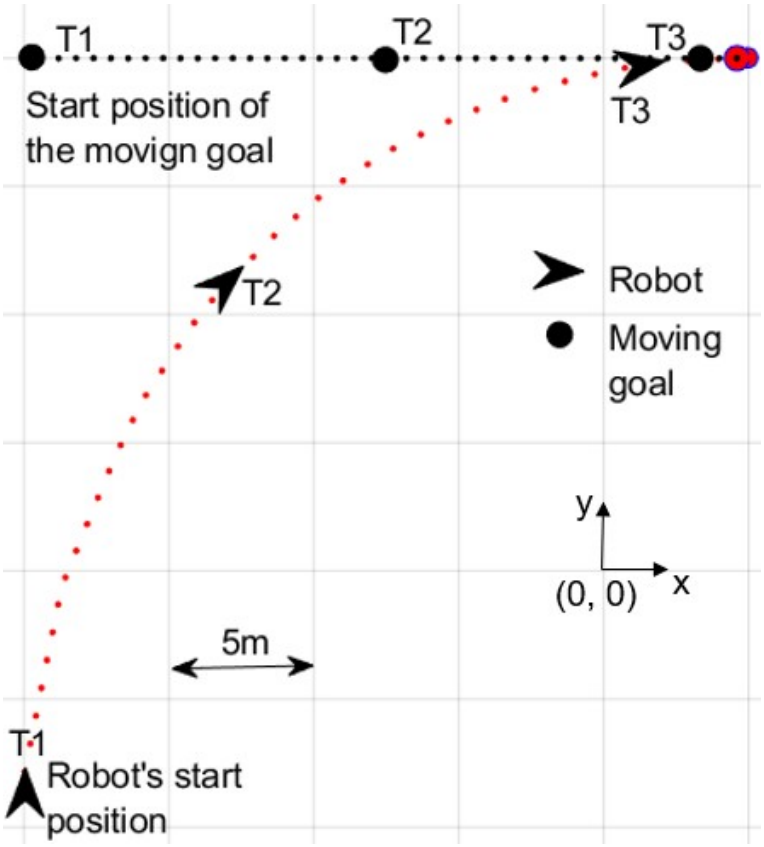


Fig. 8.10 The Robot's Path Observed in Simulation Experiment 1
Note. Locations of the robot and the goal in three different time instants are shown from T1 to T3.

goal was continuously moved directly towards the obstacle with a velocity of 14.14 $\angle(-135^0)$cm/s. This set-up creates a local minima problem for general APF-based algorithms [125]. Start locations of the moving goal and the robot were (20, 20) m and (–20, –20) m, respectively. The SO was placed at (0, 0) m (see Fig. Fig. 8.11). At the time instant T1, the goal was 56.56 m away from the robot.

The observations of this experiment are as follows (see Fig. 8.11):

1. The robot initially started moving directly towards the goal.

2. Once the robot started getting close to the SO, it started to deviate slightly from its original path to safely avoid the obstacle (time instant T2; see Fig. 8.11).

3. After successfully avoiding the obstacle, the robot changed its path towards the

Tab. 8.2 Test Results Summary of Simulation Experiment 2

Parameter	Value
Robot's path length	36.11m
Goal's path length	15.97m
Robot's avg. speed	32.8cm/s
Avg. speed of the goal	14.1cm/s
Total time	111s

moving goal.

4. The robot reached the goal at time instant T3.

During the whole process, the robot kept reducing its distance from the goal. The robot took 111 s to finish the task successfully with a path length of 36.11 m. During the process, the robot maintained an average speed of 32.8 cm/s. The robot reached the moving goal in the most efficient way. The summarised test results are shown in Table 8.2.

8.5.1.3 Experiment 3: The Robot Tries to Reach a Moving Goal in the Presence of a Moving Obstacle Challenging the Robot Continuously

This experiment was conducted to test the algorithm's capability of reaching a moving goal under a special condition with a moving goal and an MO (local minima problem for the APF method). In this experiment, the robot, the goal and an MO were placed on a straight line. The goal was continuously moved directly towards the obstacle with a velocity of $14.14\angle(-135^0)$ cm/s. The MO was challenging the robot by moving towards the robot at a velocity of $7.07\angle(-135^0)$ cm/s. This set-up creates a local minima problem for general APF-based algorithms [125]. Start locations of the moving goal, the robot and the MO were (20, 20) m, (–20, –20) m and (0, 0), respectively (see Fig.

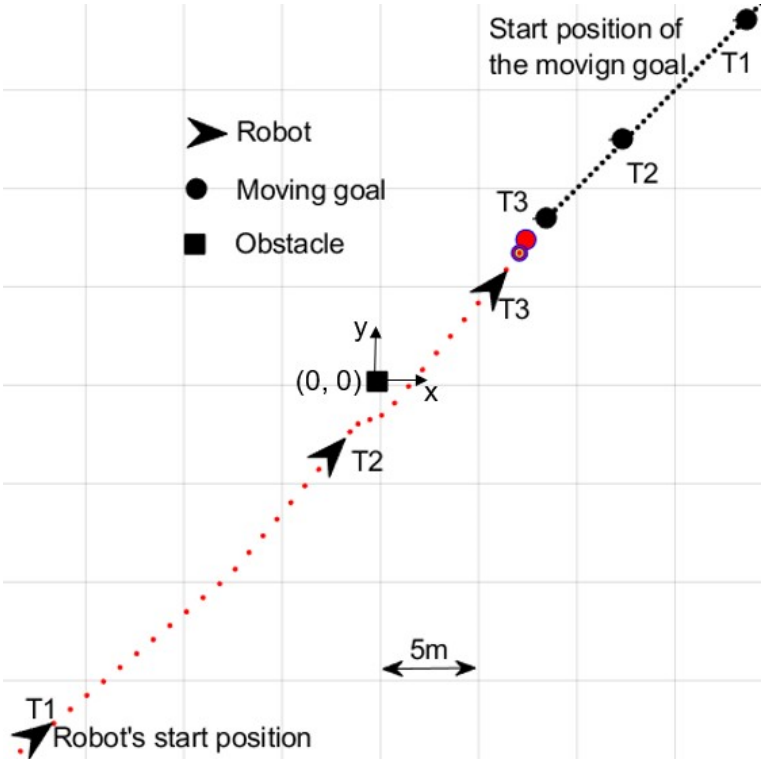


Fig. 8.11 The Robot's Path Observed in Simulation Experiment 2
Note. Locations of the robot and the goal in three different time instants are shown from T1 to T3.

8.12). At time instant T1, the goal was 56.56 m away from the robot.

The observations of this experiment are as follows (see 8.12):

1. The robot initially started moving towards the (+x, +y) direction (towards the goal).

2. At time instant T2, the robot started deviating from its original path as it identified the challenge from the MO.

3. The robot safely avoided the MO at time instant T3.

4. The robot changed its moving direction directly towards the goal after passing the MO and reached the goal at time instant T4.

During the whole process, the robot kept reducing its distance from the goal. The robot took 108 s to finish the task successfully with a path length of 36.19 m. During

Tab. 8.3 Test Results Summary of Simulation Experiment 3

Parameter	Value
Robot's path length	36.19 m
Goal's path length	15.47 m
Robot's avg. speed	33.5 cm/s
Avg. speed of the goal	14.3 cm/s
Total time	108 s

the process, the robot maintained an average speed of 33.5 cm/s. The summarised test results are shown in Table 8.3.

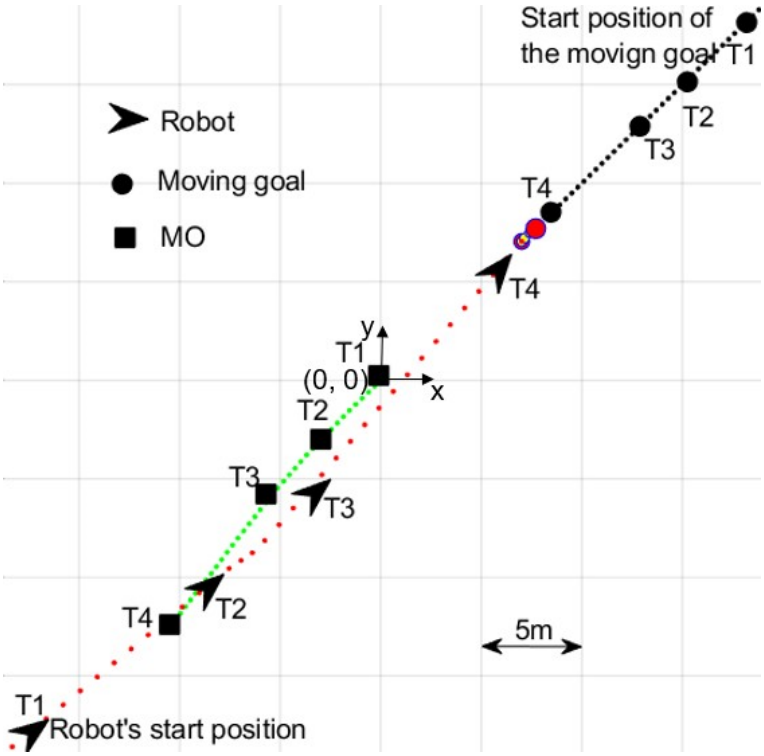


Fig. 8.12 The Robot's Path Observed in Simulation Experiment 3
Note. Locations of the robot, the goal and the moving obstacle (MO) in four different time instants are shown from T1 to T4.

8.5.1.4 Experiment 4: The Robot Tries to Reach a Moving Goal in the Presence of a Moving Obstacle Challenging the Robot by Crossing the Robot's Path

This experiment was conducted to test the algorithm's capability of reaching a moving goal with an MO crossing the robot's path in the middle. In this experiment, the robot,

the goal and the MO were placed at (–2.5, –6) m, (6.85, 20) m and (6, 5) m, respectively (see Fig. 8.13). The goal was continuously moved towards the negative x-direction with a speed of 5 cm/s. The MO challenged the robot by crossing the robot's path at a velocity of $10\angle(180^0)$cm/s.

The observations of this experiment are as follows (see Fig. 8.13):

1. At time instant T1, the goal was located directly in the (+x, +y) direction of the robot with a distance of 45.8 m.

2. The robot started moving directly towards the goal at the beginning.

3. The path prediction module identified that the MO would cross the robot's path at time instant T3.

4. As a result of Observation 3, at time instant T2

   (a) the robot started deviating from its original path

   (b) the algorithm reduced the magnitude of the robot's driving force by 56% (see Fig. 8.14).

5. At time instant T3, the robot could safely avoid the obstacle.

6. After safely passing the obstacle, the robot changed its path directly towards the moving goal (as in Experiments 1–3) and reached the goal at time instant T4.

7. In this experiment, the algorithm showed a human-like behaviour by reducing the robot's speed and allowing the MO to pass the robot.

During the whole process, the robot kept reducing its distance from the goal. The robot took 84 s to finish the task successfully with a path length of 24.75 m. During

Tab. 8.4 Test Results Summary of Simulation Experiment 4

Parameter	Value
Robot's path length	24.75m
Goal's path length	4.05m
Robot's avg. speed	14.3cm/s
Avg. speed of the goal	4.9cm/s
Total time	84s

the process, the robot maintained an average speed of 14.3 cm/s. The summarised test results are shown in Table 8.4.

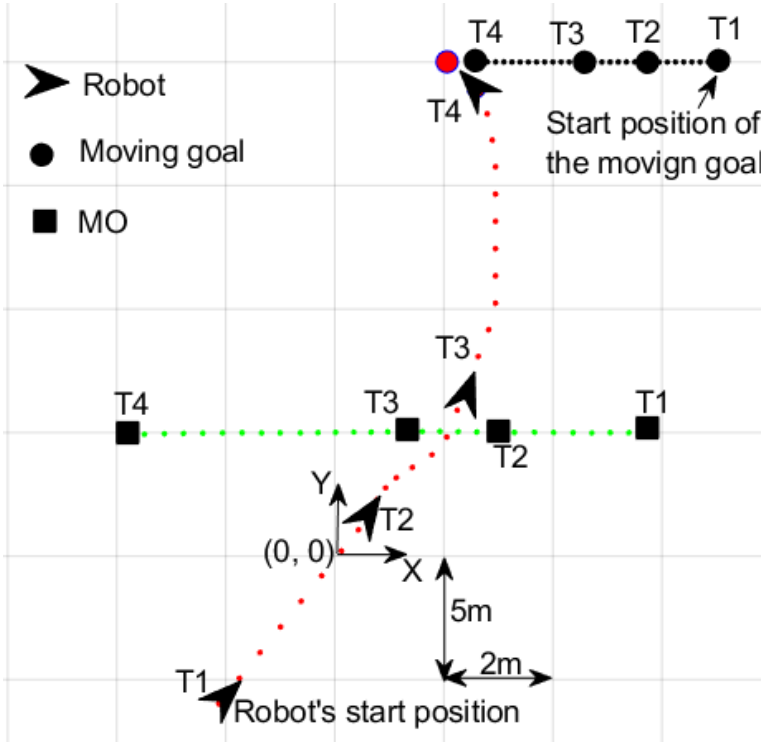


Fig. 8.13 TThe Robot's Path Observed in Simulation Experiment 4
Note. Locations of the robot, the goal and the moving obstacle (MO) in four different time instants are shown from T1 to T4.

8.5.1.5 Experiment 5: The Robot Tries to Reach a Moving Goal in a Complex Dynamic Environment in the Presence of Multiple Moving Obstacles

This experiment was conducted to test the algorithm's capability of reaching a moving goal in a dynamically cluttered environment with multiple MOs. In this experiment,

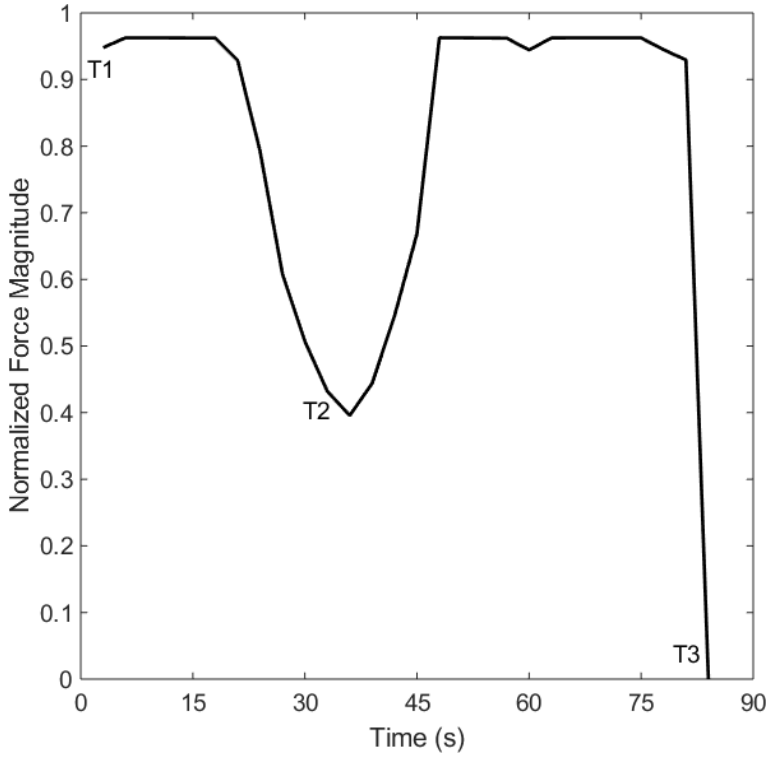


Fig. 8.14 The Robot's Normalised Driving Force Magnitude Variation over Experiment 4

the goal and some of the MOs rapidly changed their velocities. This created a rapidly changing environment with high uncertainties for the robot. The starting conditions of the experiment are shown in Table 8.5.

The observations of this experiment are as follows (see Fig. 8.15):

1. At time instant T1, the goal was located directly in the (+x, +y) direction with respect to the robot with a distance of 41.23m.

Tab. 8.5 Initial Conditions of Experiment 5

Object	Starting Location(m)	Starting Velocity(cm/s)
Robot	(-20, -20)	(0, 0)
Goal	(20, -10)	$22.360\angle(153.43^0)$
Moving Obstacle 1	(-5, -20)	$10\angle(90^0)$
Moving Obstacle 2	(-20, -5)	$25\angle(0^0)$
Moving Obstacle 3	(20, -10)	$22.36\angle(153.43^0)$

2. The robot moved directly towards the goal at the beginning.

3. The algorithm identified that MO1 would cross the robot's path at (-43.7, -14.3) m.

4. As a result of Observation 3, the robot reduced its driving force by 66% at time instant T2 and allowed MO1 to pass the robot (see Fig. 8.16).

5. The robot was again challenged by MO2 and MO3 at time instants T3 and T4, respectively.

   (a) At time instant T3, the algorithm reduced the robot's driving force by 30% and allowed MO2 to pass the robot.

   (b) Immediately after passing MO2, the algorithm increased the robot's driving force by 30%, sped up the robot during T3 to T4 and passed the MO3 safely before it crossed the robot's path.

6. After passing MO3, the robot moved directly towards the moving goal and hunted it at time instant T5.

During the whole process, the robot kept reducing its distance from the goal. The robot took 129 s to finish the task successfully with a path length of 36.04 m. During the process, the robot maintained an average speed of 27.9 cm/s. The summarised test results are shown in Table 8.6.

8.5.2 Experiment Test Results

This experiment was conducted to test the algorithm's capability of reaching a moving goal under a dynamically cluttered environment with multiple MOs in real-world circumstances. The TB3 Waffle Pi research robot platform discussed in Chapter 4 was

Tab. 8.6 Test Results Summary of Simulation Experiment 5

Parameter	Value
Robot's path length	36.04m
Goal's path length	17.44m
Robot's avg. speed	27.9cm/s
Avg. speed of the goal	13.52cm/s
Total time	129s

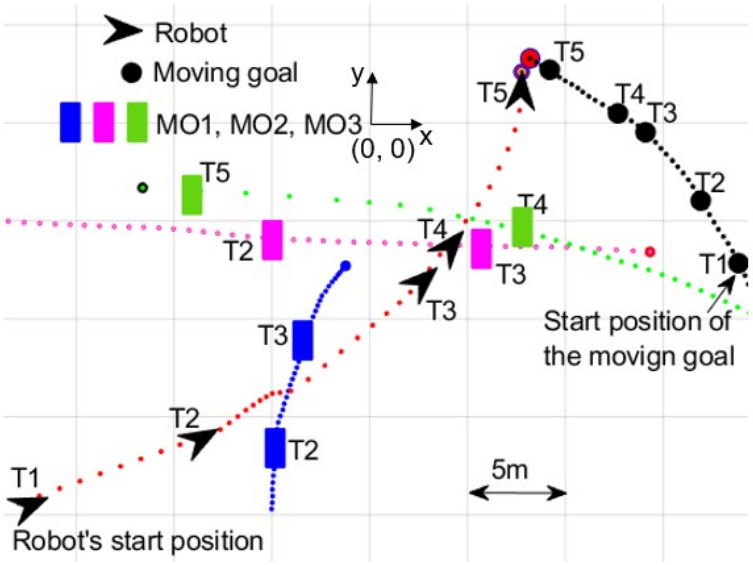


Fig. 8.15 The Robot's Path Observed in Simulation Experiment 5
Note. Locations of the robot, the goal and the moving obstacles (MOs) in four different time instants are shown from T1 to T4.

used in this experiment. In this experiment, two MOs were used with some SOs. The moving goal performed a major direction change in this experiment. The starting conditions of the experiment are shown in Table 8.7.

The goal, the robot and MOs started moving at time instant T1 (see Fig. 8.17). At this time instant, the goal was located directly in the (+x, –y) direction of the robot with a distance of 447.8 cm. During T1 to T2, MO1 and MO2 developed their movements on the robot's left-hand side with the intention of challenging the robot in the future iterations. At time instant T2, the robot had to change its direction because of the SO in front of it. In this situation, the robot had two options:

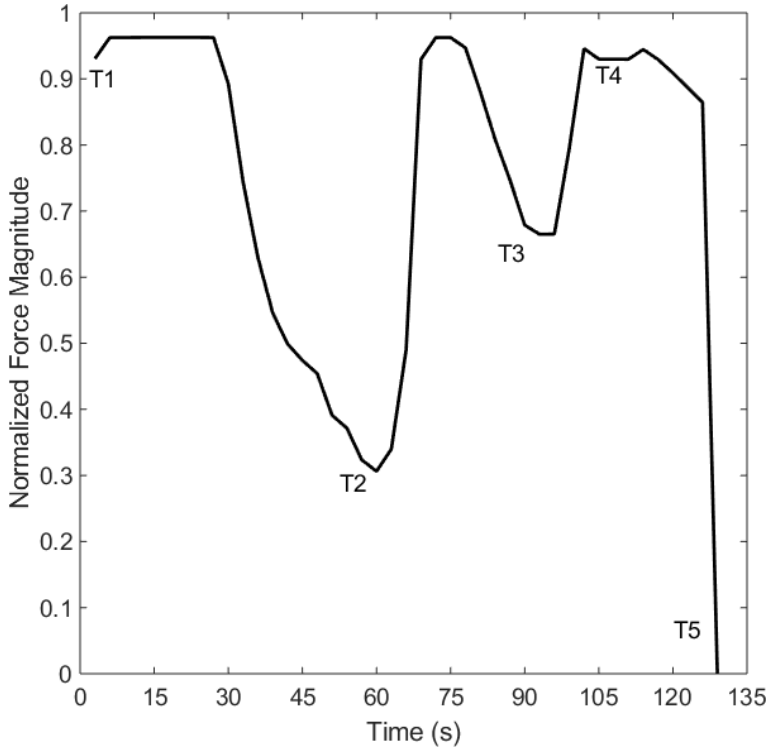


Fig. 8.16 The Robot's Normalised Driving Force Magnitude Variation over Experiment 5

Tab. 8.7 Initial Conditions of the Experiment

Object	Starting Location(cm)	Starting Velocity(cm/s)
Robot	(0, -0)	0
Goal	(306, -327)	$3.20\angle(-141.34^0)$
Moving Obstacle 1	(150, 100)	$2.5\angle(-71.57^0)$
Moving Obstacle 2	(350, 0)	$0.81\angle(178.72^0)$

1. move to the left (towards MO1 and MO2)

2. move to the right (towards more free space)

The robot picked the first option, although there was more challenge from the MOs. The new moving direction of the goal was the main reason for this decision. At time instant T2, the goal had changed its velocity towards the positive y-direction. During time instants T2 to T3, the robot moved with a low speed and allowed MO1 to pass the robot. During time instants T3 to T4, the robot moved towards the goal through

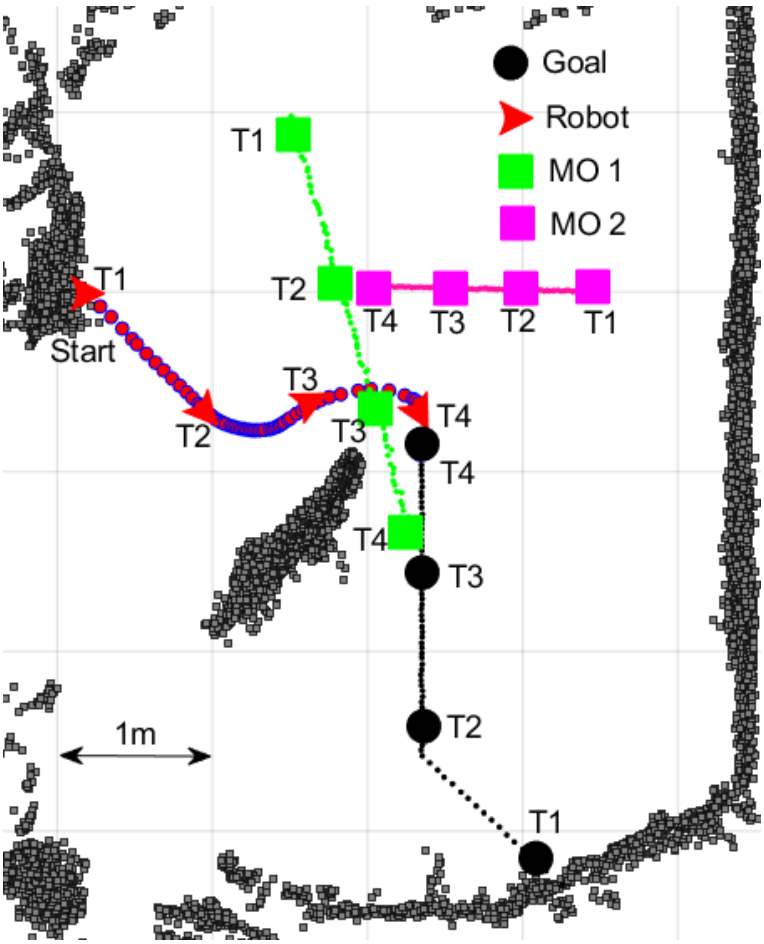


Fig. 8.17 The Robot's Path Observed in Experiment
Note. Locations of the robot, the goal and the moving obstacles (MOs) in four different time instants are shown from T1 to T4.

the free space passage between MO1, MO2 and the SO. At time instant T4, the robot successfully reached the moving goal.

The robot took 89 s to finish the task successfully with a path length of 275 cm. During the process, the robot maintained an average speed of 3 cm/s. The summarised test results are shown in Table 8.8.

8.5.2.1 Comparative Test between the Fuzzy Logic Controller–Based ANADE V and the Machine Learning–Based ANADE V

The experimental work presented in this section was conducted to demonstrate and validate the performance of the ANADE V with a novel ML-based force-shaping module.

Tab. 8.8 Test Results Summary of the Experimental Test

Parameter	Value
Robot's path length	275m
Goal's path length	301m
Robot's avg. speed	3cm/s
Avg. speed of the goal	3.4cm/s
Total time	89s

Tab. 8.9 Initial Conditions of Comparison Test 1

Object	Starting Location(cm)	Starting Velocity(cm/s)
Robot	(0,0)	0
Goal	(900, 0)	$2\angle(90^o)$
Moving obstacle	(150, 100)	$2\angle(180^o)$

The experiments were designed to test the performance of the new ML-based ANADE V with respect to FLC-based ANADE V. From the range of investigations performed, one simulation test and an experimental test is presented in this section.

Comparison 1: Simulation Test

This simulation test was conducted to test the algorithm's capability of reaching a moving goal in a dynamic environment with an MO and SO. In this simulation, two tests were conducted:

1. with the ML based force-shaping module
2. with FLC based force-shaping module

In both tests, all other parameters were kept the same. The initial conditions of the tests are shown in Table 8.9.

The goal, the robot and the MO started moving at time instant T1 (see Figure 8.18). At this time instant, the goal was located directly in the (+x, –y) direction of the robot

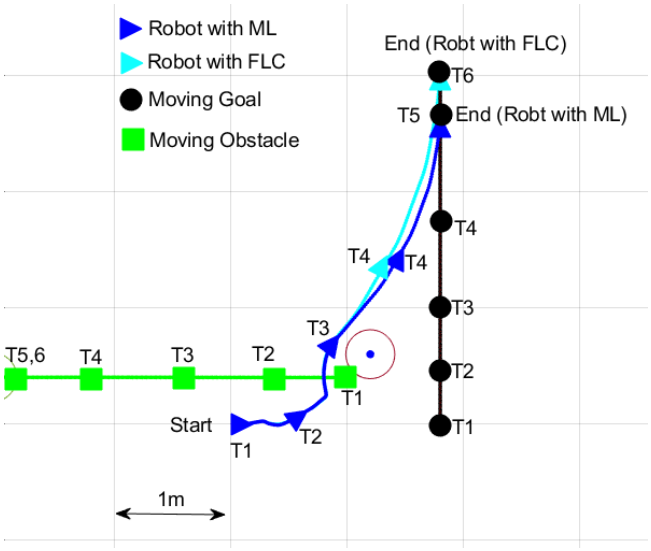


Fig. 8.18 The Robot's Paths Observed in the Simulation Test (Comparison 1)
Note. Locations of the robot, the goal and the moving obstacle in six different time instants are shown from T1 to T6. ML = machine learning; FLC = Fuzzy Logic Controller.

with a distance of 900 cm. During T1 to T2, MO1 developed its movements on the robot's right-hand side with the intention of challenging the robot. At time instant T2, both the robot with ML and the one with FLC successfully passed MO1. From T1 to T3, both robots followed exactly the same path. The speed variations were also almost the same during T1 to T3 (see Figure 8.19). However, the robot with ML started increasing its speed as soon as it passed the SO (at T3; see Figure 8.19). Also, it started moving towards the moving direction of the goal. Conversely, the robot with FLC started increasing its speed 50 iterations later. As a result, the robot with ML reached the goal using a path 10% shorter and in a time 13.78% quicker than those of the robot with FLC. Further, the robot with ML had a 4.3% higher average speed than did the robot with FLC. The summarised test results are shown in Table 8.10.

8.5.2.2 Comparison 2: Experimental Test

This experimental test was conducted to test the algorithm's capability of reaching a moving goal in a dynamic environment with multiple MOs and SOs in a real-world

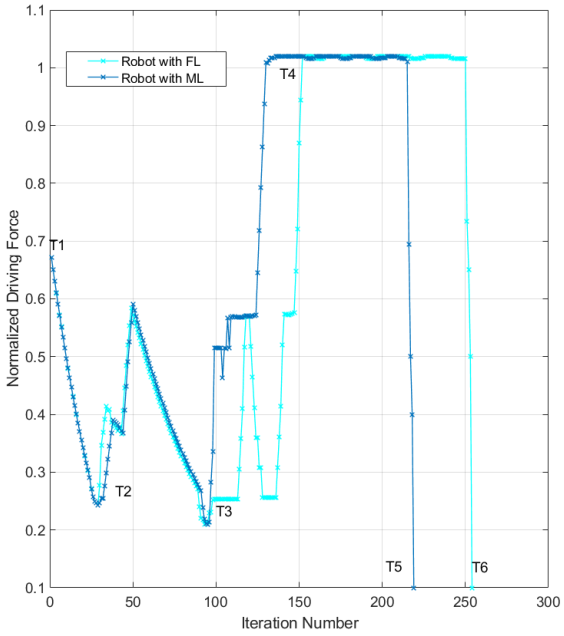


Fig. 8.19 The Robot's Normalised Driving Force Variations Observed in the Simulation Test (Comparison 1)

Tab. 8.10 Summarised Test Results of Comparison Test 1

Parameter	With ML	With FLC	20cm Performance comparison with respect to case 2
Robot's path length	1749cm	1943cm	10%
Goal's path length	1303cm	1512cm	13.78%
Robot's avg. speed	2.68cm/s	2.57cm/s	4.3%
Speed of the goal	2cm/s	2cm/s	N/A
Total time	651.8s	756s	13.78%

setting. Similar to Comparison 1, two tests were also conducted in this experiment:

1. with the ML-based force-shaping module

2. with FLC-based force-shaping module

In both tests, all other parameters were kept the same. The initial conditions of the tests are shown in Table 8.11.

The goal, the robot and the MO started moving at time instant T1 (see Fig. 8.20).

Tab. 8.11 Initial Conditions of Comparison Test 2 (Experimental Test)

Object	Starting Location(cm)	Starting Velocity(cm/s)
Robot	(0,0)	$0\angle(0^o)$
Goal	(0, -250)	$2\angle(0^o)$
Moving obstacle 1	(150, -100)	$1.50\angle(177^o)$
Moving obstacle 2	(150, -100)	$1.58\angle(179^o)$

At this time instant, the goal was located directly in the negative y-direction of the robot at a distance of 250 cm. During T1 to T2, MO1 developed its movements with the intention of challenging the robot. At time instant T2, both the robot with ML and the one with FLC successfully passed MO1. From T1 to T3, both robots followed slightly different paths. The speed variations show that both robots reduced speed at the challenge from MO2 (see Fig. 8.21). The robot with FLC started changing its direction to avoid MO2. Conversely, the robot with ML, without changing its path, allowed MO2 to pass the robot. The robot with ML increased its speed as soon as MO2 passed the robot (at T3; see Fig. 8.21). Also, it started moving towards the moving direction of the goal. Conversely, the robot with FLC started increasing its speed 35 iterations later. As a result, the robot with ML reached the goal with a path 19% shorter and in a time 35% quicker than did the robot with FLC. However, the robot with FLC had an 8% higher average speed than did the robot with FLC. The summarised test results are shown in Table 8.12.

Fig. 8.22 shows a video of two experiments that ANADE V navigates the TB3 waffle-pi robot in a dynamic environment. In the first experiment goal is static and in the second experiment goal in dynamic.

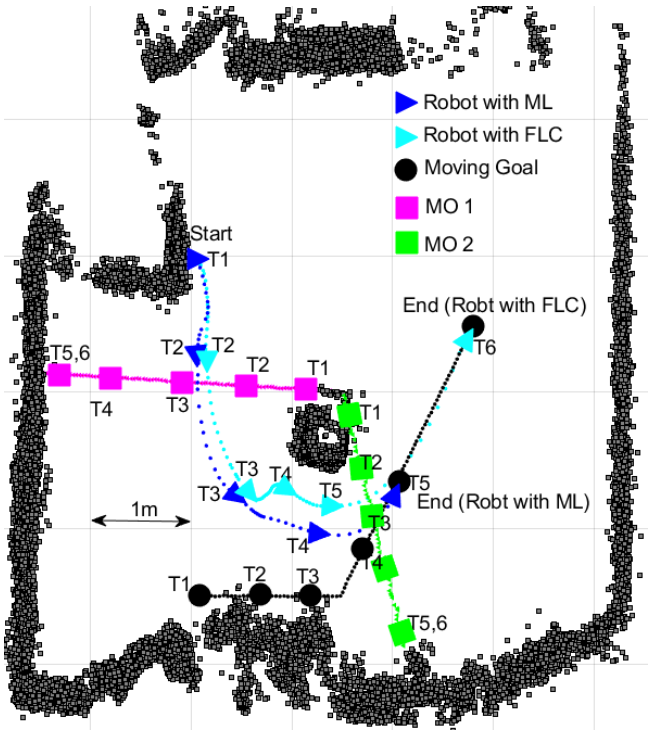


Fig. 8.20 The Robot's Paths Observed in the Experimental Test (Comparison 2)
Note. Locations of the robot, the goal and the moving obstacles (MOs) in six different time instants are shown from T1 to T6. ML = machine learning; FLC = Fuzzy Logic Controller.

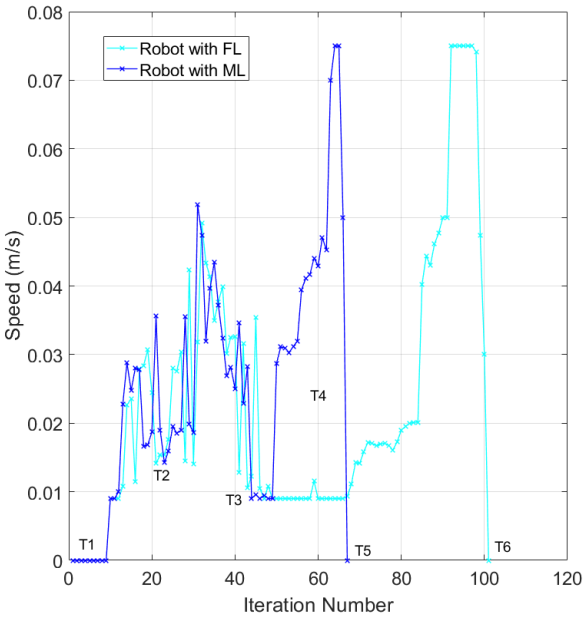


Fig. 8.21 The Robot's Speed Variations Observed in the Experimental Test (Comparison 2)

Tab. 8.12 Summarised test results of the comparison test 2 (Experimental test)

Parameter	Without ML	With FLC	Performance comparison with respect to case 2
Robot's path length	376cm	467cm	19%
Goal's path length	252cm	379cm	33%
Robot's avg. speed	2.1cm/s	2.3cm/s	8%(low)
Avg. Speed of the goal	1.54cm/s	1.52cm/s	N/A
Total time	132s	202s	35%
Avg. sample time	1.9s	2.0s	5%

Fig. 8.22 A Video of ANADE V Navigating a Robot in a Dynamic Environment (GIF video playback speed $\sim 9\times$)
`https://drive.google.com/file/d/1zRa8PnyMAdGmL9Ulb4HfWNmhhWNEq0Vf/view?usp=sharing`

8.6 Summary

The ANADE V is successful in navigating robots under unknown, static and dynamic environments with a moving goal. The algorithm uses a single attractive force to pull the robot towards the moving goal via current free space leading to future free space passages. The presented goal tracking module, goal path prediction modules, new current global map generation module and improved future global map generation module combined with other main modules discussed in Chapters 3–7 allow the algorithm to

incorporate the rapid changes that occur in unknown dynamic environments with high uncertainties.

In this chapter, five simulation experiments and one real-world experiment were presented to prove the effectiveness of the novel algorithm in navigating a robot with a moving goal. The conducted first experiment showed that the robot could reach a moving goal in an open environment without any issue. The second and third experiments showed that the algorithm has no problems with common local minima issues with the presence of static or dynamic obstacles with a moving goal. A moving obstacle challenged the robot by crossing the robot's path while it was chasing the moving goal in the fourth experiment. In this experiment, the algorithm made human-like decisions to reach the moving goal safely. The final simulation experiment showed that the algorithm could successfully reach a moving target even in a dynamically cluttered environment with multiple moving obstacles. The real-world experiment validates the algorithm's performance in real-world applications.

Further, two comparison experiments were conducted to show the importance of the machine learning technique. The comparison experiments proved that the novel machine learning-based method improved the algorithm's performance by reducing path length, the time to hunt the moving goal and the sample time. Comparison Experiment 1 showed that the algorithm with machine learning reached the goal with a path 10% shorter and in a time 13.8% quicker than did the robot with fuzzy logic controller. Further, in this experiment, the robot with machine learning had a 4.3% higher average speed than did the robot with fuzzy logic controller. The second comparison experiment showed that the algorithm with machine learning reached the goal with a path 19% shorter and in a time 35% quicker than did the robot with fuzzy logic controller.

Also, the algorithm with ML had a 5% lesser average sampling time than did the robot with fuzzy logic controller. These comparative and experimental results demonstrate the improvements of the algorithm after introducing the machine learning technique. The presented improved ANADE demonstrated the enhanced capability to successfully navigate mobile robots in dynamically cluttered environments to hunt a moving target.

9

Chapter 9: Conclusion

This research introduced a novel navigation algorithm based on the free space attraction concept. This new navigation algorithm is capable of navigating robots not only in static environments but also in unknown dynamic environments with moving targets. In this thesis, we showed the step-by-step development of the novel free space attraction-based navigation algorithm while addressing the identified general weaknesses of other algorithms.

Chapter 4 introduced the Agoraphilic Navigation Algorithm in Dynamic Environment I (ANADE I), which is a free space attraction-based navigation algorithm. It uses only attractive forces created by the current free space in the robot's surrounding environment to create the robot's driving force. The ANADE I uses an object tracking method to accurately estimate the locations of moving obstacles. The new algorithm drives the robot through the current free spaces leading to the goal. This proposed methodology was able to overcome the main limitation of the original Agoraphilic algorithm—which was limited to static environments—while keeping the original algorithm's advantages in static environments. The simulation test results demonstrated the success of the ANADE I.

The ANADE II is also a local path planner to navigate mobile robots in unknown

static and dynamic environments. It does not look for obstacles to avoid but for space solutions to follow. Compared to the ANADE I, the ANADE II uses a dynamic obstacle path prediction methodology to identify the robot's future environments. The ANADE II pulls the robot through the current free space towards the future growing free space passages leading to the goal. The ANADE I only considers the current free space. This new methodology has successfully improved the ANADE I, which has some limitations under cluttered dynamic environments. The free space attraction approach of the new algorithm combined with future dynamic obstacle path prediction makes the ANADE II superior to the ANADE I. The advantages of the ANADE II were proven by a comparative study. Further, the experimental and simulation results demonstrated the success of the ANADE II in dynamically cluttered environments.

Chapter 6 introduced an improved version of the ANADE II: the ANADE III. In the ANADE III, a novel controller based on fuzzy logic was developed and combined with the new free space attraction concept to provide optimal navigational solutions at a low computational cost. The effectiveness of the ANADE II was improved further by incorporating the velocity vectors of moving obstacles into decision-making.

In the ANADE IV, a self-tuning system was successfully applied to the ANADE III to further improve the algorithm's decision-making capability under high uncertainties. The new self-tuning fuzzy logic controller maximised the decision-making flexibility of the algorithm. A new supervisory controller was integrated into the new self-tuning fuzzy logic controller to tune the parameters of the main fuzzy logic controller. Those parameters are tuned according to the robot's environment. The enhancement of the new self-tuning fuzzy logic controller was verified by the experimental tests.

ANADE IV was further improved in Chapter 8 to make it capable of hunting a mov-

ing target in an unknown, dynamically cluttered environment. This resulted in ANADE V. This algorithm has all the advantages of the ANADE IV combined with the new machine learning-based moving goal hunting feature. According to the literature review, there is only a handful of navigation algorithms capable of hunting a moving goal or target in an unknown, dynamic environment. Further, the simulation and experimental test results in Chapter 8 validated the algorithm.

In future research, the new machine-learning-based force-shaping module discussed in Chapter 8 can be further improved by using different machine-learning techniques. These improvements will potentially further enhance the algorithm's human like decision making ability, overall efficiency and also the computational cost. Furthermore, in future research, the ANDE V algorithm can be advanced for navigating robots in a 3-dimensional space.

This thesis presents a new approach to the virtual force–based mobile robot navigation methodologies. As highlighted earlier, the developed novel free space attraction-based navigation algorithm has several primary contributions to mobile robot navigation in unknown dynamic environments. The concept presented in this thesis provides the foundation for a new generation of local path planning algorithms that can be used in dynamic environment.

REFERENCES

[1] W. Liu, Z. Li, S. Sun, M. K. Gupta, H. Du, R. Malekian, M. A. Sotelo, and W. Li, "Design a novel target to improve positioning accuracy of autonomous vehicular navigation system in gps denied environments," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2021.

[2] Y. Gao and S. Chien, "Review on space robotics: Toward top-level science through space exploration," *Science Robotics*, vol. 2, no. 7, 2017.

[3] I. Rangapur, B. S. Prasad, and R. Suresh, "Design and development of spherical spy robot for surveillance operation," *Procedia Computer Science*, vol. 171, pp. 1212 – 1220, 2020, third International Conference on Computing and Network Communications (CoCoNet'19).

[4] D. Patil, M. Ansari, D. Tendulkar, R. Bhatlekar, V. N. Pawar, and S. Aswale, "A survey on autonomous military service robot," in *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, 2020, pp. 1–7.

[5] X. Huang, Q. Cao, and X. Zhu, "Mixed path planning for multi-robots in structured hospital environment," *The Journal of Engineering*, vol. 2019, no. 14, pp. 512–516, 2019.

[6] E. Kayacan, E. Kayacan, H. Ramon, O. Kaynak, and W. Saeys, "Towards agrobots: Trajectory control of an autonomous tractor using type-2 fuzzy logic controllers," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 1, pp. 287–298, 2015.

[7] G. Sawadwuthikul, T. Tothong, T. Lodkaew, P. Soisudarat, S. Nutanong, P. Manoonpong, and N. Dilokthanakul, "Visual goal human-robot communication framework with few-shot learning: a case study in robot waiter system," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2021.

[8] A. Toyama, K. Mitsugi, K. Matsuo, and L. Barolli, "Optimal number of moap robots for wmns using silhouette theory," in *Advances in Intelligent Networking*

and Collaborative Systems, L. Barolli, K. F. Li, and H. Miwa, Eds. Cham: Springer International Publishing, 2021, pp. 281–290.

[9] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "Heuristic approaches in robot path planning: A survey," *Robotics and Autonomous Systems*, vol. 86, pp. 13 – 28, 2016.

[10] B. B. K. Ayawli, X. Mei, M. Shen, A. Y. Appiah, and F. Kyeremeh, "Mobile robot path planning in dynamic environment using voronoi diagram and computation geometry technique," *IEEE Access*, vol. 7, pp. 86 026–86 040, 2019.

[11] B. Oommen, S. Iyengar, N. Rao, and R. Kashyap, "Robot navigation in unknown terrains using learned visibility graphs. part i: The disjoint convex obstacle case," *IEEE Journal on Robotics and Automation*, vol. 3, no. 6, pp. 672–681, 1987.

[12] F. Samaniego, J. Sanchis, S. García-Nieto, and R. Simarro, "Uav motion planning and obstacle avoidance based on adaptive 3d cell decomposition: Continuous space vs discrete space," in *2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM)*, 2017, pp. 1–6.

[13] N. Malone, H. T. Chiang, K. Lesser, M. Oishi, and L. Tapia, "Hybrid dynamic moving obstacle avoidance using a stochastic reachable set-based potential field," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1124–1138, 2017.

[14] D. H. Lee, S. S. Lee, C. K. Ahn, P. Shi, and C. Lim, "Finite distribution estimation-based dynamic window approach to reliable obstacle avoidance of mobile robot," *IEEE Transactions on Industrial Electronics*, pp. 1–1, 2020.

[15] J. . Kim and P. K. Khosla, "Real-time obstacle avoidance using harmonic potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 3, pp. 338–349, June 1992.

[16] P. Szulczyński, D. Pazderski, and K. Kozłowski, "Real-time obstacle avoidance using harmonic potential functions," *Journal of Automation Mobile Robotics and Intelligent Systems*, vol. Vol. 5, No. 3, pp. 59–66, 2011.

[17] C. I. Connolly, J. B. Burns, and R. Weiss, "Path planning using laplace's equation," in *Proceedings., IEEE International Conference on Robotics and Automation*, May 1990, pp. 2102–2106 vol.3.

[18] J. Guldner and V. I. Utkin, "Sliding mode control for gradient tracking and robot navigation using artificial potential fields," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 2, pp. 247–254, April 1995.

[19] R. Iraji and M. T. Manzuri-Shalmani, "A new fuzzy-based spatial model for robot navigation among dynamic obstacles," in *2007 IEEE International Conference on Control and Automation*, May 2007, pp. 1323–1328.

[20] Min Cheol Lee and Min Gyu Park, "Artificial potential field based path planning for mobile robots using a virtual obstacle concept," in *Proceedings 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2003)*, vol. 2, July 2003, pp. 735–740 vol.2.

[21] G. Li, A. Yamashita, H. Asama, and Y. Tamura, "An efficient improved artificial potential field based regression search method for robot path planning," in *2012 IEEE International Conference on Mechatronics and Automation*. IEEE, 2012, pp. 1227–1232.

[22] L. Tang, S. Dian, G. Gu, K. Zhou, S. Wang, and X. Feng, "A novel potential field method for obstacle avoidance and path planning of mobile robot," in *2010 3rd international conference on computer science and information technology*, vol. 9. IEEE, 2010, pp. 633–637.

[23] S. S. Ge and Y. J. Cui, "New potential functions for mobile robot path planning," *IEEE Transactions on robotics and automation*, vol. 16, no. 5, pp. 615–620, 2000.

[24] J. Ren, K. A. McIsaac, and R. V. Patel, "Modified newton's method applied to potential field-based navigation for mobile robots," *IEEE Transactions on Robotics*, vol. 22, no. 2, pp. 384–391, 2006.

[25] J. Ren, K. A. McIsaac, and R. Patel, "Modified newton's method applied to potential field-based navigation for nonholonomic robots in dynamic environments," *Robotica*, vol. 26, no. 1, pp. 117–127, 2008.

[26] J. Sfeir, M. Saad, and H. Saliah-Hassane, "An improved artificial potential field approach to real-time mobile robot path planning in an unknown environment," in *2011 IEEE International Symposium on Robotic and Sensors Environments (ROSE)*, Sep. 2011, pp. 208–213.

[27] O. Montiel, U. Orozco-Rosas, and R. Sepúlveda, "Path planning for mobile robots using bacterial potential field for avoiding static and dynamic obstacles," *Expert Systems with Applications*, vol. 42, no. 12, pp. 5177 – 5191, 2015.

[28] W. Siming, Z. Tiantian, and L. Weijie, "Mobile robot path planning based on improved artificial potential field method," in *2018 IEEE International Conference of Intelligent Robotic and Control Engineering (IRCE)*, Aug 2018, pp. 29–33.

[29] M. A. K. Jaradat, M. H. Garibeh, and E. A. Feilat, "Autonomous mobile robot dynamic motion planning using hybrid fuzzy potential field," *Soft Computing*, vol. 16, no. 1, pp. 153–164, Jan 2012. [Online]. Available: https://doi.org/10.1007/s00500-011-0742-z

[30] B. Kovács, G. Szayer, F. Tajti, M. Burdelis, and P. Korondi, "A novel potential field method for path planning of mobile robots by adapting animal motion attributes," *Robotics and Autonomous Systems*, vol. 82, pp. 24 – 34, 2016.

[31] R. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal on Robotics and Automation*, vol. 2, no. 1, pp. 14–23, March 1986.

[32] L. McFetridge and M. Ibrahim, "A new methodology of mobile robot navigation: The agoraphilic algorithm," *Robotics and Computer-Integrated Manufacturing*, vol. 25, no. 3, pp. 545 – 551, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0736584508000495

[33] M. Y. Ibrahim and L. McFetridge, "The agoraphilic algorithm: a new optimistic approach for mobile robot navigation," in *2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics. Proceedings (Cat. No.01TH8556)*, vol. 2, July 2001, pp. 1334–1339 vol.2.

[34] L. McFetridge and M. Y. Ibrahim, "Behavior fusion via free-space force shaping," in *IEEE International Conference on Industrial Technology, 2003*, vol. 2, Dec 2003, pp. 818–823 Vol.2.

[35] B. Patle, G. Babu L, A. Pandey, D. Parhi, and A. Jagadeesh, "A review: On path planning strategies for navigation of mobile robot," *Defence Technology*, vol. 15, no. 4, pp. 582 – 606, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2214914718305130

[36] F. Kamil, T. S. Hong, W. Khaksar, M. Y. Moghrabiah, N. Zulkifli, and S. A. Ahmad, "New robot navigation algorithm for arbitrary unknown dynamic environments based on future prediction and priority behavior," *Expert Systems with Applications*, vol. 86, pp. 274 – 291, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417417303809

[37] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.

[38] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "Heuristic approaches in robot path planning: A survey," *Robotics and Autonomous Systems*, vol. 86, pp. 13–28, 2016.

[39] W. Yaonan, Y. Yimin, Y. Xiaofang, Z. Yi, Z. Yuanli, Y. Feng, and T. Lei, "Autonomous mobile robot navigation system designed in dynamic environment based on transferable belief model," *Measurement*, vol. 44, no. 8, pp. 1389–1405, 2011.

[40] N. Y. Ko and B. H. Lee, "Avoidability measure in moving obstacle avoidance problem and its use for robot motion planning," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS'96*, vol. 3. IEEE, 1996, pp. 1296–1303.

[41] N. Leena and K. Saju, "A survey on path planning techniques for autonomous mobile robots," *IOSR Journal of Mechanical and Civil Engineering (IOSR-JMCE)*, vol. 8, pp. 76–79, 2014.

[42] T. S. Hong, D. Nakhaeinia, and B. Karasfi, "Application of fuzzy logic in mobile robot navigation," *Fuzzy Logic-Controls, Concepts, Theories and Applications*, pp. 21–36, 2012.

[43] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[44] C. J. C. H. Watkins, "Learning from delayed rewards," 1989.

[45] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*. Ieee, 1995, pp. 39–43.

[46] V. Sharma, S. Pattnaik, and T. Garg, "A review of bacterial foraging optimization and its applications," in *National Conference on Future Aspects of Artificial intelligence in Industrial Automation, NCFAAIIA*, 2012, pp. 9–12.

[47] L. Tan, H. Wang, C. Yang, and B. Niu, "A multi-objective optimization method based on discrete bacterial algorithm for environmental/economic power dispatch," *Natural Computing*, vol. 16, no. 4, pp. 549–565, 2017.

[48] B. Patle, A. Pandey, D. Parhi, A. Jagadeesh *et al.*, "A review: On path planning strategies for navigation of mobile robot," *Defence Technology*, vol. 15, no. 4, pp. 582–606, 2019.

[49] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on evolutionary computation*, vol. 1, no. 1, pp. 53–66, 1997.

[50] F. Kamil, T. S. Hong, W. Khaksar, M. Y. Moghrabiah, N. Zulkifli, and S. A. Ahmad, "New robot navigation algorithm for arbitrary unknown dynamic environments based on future prediction and priority behavior," *Expert Systems with Applications*, vol. 86, pp. 274–291, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417417303809

[51] H. Hewawasam, M. Y. Ibrahim, G. Kahandawa, and T. A. Choudhury, "Agoraphilic navigation algorithm in dynamic environment with and without prediction of moving objects location," in *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*, vol. 1, 2019, pp. 5179–5185.

[52] D. Bodhale, N. Afzulpurkar, and N. T. Thanh, "Path planning for a mobile robot in a dynamic environment," in *2008 IEEE International Conference on Robotics and Biomimetics*. IEEE, 2009, pp. 2115–2120.

[53] P. Wang, S. Gao, L. Li, B. Sun, and S. Cheng, "Obstacle avoidance path planning design for autonomous driving vehicles based on an improved artificial potential field algorithm," *Energies*, vol. 12, no. 12, p. 2342, 2019.

[54] L. Huang, "Velocity planning for a mobile robot to track a moving target — a potential field approach," *Robotics and Autonomous Systems*, vol. 57, no. 1, pp. 55–63, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889008000183

[55] M. A. K. Jaradat, M. H. Garibeh, and E. A. Feilat, "Autonomous mobile robot dynamic motion planning using hybrid fuzzy potential field," *Soft Computing*, vol. 16, no. 1, pp. 153–164, 2012.

[56] O. Montiel, U. Orozco-Rosas, and R. Sepúlveda, "Path planning for mobile robots using bacterial potential field for avoiding static and dynamic obstacles," *Expert Systems with Applications*, vol. 42, no. 12, pp. 5177–5191, 2015.

[57] B. Patle, D. Parhi, A. Jagadeesh, and S. K. Kashyap, "Matrix-binary codes based genetic algorithm for path planning of mobile robot," *Computers and Electrical Engineering*, vol. 67, pp. 708–728, 2018.

[58] S. X. Yang, Y. Hu, and M. Q.-h. Meng, "A knowledge based ga for path planning of multiple mobile robots in dynamic environments," in *2006 IEEE Conference on Robotics, Automation and Mechatronics*. IEEE, 2006, pp. 1–6.

[59] P. Shi and Y. Cui, "Dynamic path planning for mobile robot based on genetic algorithm in unknown environment," in *2010 Chinese control and decision conference*. IEEE, 2010, pp. 4325–4329.

[60] A. Aouf, L. Boussaid, and A. Sakly, "A pso algorithm applied to a pid controller for motion mobile robot in a complex dynamic environment," in *2017 International Conference on Engineering and MIS (ICEMIS)*. IEEE, 2017, pp. 1–7.

[61] Z. Bi, Y. Yimin, and X. Yisan, "Mobile robot navigation in unknown dynamic environment based on ant colony algorithm," in *2009 WRI Global Congress on Intelligent Systems*, vol. 3. IEEE, 2009, pp. 98–102.

[62] M. Faisal, R. Hedjar, M. Al Sulaiman, and K. Al-Mutib, "Fuzzy logic navigation and obstacle avoidance by a mobile robot in an unknown dynamic environment," *International Journal of Advanced Robotic Systems*, vol. 10, no. 1, p. 37, 2013.

[63] M. Faisal, K. Al-Mutib, R. Hedjar, H. Mathkour, M. Alsulaiman, and E. Mattar, "Multi modules fuzzy logic for mobile robots navigation and obstacle avoidance in unknown indoor dynamic environment," in *Proceedings of the 2013 International Conference on Systems, Control and Informatics*, 2013, pp. 371–379.

[64] V. M. Peri and D. Simon, "Fuzzy logic control for an autonomous robot," in *NAFIPS 2005-2005 Annual Meeting of the North American Fuzzy Information Processing Society*. IEEE, 2005, pp. 337–342.

[65] A. Zhu and S. X. Yang, "A fuzzy logic approach to reactive navigation of behavior-based mobile robots," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 5. IEEE, 2004, pp. 5045–5050.

[66] F. Abdessemed, M. Faisal, M. Emmadeddine, R. Hedjar, K. Al-Mutib, M. Alsulaiman, and H. Mathkour, "A hierarchical fuzzy control design for indoor mobile robot," *International Journal of Advanced Robotic Systems*, vol. 11, no. 3, p. 33, 2014.

[67] N. H. Singh and K. Thongam, "Mobile robot navigation using mlp-bp approaches in dynamic environments." *Arabian Journal for Science and Engineering (Springer Science and Business Media BV)*, vol. 43, no. 12, 2018.

[68] S. X. Yang and M. Meng, "An efficient neural network approach to dynamic robot motion planning," *Neural networks*, vol. 13, no. 2, pp. 143–148, 2000.

[69] I. Engedy and G. Horváth, "Artificial neural network based mobile robot navigation," in *2009 IEEE International Symposium on Intelligent Signal Processing*. IEEE, 2009, pp. 241–246.

[70] X. Chen and Y. Li, "Smooth formation navigation of multiple mobile robots for avoiding moving obstacles," *International Journal of Control, Automation, and Systems*, vol. 4, no. 4, pp. 466–479, 2006.

[71] M. A. K. Jaradat, M. Al-Rousan, and L. Quadan, "Reinforcement based mobile robot navigation in dynamic environment," *Robotics and Computer-Integrated Manufacturing*, vol. 27, no. 1, pp. 135–149, 2011.

[72] N. ALTUNTAŞ, E. Imal, N. Emanet, and C. N. Öztürk, "Reinforcement learning-based mobile robot navigation," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 24, no. 3, pp. 1747–1767, 2016.

[73] A. Demir and V. Sezer, "Intersection navigation under dynamic constraints using deep reinforcement learning," in *2018 6th International Conference on Control Engineering and Information Technology (CEIT)*. IEEE, 2018, pp. 1–5.

[74] L. Wang, Y. Liu, H. Deng, and Y. Xu, "Obstacle-avoidance path planning for soccer robots using particle swarm optimization," in *2006 IEEE International Conference on Robotics and Biomimetics*. IEEE, 2006, pp. 1233–1238.

[75] M. K. Rath and B. Deepak, "Pso based system architecture for path planning of mobile robot in dynamic environment," in *2015 Global Conference on Communication Technologies (GCCT)*. IEEE, 2015, pp. 797–801.

[76] M. A. Hossain and I. Ferdous, "Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique," *Robotics and Autonomous Systems*, vol. 64, pp. 137–141, 2015.

[77] F. H. Ajeil, I. K. Ibraheem, A. T. Azar, and A. J. Humaidi, "Grid-based mobile robot path planning using aging-based ant colony optimization algorithm in static and dynamic environments," *Sensors*, vol. 20, no. 7, p. 1880, 2020.

[78] T. Guan-Zheng, H. Huan, and A. Sloman, "Ant colony system algorithm for real-time globally optimal path planning of mobile robots," *Acta automatica sinica*, vol. 33, no. 3, pp. 279–285, 2007.

[79] S. Choi, E. Kim, and S. Oh, "Real-time navigation in crowded dynamic environments using gaussian process motion control," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 3221–3226.

[80] P. Yao, H. Wang, and Z. Su, "Real-time path planning of unmanned aerial vehicle for target tracking and obstacle avoidance in complex dynamic environment," *Aerospace Science and Technology*, vol. 47, pp. 269–279, 2015.

[81] B. Zhang and H. Duan, "Three-dimensional path planning for uninhabited combat aerial vehicle based on predator-prey pigeon-inspired optimization in dynamic environment," *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 14, no. 1, pp. 97–107, 2015.

[82] R. Bis, H. Peng, and A. G. Ulsoy, "Velocity occupancy space: autonomous navigation in an uncertain, dynamic environment," *International Journal of Vehicle Autonomous Systems*, vol. 10, no. 1-2, pp. 41–66, 2012.

[83] H. Naeem, J. Ahmad, and M. Tayyab, "Real-time object detection and tracking," in *INMIC*, 2013, pp. 148–153.

[84] W. Yi, M. R. Morelande, L. Kong, and J. Yang, "An efficient multi-frame track-before-detect algorithm for multi-target tracking," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 3, pp. 421–434, 2013.

[85] A. H. A. Rahman, H. Zamzuri, S. A. Mazlan, M. A. A. Rahman, and M. A. Zakaria, "Tracking uncertain moving objects using dynamic track management in multiple hypothesis tracking," in *2014 International Conference on Connected Vehicles and Expo (ICCVE)*, 2014, pp. 345–350.

[86] T. Chen, R. Wang, B. Dai, D. Liu, and J. Song, "Likelihood-field-model-based dynamic vehicle detection and tracking for self-driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 11, pp. 3142–3158, 2016.

[87] J. Dey and N. Praveen, "Moving object detection using genetic algorithm for traffic surveillance," in *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, 2016, Conference Proceedings, pp. 2289–2293.

[88] D. Crouse, "Basic tracking using nonlinear continuous-time dynamic models [tutorial]," *IEEE Aerospace and Electronic Systems Magazine*, vol. 30, no. 2, pp. 4–41, 2015.

[89] H. Kim and K. Sim, "A particular object tracking in an environment of multiple moving objects," in *ICCAS 2010*, 2010, Conference Proceedings, pp. 1053–1056.

[90] S. R. Balaji and S. Karthikeyan, "A survey on moving object tracking using image processing," in *2017 11th International Conference on Intelligent Systems and Control (ISCO)*, 2017, pp. 469–474.

[91] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, X. Zhao, and T.-K. Kim, *Multiple Object Tracking: A Literature Review*, 2017.

[92] R. Madhavan and C. I. Schlenoff, "Moving object prediction for off-road autonomous navigation," in *AeroSense 2003*, vol. 5083. SPIE, 2003, p. 12.

[93] R. Karima, A. Benabderrahmane, O. Azouaoui, and N. Ouadah, *Moving obstacles detection and tracking with laser range finder*, 2009.

[94] L. O. Russo, G. A. Farulla, M. Indaco, S. Rosa, D. Rolfo, and B. Bona, "Blurring prediction in monocular slam," in *2013 8th IEEE Design and Test Symposium*, 2013, pp. 1–6.

[95] J. S. Kulchandani and K. J. Dangarwala, "Moving object detection: Review of recent research trends," in *2015 International Conference on Pervasive Computing (ICPC)*, 2015, pp. 1–5.

[96] S. V. Amarasinghe, H. S. Hewawasam, W. B. D. K. Fernando, J. V. Wijayakulasooriya, G. M. R. I. Godaliyadda, and M. P. B. Ekanayake, "Vision based obstacle detection and map generation for reconnaissance," in *2014 9th International Conference on Industrial and Information Systems (ICIIS)*, 2014, pp. 1–6.

[97] c. Jayawardena, "A technical review of motion prediction methods for indoor robot navigation," 2015, Book.

[98] Q. Miao and Y. Gu, "Kernel-based online object tracking via gaussian mixture model learning," in *2016 Sixth International Conference on Instrumentation Measurement, Computer, Communication and Control (IMCCC)*, 2016, pp. 522–525.

[99] W. Liang, T. Tieniu, N. Huazhong, and H. Weiming, "Silhouette analysis-based gait recognition for human identification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 12, pp. 1505–1518, 2003.

[100] M. Keshmiri and M. Keshmiri, "Performance comparison of various navigation guidance methods in interception of a moving object by a serial manipulator considering its kinematic and dynamic limits," in *2010 15th International Conference on Methods and Models in Automation and Robotics*, 2010, Conference Proceedings, pp. 212–217.

[101] A. Elnagar, "Prediction of moving objects in dynamic environments using kalman filters," in *Proceedings 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation (Cat. No.01EX515)*, 2001, Conference Proceedings, pp. 414–419.

[102] D. Vasquez and T. Fraichard, "Motion prediction for moving objects: a statistical approach," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, vol. 4, 2004, pp. 3931–3936 Vol.4.

[103] A. F. Foka and P. E. Trahanias, "Probabilistic autonomous robot navigation in dynamic environments with human motion prediction," *International Journal of Social Robotics*, vol. 2, no. 1, pp. 79–94, 2010. [Online]. Available: https://doi.org/10.1007/s12369-009-0037-z

[104] R. Nicholas, N. Paul, and S. Siddhartha, *Modeling and Prediction of Pedestrian Behavior Based on the Sub-Goal Concept*. MITP, 2013, p. 1. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6578005

[105] C. Hermes, C. Wohler, K. Schenk, and F. Kummert, "Long-term vehicle motion prediction," in *2009 IEEE Intelligent Vehicles Symposium*, 2009, Conference Proceedings, pp. 652–657.

[106] U. K. J. H. S. Parekh, D. G. Thakore, "A survey on object detection and tracking methods," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. vol. 2, 2014.

[107] M. Marron, J. C. Garcia, M. A. Sotelo, M. Cabello, D. Pizarro, F. Huerta, and J. Cerro, "Comparing a kalman filter and a particle filter in a multiple objects tracking application," in *2007 IEEE International Symposium on Intelligent Signal Processing*, 2007, pp. 1–6.

[108] A. Elnagar, "Prediction of moving objects in dynamic environments using kalman filters," in *Proceedings 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation (Cat. No.01EX515)*, 2001, Conference Proceedings, pp. 414–419.

[109] J. Gómez-Ortega, D. R. Ramírez, D. Limón-Marruedo, and E. F. Camacho, "Genetic algorithms based predictive control for mobile robot navigation in changing environments," in *2001 European Control Conference (ECC)*, 2001, Conference Proceedings, pp. 3179–3184.

[110] Q. Li, R. Li, K. Ji, and W. Dai, "Kalman filter and its application," in *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*, 2015, pp. 74–77.

[111] D. H. Santosh and P. G. K. Mohan, "Multiple objects tracking using extended kalman filter, gmm and mean shift algorithm - a comparative study," in *2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies*, 2014, pp. 1484–1488.

[112] J. Almeida, A. Almeida, and R. Araujo, "Tracking multiple moving objects for mobile robotics navigation," in *2005 IEEE Conference on Emerging Technologies and Factory Automation*, vol. 1, 2005, pp. 8 pp.–210.

[113] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[114] J. Almeida and R. Araujo, "Tracking multiple moving objects in a dynamic environment for autonomous navigation," in *2008 10th IEEE International Workshop on Advanced Motion Control*, 2008, pp. 21–26.

[115] L. McFetridge and M. Y. Ibrahim, "Behavior fusion via free-space force shaping," in *IEEE International Conference on Industrial Technology, 2003*, vol. 2. IEEE, 2003, pp. 818–823.

[116] W. Siming, Z. Tiantian, and L. Weijie, "Mobile robot path planning based on improved artificial potential field method," in *2018 IEEE International Conference of Intelligent Robotic and Control Engineering (IRCE)*. IEEE, 2018, pp. 29–33.

[117] A. Pandey and D. R. Parhi, "Optimum path planning of mobile robot in unknown static and dynamic environments using fuzzy-wind driven optimization algorithm," *Defence Technology*, vol. 13, no. 1, pp. 47–58, 2017.

[118] J. Xin, X. Jiao, Y. Yang, and D. Liu, "Visual navigation for mobile robot with kinect camera in dynamic environment," in *2016 35th Chinese Control Conference (CCC)*, 2016, pp. 4757–4764.

[119] P. Viljamaa, *Fuzzy Gain Scheduling and Tuning of Multivariable Fuzzy Control: Methods of Fuzzy Computing in Control Systems*. Citeseer, 2000.

[120] J. E. Rodríguez-Castellanos and V. H. Grisales-Palacio, "A tuning proposal for fuzzy gain scheduling controllers for industrial continuous processes," in *2018 IEEE 2nd Colombian Conference on Robotics and Automation (CCRA)*, 2018, pp. 1–6.

[121] D.-L. Tsay, H.-Y. Chung, and C.-J. Lee, "The adaptive control of nonlinear systems using the sugeno-type of fuzzy logic," *IEEE Transactions on fuzzy systems*, vol. 7, no. 2, pp. 225–229, 1999.

[122] J. Arroyo and C. Maté, "Forecasting histogram time series with k-nearest neighbours methods," *International Journal of Forecasting*, vol. 25, no. 1, pp. 192–207, 2009.

[123] A. Khamparia, S. K. Singh, A. K. Luhach, and X.-Z. Gao, "Classification and analysis of users review using different classification techniques in intelligent e-learning system," *International Journal of Intelligent Information and Database Systems*, vol. 13, no. 2-4, pp. 139–149, 2020.

[124] M. H. Nampoothiri, P. G. Anand, and R. Antony, "Real time terrain identification of autonomous robots using machine learning," *International Journal of Intelligent Robotics and Applications*, vol. 4, no. 3, pp. 265–277, 2020.

[125] M. A. K. Jaradat, M. H. Garibeh, and E. A. Feilat, "Autonomous mobile robot dynamic motion planning using hybrid fuzzy potential field," *Soft Computing*, vol. 16, no. 1, pp. 153–164, 2012.