الجمهورية الجزائرية الديمقراطية الشعبية

People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

Université Ferhat ABBAS Sétif 1

UNIVERSITY OF SETIF 1 FERHAT ABBAS

FACULTY OF TECHNOLOGY

THESIS

Submitted to the Department of Electronics

In Fulfilment of the Requirements for the degree of

DOCTORATE

Domain: Science and Technology

Field: Electronic **Specialty: Electronic Instrumentation**

By

Souhaib LOUDA

THEME

Motion planning and fuzzy reasoning for autonomous navigation of a mobile robot

Defended on 09/11/2025 in front of Jury:

Lahcene ZIET	**Professor**	**Univ. Setif 1 Ferhat Abbas**	**President**
Nora KARKAR	**M.C.A.**	**Univ. Setif 1 Ferhat Abbas**	**Thesis director**
Fateh SEGHIR	**M.C.A.**	**Univ. Setif 1 Ferhat Abbas**	**Thesis co-director**
Laarem GUESSAS	**Professor**	**Univ. Setif 1 Ferhat Abbas**	**Examiner**
Foudil ABDESSEMED	**Professor**	**Univ. Batna 2**	**Examiner**
Ziyad BOUCHAMA	**Professor**	**Univ. M El Bachir El Ibrahimi, BBA**	**Examiner**
Khier BENMAHAMMED	**Professor**	**Univ. Setif 1 Ferhat Abbas**	**Guest**

الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

Ministère de L'Enseignement Supérieur et de la Recherche Scientifique

Université Ferhat ABBAS Sétif 1

UNIVERSITÉ DE SÉTIF 1 FERHAT ABBAS

FACULTÉ DE TECHNOLOGIE

THÈSE

Présentée au Département d'Électronique

Pour l'obtention du diplôme de

DOCTORAT

Domaine: Sciences et Technologie

Filière: Électronique **Option : Instrumentation Électronique**

Par

LOUDA Souhaib

THÈME

Planification du mouvement et raisonnement flou pour la navigation autonome d'un robot mobile

Soutenue le 09/11/2025 devant le Jury :

ZIET Lahcene	**Professeur**	**Univ. Sétif 1 Ferhat Abbas**	**Président**
KARKAR Nora	**M.C.A.**	**Univ. Sétif 1 Ferhat Abbas**	**Directrice de thèse**
SEGHIR Fateh	**M.C.A.**	**Univ. Sétif 1 Ferhat Abbas**	**Co-directeur de thèse**
GUESSAS Laarem	**Professeur**	**Univ. Sétif 1 Ferhat Abbas**	**Examinatrice**
ABDESSEMED Foudil	**Professeur**	**Univ. Batna 2**	**Examinateur**
BOUCHAMA Ziyad	**Professeur**	**Univ. M El Bachir El Ibrahimi, BBA**	**Examinateur**
BENMAHAMMED Khier	**Professeur**	**Univ. Sétif 1 Ferhat Abbas**	**Invité**

Dedication

"I would like to dedicate this thesis to

My mother, may Allah protect her,

In memory of my father, Allah Yerhamo inshallah, on July 11, 2016,

To my brothers and sisters, thank you for your encouragement and support

The readers of this thesis, for whom I hope it will be useful,

My entire family,

All those who taught me, I am very grateful,

"

Acknowledgements

First and foremost, I express my deepest gratitude to Allah, the Almighty, for granting me the strength, perseverance, and determination to carry out and complete this humble work. Without His divine guidance, this achievement would not have been possible. I would like to express my sincere and heartfelt thanks to my supervisor, **Dr. Nora KARKAR**, Lecturer at Setif 1 University Ferhat Abbas , for her invaluable guidance, unwavering support, and remarkable patience throughout this research. Her insightful advice and continuous encouragement have contributed a fundamental role in shaping the direction and depth of this dissertation. My profound thanks also go to my co-supervisor, **Dr. Fateh SEGHIR**, Lecturer at Setif 1 University Ferhat Abbas and head of the LSI Laboratory, for his generous support, thoughtful input, and ongoing availability. His contributions have been essential to the successful completion of this work.

I am also deeply grateful to all the distinguished members of the thesis jury for honoring me with their time and expertise. I thank **Prof. Lahcen ZIET**, Professor at Setif 1 University Ferhat Abbas, for the honor of presiding over the jury. I am equally thankful to **Prof. Laarem GUESSAS**, Professor at Setif 1 University Ferhat Abbas, for his critical review and valuable remarks. My sincere appreciation extends to **Prof. Foudil ABDESSEMED**, Professor at Batna 2 University, and **Prof. Ziyad BOUCHAMA**, Professor at Mohamed El Bachir El Ibrahimi University of BBA, for their interest in my research and their constructive evaluations.

I also warmly thank **Prof. Khier BENMAHAMMED** for accepting my invitation and attending as a guest, adding value to this important academic milestone. In addition, I would like to acknowledge **Dr. Oussama BOUTALBI**, Lecturer at Morsli Abdallah University of Tipaza, for his support and encouragement throughout this research journey.

My sincere thanks also go to all the professors of the department of electronics who have contributed to my academic development over the years. I am especially grateful to the members of the LSI Laboratory, including professors, researchers, and fellow PhD students, for their constant support, collaboration, and stimulating scientific discussions. Finally, I wish to thank all those who, in one way or another, contributed directly or indirectly to the realization of this thesis. Your support has been truly appreciated and will always be remembered.

Setif, Algeria
November, 2025

Souhaib LOUDA

Table of Contents

List of Figures

List of Tables

List of Algorithms

Acronyms and Symbols

Acronym	Definition	Acronym	Definition
ROS	Robot Operating System	**DWA**	Dynamic Window Approach
FLC	Fuzzy Logic Controller	**APF**	Artificial Potential Field
EKF	Extended Kalman Filter	**LiDAR**	Light Detection and Ranging
PID	Proportional-Integral-Derivative	**MPC**	Model Predictive Control
WMR	Wheeled Mobile Robot	**DFL**	Dynamic Feedback Linearization
EA*	Enhanced A*	**TDWA**	Traditional Dynamic Window Approach
IDWA	Improved Dynamic Window Approach	**TA***	Traditional A*
SLAM	Simultaneous Localization and Mapping	**COG**	Center of Gravity
MDPs	Markov Decision Processes	**MF**	Membership Function
MIMO	Multi Input Multi Output	**ABC**	Artificial Bee Colony
PRM	Probabilistic Roadmap	**RRT**	Rapidly Exploring Random Tree
TEB	Time Elastic Band	**VFH**	Vector Field Histogram
IMUs	Inertial Measurement Units	**RGB-D**	Red Green Blue Depth
UAVs	Unmanned Aerial Vehicles	**GPS**	Global Positioning System
ACO	Ant Colony Optimization	**SIFT**	Scale Invariant Feature Transform
SURF	Speeded Up Robust Features	**ORB**	Oriented FAST and Rotated BRIEF
PSO	Particle Swarm Optimization	**PF**	Particle Filter
LQR	Linear Quadratic Regulator	**FDFL**	Fuzzy Dynamic Feedback Linearization
RViz	ROS Visualizer	**CNNs**	Convolutional Neural Networks
RNNs	Recurrent Neural Networks	**FNNs**	Feedforward Neural Networks
SGD	Stochastic Gradient Descent	**GANs**	Generative Adversarial Networks
LSTM	Long Short-Term Memory	**RMS**	Root Mean Square
ReLU	Rectified Linear Unit	**DL**	Deep Learning
RL	Reinforcement Learning	**DQN**	Deep Q-Networks

Symbols:

C_{sound}, C_{light}	Sound and light speed	K	Intrinsic Matrix
T	Round-trip time	R	Rotation matrix
D	Distance to the object	t	Translation vector
f_x, f_y	Focal lengths in pixels	s	Scale factor
c_x, c_y	Coordinates of the principal point	f	Focal length
Z	Depth from camera	B	Baseline distance
d	Disparity	Q	State weighting matrix
R	Control input weighting matrix	K_p	Proportional gain
K_i	Integral gain	K_d	Derivative gain
γ	Discount factor	α	Learning rate
F	Flatness output	μ	Membership function
$d_{\min}(t)$	Minimum distance	u_{hybrid}	Hybrid control input
u_{DFL}	Control input from DFL	N_{tl}	Total local grid cells
N_{to}	Occupied local grid cells	σ	Standard deviation
v, ω	Linear and angular velocities	(x,y,θ)	Robot pose
v_L, v_R	Left/right wheel velocities	M_c, W_c	Controllability/Gramian matrix
$\phi(z)$	Diffeomorphism	λ	Eigenvalue
$v_{\min}$	Min linear velocity	$v_{\max}$	Max linear velocity
$w_{\min}$	Min angular velocity	$w_{\max}$	Max angular velocity
α, β, γ	Weight coefficients	$d_{\max}$	Max distance (threshold)
(x_s, y_s)	Start point coords	(x_c, y_c)	Current point coords
(x_g, y_g)	Goal point coords	s, n, g	Start, current, and goal nodes
d	Distance from n to g	D	Distance from s to g
$h(n)$	Heuristic function	$h'(n)$	Modified heuristic
$G(v, \omega)$	Evaluation function	$g(n)$	Actual cost
$f(n)$	Total cost	P_i	Control points
$x_d(t)$, $y_d(t)$	Desired smooth trajectory	$k_{p_1}, k_{p_2}, k_{d_1}, k_{d_2}$	Gain coefficients
v_{tr}, ω_{tr}	Tracking velocities	v_{av}, ω_{av}	Avoidance velocities
T	Execution time	dt	Sampling time
$G(n)$	Cost function		

General Introduction

Overview and Objectives:

Nowadays, mobile robot navigation plays a crucial role in robotics, involving the autonomous guidance of a robot from its starting location to destination within its environment [1–5]. Typically, a navigation system comprises four main modules: localization, mapping, planning, and motion control. As a critical module, the path and motion planning module aims to determine an optimal or suboptimal collision-free path from the starting position to the target position [6–14]. In the first part, path planning helps robots map out a path as straight as possible from a starting point A to a target configuration B while avoiding collision with obstacles. Path planning techniques may be classified into global and local path planning [15–17].

Global path planning is an offline planning approach that determines the optimal solution from a starting point to a specified destination within a static and known environment or a global map [18, 19]. Global path planning methods can be categorized into different approaches. Graph-based methods include Dijkstra's algorithm [20] and the A* algorithm [21], while sampling-based methods encompass Rapidly-exploring Random Trees (RRTs) [22] and Probabilistic Roadmaps (PRMs) [23]. Additionally, evolutionary-based optimization techniques, such as Particle Swarm Optimization (PSO) [24], Ant Colony Optimization (ACO) [25], and Artificial Bee Colony (ABC) [26], have been widely explored for global path planning.

The A* graph search algorithm is one of the most widely used for optimal global path planning due to its high search efficiency, fast computation, simplicity, ease of implementation, and broad applicability [27]. However, the traditional A* algorithm has several drawbacks, including an excessive number of search nodes, redundant path nodes in the generated path, and poor path smoothness and low safety. These limitations not only affect the stability of autonomous mobile robots but also reduce the overall efficiency

of path planning. Therefore, further improvements to the A* algorithm are essential to address these limitations and enhance its efficiency, safety, and path quality. In recent years, numerous studies have made significant advancements in refining the A* algorithm to mitigate these challenges and improve its overall performance.

Yin et al. [28] introduced an improved evaluation function along with a bidirectional search strategy to optimize the search direction. These enhancements reduced the number of explored nodes and eliminated redundant path nodes, thereby increasing search efficiency. Additionally, cubic uniform B-spline curves were employed to smooth the path by removing unnecessary inflection points. Yang et al. [29] proposed a global path planning method that minimizes the risk index to address key limitations of the A* algorithm, such as low safety and poor path smoothness. Their approach reduces the number of trajectory inflection points, enhancing both safety and smoothness. Sun et al. [30] proposed an improved A* algorithm for robot path planning in various environments. They introduced different strategies to enhance search efficiency, significantly increasing the speed of path planning while reducing the planned path length and improving the algorithm's reliability. Li et al. [31] proposed an improved A* algorithm that integrates the jump point search strategy with an adaptive arc optimization strategy. This approach addresses challenges such as excessive node expansion, high memory overhead, low computational efficiency, and numerous path corners. The enhanced algorithm outperforms traditional A* in terms of path length, safety, and smoothness.

Local path planning dynamically adjusts a robot's trajectory using sensor data. Unlike global planning, it continuously updates the path to avoid obstacles and ensure safe navigation. There exist Various methods and approaches in this class of planning systems, including the Dynamic Window Approach (DWA) [32,33], Artificial Potential Field (APF) [34], Time Elastic Band (TEB) [35], Fuzzy Logic Controller (FLC) [36], as well as advanced techniques like Reinforcement Learning (RL) [37,38] and Deep Learning (DL) [39].

The Dynamic Window Approach (DWA) is one of the most widely used methods for local path planning, primarily due to its real-time efficiency and its ability to balance obstacle avoidance with smooth navigation. However, the DWA in its traditional form exhibits several limitations, such as susceptibility to local minima, difficulty in navigating narrow passages, and occasional failure to reach the target. Moreover, it demonstrates limited capability in effectively avoiding dynamic obstacles in complex envi-

ronments. Furthermore, the use of fixed weight coefficients in the DWA evaluation function reduces the algorithms ability to adapt to varying environmental conditions. This rigidity often results in suboptimal performance, including longer path lengths and increased planning time. Such limitations not only compromise the efficiency of the robot but also restrict its adaptability in dynamic and unpredictable environments. Consequently, enhancing the DWA algorithm is crucial for improving its planning efficiency, adaptability, and path quality. In recent years, a growing body of research has been dedicated to refining the DWA framework to overcome these challenges and improve its overall performance.

Gong et al. [40] proposed an improved dynamic window approach (DWA) for local obstacle avoidance in robots. This enhancement addresses the issue where the original DWA algorithm struggles to balance safety and velocity due to its fixed parameters in complex environments filled with numerous obstacles. The proposed method allows for more rational path selection and enhances the safety of inspection robots. Additionally, it optimizes the safe distance, path length, and the number of samples used. Xu et al. [41] propose an enhanced version of Dynamic Weighted A* (DWA) that dynamically adjusts the combination of weight coefficients using fuzzy control logic. This optimization improves the evaluation function and enhances the algorithm's performance by providing safe and dynamic obstacle avoidance. The proposed approach effectively reduces both path length and planning time while ensuring effective obstacle avoidance in various dynamic environments. Kobayashi et al. [42] propose an optimized Dynamic Window Approach (DWA) that utilizes reinforcement learning to dynamically adjust the fixed weights in the evaluation function. This method addresses challenges faced by traditional DWA, such as congestion levels, road width, and obstacle density. The proposed solution employs a pre-learned Q-table, which includes robot states, environmental conditions, and action weight coefficients. The effectiveness of this approach in local path planning is demonstrated. Abubakr et al. [43] propose an adaptive dynamic window approach (DWA) for mobile robot dynamic obstacle avoidance. Their method is optimized using a fuzzy logic controller, which dynamically adjusts the fixed weights of the evaluation function to adapt to different environmental scenarios. This approach overcomes limitations such as long planning times and excessive path lengths, allowing for safe navigation around dynamic obstacles.

From another point of view, most of the above references focus on improving traditional Dynamic Window Approach (DWA) methods without

considering global optimality. To address this issue, the authors propose a fusion approach that combines enhanced A* and improved DWA techniques to resolve the cited problems. Guo et al. [44] propose a fusion approach that combines an enhanced A* algorithm with an improved Dynamic Window Approach (DWA). In this method, the key points extracted from the optimal global path serve as sub-targets for the local path. This guidance enables the robot to reach its goal quickly and prevents it from getting trapped in local minima. Wang et al. [45] propose a hybrid method that combines the A* algorithm with fuzzy Dynamic Window Approach (DWA). In this method, the confidence weights of the evaluation function are adjusted using fuzzy control to better respond to changes in the environment. This approach is particularly effective in dynamic environments.

This study has several limitations. It does not consider the complexities of dynamic environments, and certain metrics, such as planning time and path length, are already lengthy. To address these issues, we propose an intelligent approach that combines the Fuzzy Dynamic Window Approach (DWA) with Adaptive A*. Additionally, we have suggested another fusion method that utilizes improved A* and improved DWA to accommodate various levels of environmental complexity.

In the second part, motion planning plays a pivotal role in determining the precise sequence of actions that a robot must perform to follow a predefined path and reach its target destination efficiently. This research focuses specifically on Differential Wheeled Mobile Robots (DWMRs), a class of mobile robots that has received considerable attention in recent studies due to their simplicity, maneuverability, and widespread application in motion control research. [46–51]. Within this domain, achieving accurate trajectory tracking and ensuring safe navigation remain fundamental objectives that have been the focus of extensive research. [52–55].

The tracking control problem in mobile robotics centers on the development of robust control strategies that enable a robot to accurately follow a predefined path or trajectory [56–58]. The principal objective is to minimize the deviation between the robots actual trajectory and the desired one throughout the navigation process. Achieving effective tracking requires continuous real-time adjustments to the robots pose, velocity, and heading to ensure close adherence to the planned trajectory.

Safe navigation tasks prioritize ensuring that a mobile robot can reach its destination while systematically avoiding collisions and adhering to predefined safety protocols [59–61]. This involves dynamic obstacle detection and avoidance, real-time path adjustments, and maintaining appropri-

ate speeds and safety margins in both static and dynamic environments. To achieve robust and reliable navigation, robots utilize a variety of sensorssuch as LiDAR, cameras, and sonar to perceive their surroundings and make informed decisions. These decisions account for the robots geometry, velocity, and dynamic constraints to ensure sufficient clearance from obstacles.

The presence of non-holonomic kinematic constraints, inherent to these non-integrable systems, has driven the development of advanced nonlinear control methodologies [62]. Such robots pose unique challenges that significantly restrict the solution space for both trajectory planning and control design. Specifically, they belong to a class of systems that encounter several fundamental difficulties: first, they cannot be stabilized using smooth, time-invariant controllers; second, controllers designed for trajectory tracking cannot be directly repurposed for set-point stabilization, and vice versa. Consequently, careful consideration must be given during the design of reference trajectories to ensure the feasibility and successful execution of the intended tasks [63, 64].

Numerous effective approaches have been developed to address the regulation and trajectory tracking challenges in mobile robots. These include Sliding Mode Control (SMC) [65–67], backstepping techniques [68], Fuzzy Logic Controllers (FLCs) [69], and time-varying controllers grounded in Lyapunov stability theory [70, 71]. Comprehensive overviews of fundamental methods for trajectory tracking and feedback motion control can be found in [63, 72].

Among these methods, Dynamic Feedback Linearization (DFL) stands out as a powerful control design tool, particularly suited for open-loop trajectory planning [73]. Fundamental challenges in trajectory tracking and set-point regulation are comprehensively addressed in [74–76]. A key characteristic of feedback-linearizable systems is the property of differential flatness [77, 78], which is intrinsically linked to the system's controllability structure. Building upon Brunovskys canonical forms, this analytical framework has proven invaluable for the analysis and design of both open-loop and stabilizing feedback control laws in nonlinear finite-dimensional systems. Its applicability spans a wide range of control problems, from linear systems to highly nonlinear controlled systems [79–84]. The flatness property has been effectively employed for the regulation of continuous nonlinear systems and has consistently shown strong performance in trajectory tracking tasks [85, 86].

The obstacle avoidance task [87] is crucial for ensuring the safe move-

ment of mobile robots and preventing potential collisions. Various techniques have been developed to address this challenge, including reinforcement learning [88], neural networks [89], the Artificial Potential Field (APF) method [34, 90], the Dynamic Window Approach (DWA) [32], the Time Elastic Band (TEB) method [35], and the Vector Field Histogram (VFH) [91].

The Fuzzy Logic Controller (FLC), introduced by mathematician Lotfi Zadeh in 1965 [92], is a well-established control technique for obstacle avoidance. Its adoption in this work is primarily motivated by its simplicity of implementation and high adaptability to diverse and dynamic environments.

Research Contributions:

To achieve robust path planning, accurate trajectory tracking, and reliable collision avoidance in both static and dynamic environments within the ROS middleware framework, the key contributions of this research are as follows:

- **Enhanced global path planning based on A* algorithm:** An improved heuristic function is proposed to accelerate the A* search algorithm by reducing the number of explored nodes. The resulting path is further refined through redundant waypoint removal and smoothed using cubic B-spline curves. This process ensures path continuity, enhances safety margins, and increases the feasibility of real-world deployment.
- **Optimized local path planning via Fuzzy Logic controller:** A fuzzy logic-based dynamic weight adjustment strategy is introduced to improve the evaluation function of the Dynamic Window Approach (DWA). This method adaptively adjusts parameters based on the robot's distance and heading to the goal, as well as its proximity to obstacles, enhancing local planning performance in complex and changing environments.
- **Fuzzy-Dynamic Feedback Linearization (FDFL):** A novel hybrid control framework, termed FDFL, is developed. It integrates fuzzy logic with dynamic feedback linearization to achieve simultaneous trajectory tracking and obstacle avoidance, tailored specifically for differentially driven mobile robots.

- **ROS-Gazebo simulation and Matlab integration:** The proposed FDFL method is implemented using MATLAB within the ROS environment and evaluated in real-time using the ROS-Gazebo simulator. The strong consistency between MATLAB and Gazebo results confirms the validity and reliability of the approach, while respecting the kinematic constraints of the mobile platform.
- **Real-Time experimental validation:** The effectiveness of the proposed framework is demonstrated in a real-world experimental setup using ROS. A high-performance master PC handles computationally intensive tasks, while a lower-level slave laptop onboard the robot executes control commands. Both systems communicate wirelessly, ensuring real-time responsiveness and coordination.

Thesis Organization:

To address the challenges of autonomous navigation in mobile robots, this thesis is structured into five main chapters, each building progressively upon the previous one. Chapter 1 provides a comprehensive review of intelligent navigation systems for mobile robots, with a focus on state-of-the-art methods and enabling technologies. Chapter 2 explores low-level control strategies, emphasizing trajectory tracking and obstacle avoidance through the integration of fuzzy logic and Dynamic Feedback Linearization (DFL). In Chapter 3, an intelligent navigation framework is introduced that combines an adaptive A* algorithm with a fuzzy-enhanced Dynamic Window Approach (DWA) for efficient path planning in static environments. Chapter 4 extends this framework to dynamic environments by integrating an improved A* algorithm with an enhanced DWA. This enables real-time path re-planning in the presence of moving obstacles, thereby increasing navigation robustness and safety. Chapter 5 presents an intelligent fuzzy-DFL control system implemented under the ROS middleware, aimed at improving trajectory tracking and dynamic obstacle avoidance performance in real-time applications. Finally, the concluding chapter summarizes the key contributions of the thesis and outlines promising directions for future research.

1 Autonomous Navigation System for Mobile Robots: State-of-the-art

This chapter explores the fundamental elements of autonomous navigation for mobile robots. It presents an overview and examines the essential modules required to achieve autonomous operation, including perception, localization and mapping, path and motion planning, motion control, and decision-making.

Contents

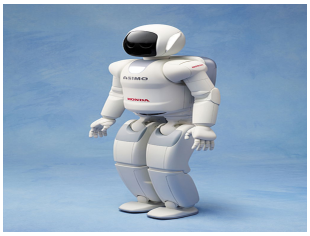
1.1 Introduction

Autonomous navigation is a fundamental capability for mobile robots, enabling them to perform tasks in diverse and often unpredictable environments without direct human control. Achieving reliable navigation requires the integration of multiple interrelated components, each addressing specific challenges. This chapter systematically explores these components, starting with the basic principles of mobile robot navigation. It then delves into perception methods that allow robots to sense and interpret their surroundings, followed by localization and mapping techniques that provide spatial awareness and environmental representation. Path planning algorithms are examined next, focusing on generating feasible and efficient routes. To execute these plans, motion control and trajectory tracking strategies are discussed, ensuring accurate and smooth robot movements. Finally, the chapter highlights the role of reasoning and decision-making frameworks in enabling adaptive and intelligent navigation behaviors. Together, these topics provide a comprehensive foundation for understanding and advancing autonomous navigation systems in mobile robotics.

1.2 Mobile Robot Navigation

This section presents the different categories of robots based on their locomotion mechanisms and operational workspaces. Broadly, robots can be classified into three main types: fixed (non-mobile) robots, such as robotic arms and industrial manipulators, and mobile robots, which include wheeled ground robots, aerial drones, and underwater robots. A third category comprises hybrid robots, which combine features of both typesfor example, a wheeled mobile robot equipped with a robotic manipulator. In this work, our focus is on differential-drive mobile robots, commonly modeled as unicycle-type robots. The section also introduces the principal classes of autonomous navigation strategies described in the literature. These include offline deliberative approaches, which follow the sense-plan-act paradigm; online reactive approaches, which are based on the sense-act paradigm; and hybrid approaches, which integrate both paradigms to leverage their respective advantages. Autonomous navigation in mobile robots encompasses the ability to perceive and interpret the environment, estimate the robot's pose within a workspace, plan an optimal or feasible path toward a target, and execute that path reliably while avoiding obstacles and adapting to dynamic changes. [93–95].

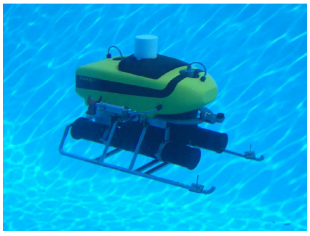
(a) Manipulator arm (Kuka)



(b) Humanoid robot (Asimo)

(c) Mobile Manipulator (Pal Robotics Tiago)

(d) Flying robot (Amazon)



(e) Underwater robot

(f) Underground robot (Wheeled robot + arm)

Figure 1.1: Different types of robots.

Wheeled Mobile Robots (WMRs) is the famous category in autonomous navigation due to their simplicity ease of control and reduced energy consumption [96, 97]. Additionally, their high accuracy makes them widely used in various applications, including locomotion, transportation, logistics, and navigation.

1.2.1 Classification of mobile robots

In the field of robotics, robots can be classified into three main categories [98]. The first category includes fixed robots, such as manipulators and robotic arms [99]. The second category consists of mobile robots [100], including humanoid robots, wheeled robots, underwater robots, unmanned robots and legged robots. The third category comprises hybrid robots (a combination of fixed and mobile robots), such as a wheeled mobile robot equipped with a robotic arm [101], as shown in Figure 1.1.

1.2.2 Wheeled mobile robots

Wheeled Mobile Robots (WMRs) are a class of mobile robots that navigate across flat surfaces using motorized wheels as their primary means of locomotion. These robots are widely used due to their mechanical simplicity, energy efficiency, and ease of control, making them ideal for indoor

and structured environments. [102, 103] . This design is simpler than using treads or legs and by using wheels they are easier to design, build, and program for movement in flat, not-so-rugged terrain. They are also more well controlled than other types of robots. Disadvantages of wheeled robots are that they can not navigate well over obstacles, such as rocky terrain, sharp declines, or areas with low friction. Wheeled robots are most popular among the consumer market, their differential steering provides low cost and simplicity. Robots can have any number of wheels, but three wheels are sufficient for static and dynamic balance. Additional wheels can add to balance; however, additional mechanisms will be required to keep all the wheels in the ground, when the terrain is not flat. It exist many types based on locomotion and wheels (steering), such as differential drive robots [104, 105] , Car-like mobile robots [106], and Omnidirectional robots [107] as shown in Figure 1.2. Most wheeled robots use differential steering, which uses separately driven wheels for movement. They can change direction by rotating each wheel at a different speed. There may be additional wheels that are not driven by a motor these extra wheels help keep it balanced.

Figure 1.2: Different types of wheeled mobile robots (Differential drive in the left, Car-like drive in the center, Omnidirectional drive in the right).

1.2.3 Autonomous navigation of a mobile robots: Overview and challenges

Autonomous navigation refers to the capability of mobile robots to move and operate independently within their environment without human intervention. This complex task encompasses several core components, including localization, obstacle avoidance, mapping, and motion planning. For a mobile robot to navigate efficiently and accurately, it must be able to precisely estimate its position, construct or interpret a map of the environment, compute an optimal or feasible path from a starting point to a target location, and execute that path through an appropriate control strategy.

Autonomous navigation strategies can be broadly categorized into three main types:

- **Deliberative navigation**: follows a sense-plan-act paradigm and relies on detailed world models.
- **Reactive navigation**: based on a sense-act paradigm for fast, real-time responses to sensory input without global planning.
- **Hybrid navigation**: integrates both deliberative and reactive approaches to combine global planning with local adaptability.

These categories are illustrated in Figure 1.3, highlighting their respective architectures and decision-making processes.

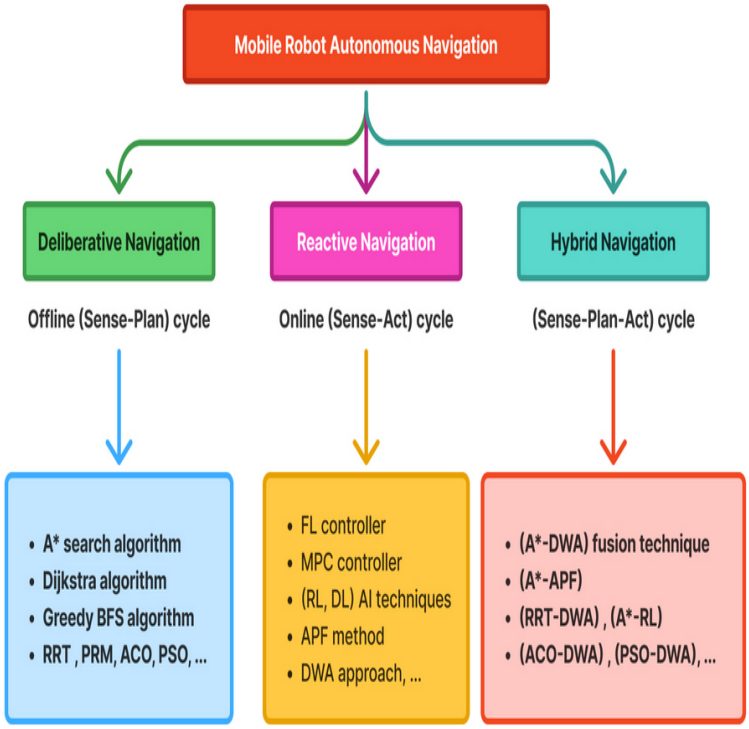


Figure 1.3: Autonomous navigation of mobile robot.

1.3 Perception

Perception in mobile robot navigation is a crucial component for successful autonomous movement. It refers to the robots ability to understand its surroundings and estimate the state of the environment or the generated map. The quality of perception largely depends on the accuracy of the sensors used, such as laser range finders, cameras, encoders, and others [108, 109].

1.3.1 Mobile robot sensors

Sensors are devices that are capable of perceiving and converting environmental information into electrical signals or other required formats according to specific rules, as well as transmitting them to other devices. Robots utilize a variety of sensors to detect different aspects of their environment. Generally, sensors are divided into two main categories based on their operating principles. Proprioceptive sensors are used to measure the internal values of a dynamic system (such as a robot), like motor speed, robot arm joint angles, and robot pose . The proprioceptive sensors used in robot control applications, including Inertial Measurement Units (IMUs), magnetometers, accelerometers, and gyroscopes. Exteroceptive sensors, in contrast, acquire information from a robots environment, such as, measurements of distance, light intensity, sound amplitude, temperature, force magnitude, gas concentration, and image information. Therefore, the measurements obtained from exteroceptive sensors are interpreted by the robot to extract meaningful environmental features. In robot control applications, exteroceptive sensors include ultrasonic, infrared, LiDAR, and vision sensors [110].

1.3.2 Wheel encoder

A wheel encoder is a sensor used in mobile robots to measure the rotation of the wheels. This information is crucial for odometry, a localization technique that estimates the robots position and orientation (pose) over time by integrating the wheel movements. Wheel encoders typically output pulses or counts corresponding to incremental wheel movements, allowing the robot to estimate distance traveled and change in orientation [111]. The operational principle of an optical encoder as illustrated in the Figure 1.4 Assuming a differential drive robot with two wheels, the following notations and equations are used:

- R : Radius of each wheel (in meters)
- L : Distance between the two wheels (wheelbase)
- $\Delta\phi_L$, $\Delta\phi_R$: Angular displacement of the left and right wheels (in radians)
- $\Delta s_L = R \cdot \Delta\phi_L$: Linear distance traveled by the left wheel
- $\Delta s_R = R \cdot \Delta\phi_R$: Linear distance traveled by the right wheel

- $\Delta s = \dfrac{\Delta s_R + \Delta s_L}{2}$: Linear displacement of the robot
- $\Delta\theta = \dfrac{\Delta s_R - \Delta s_L}{L}$: Change in orientation of the robot

Given the robot's previous pose (x, y, θ), the updated pose (x', y', θ') is computed as:

$$\begin{aligned} x' &= x + \Delta s \cdot \cos\left(\theta + \frac{\Delta\theta}{2}\right) \\ y' &= y + \Delta s \cdot \sin\left(\theta + \frac{\Delta\theta}{2}\right) \\ \theta' &= \theta + \Delta\theta \end{aligned} \tag{1.1}$$

If the encoder provides tick counts, the angular displacement of a wheel is:

$$\Delta\phi = \frac{2\pi \cdot \Delta\text{ticks}}{N} \tag{1.2}$$

where:

- Δticks : Number of encoder ticks during a time interval
- N : Encoder resolution (ticks per revolution)

Thus, the linear distance traveled by a wheel is:

$$\Delta s = R \cdot \left(\frac{2\pi \cdot \Delta\text{ticks}}{N}\right) \tag{1.3}$$

(a) Optical encoder working principle

(b) Wheel optical encoder

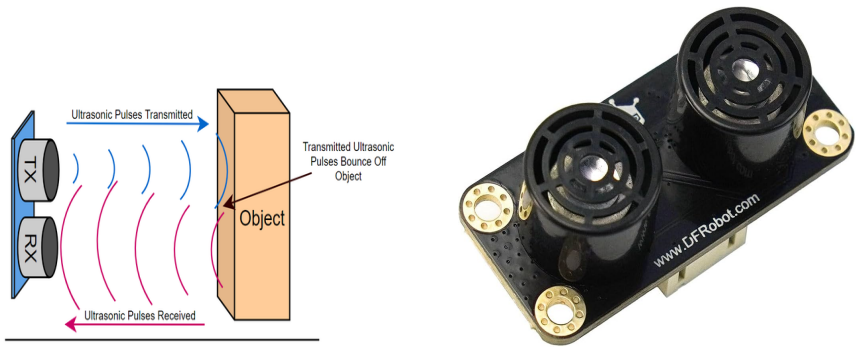
Figure 1.4: Encoder sensor.

1.3.3 Ultrasonic range sensor

An ultrasonic sensor is a device which utilizes ultrasonic waves to measure distance or detect objects. In industrial automation technology, ultrasonic sensors are particularly used for precise control in production processes. Ultrasonic sensors reliably measure without contact a distance within the defined range, no matter which object properties and ambient conditions such as the presence of dust. For doing so, an ultrasonic sensor emits high-frequency sound waves and measures the time it takes the waves to hit an object and to reflect back to the sensor (time-of-flight principle). Based on the time measured, the distance between sensor and object is calculated by the speed of sound propagation. In this way, ultrasonic sensors precisely detect the presence and position of objects. The measuring range of ultrasonic sensors varies according to sensor principle, model and environmental factors; typically between a few centimeters and several meters. Key for the measuring range are frequency and amplitude of the ultrasonic waves as well as transmitter performance and receiver sensitivity [112]. The distance is measured by detecting and calculating the time elapsed between ultrasonic wave emission and reception as depicted in Figure 1.5. The actual distance to the object is expressed as follows:

$$D = \frac{C_{sound} \cdot T}{2} \tag{1.4}$$

Where, D represents the distance to the object, T is the time interval between the emission and reception of the sound wave, and C (where $C_{sound} = 343$ m/s) denotes the speed of sound in air. Since T corresponds to the round-trip time (i.e., the time taken for the sound wave to travel to the object and back),



(a) Working principle of an ultrasonic sensor

(b) Ultrasonic sensor module

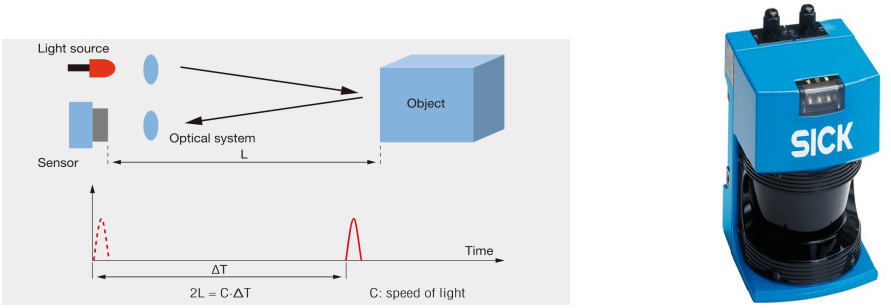
Figure 1.5: Ultrasonic range sensor.

1.3.4 Laser rangefinder

A laser rangefinder, also known as a laser telemeter or laser distance meter, is a rangefinder that uses a laser beam to determine the distance to an object. The most common form of laser rangefinder operates on the time of flight principle by sending a laser pulse in a narrow beam towards the object and measuring the time taken by the pulse to be reflected off the target and returned to the sender [113]. Due to the high speed of light, this technique is not appropriate for high precision sub-millimeter measurements, is shown in Figure 1.6, where triangulation and other techniques are often used instead Laser rangefinders are sometimes classified as type of handheld scanner less. The actual distance to the object is given as follows:

$$D = \frac{C_{light} \cdot T}{2} \tag{1.5}$$

Where, D signifies the distance to the object, T indicates the time interval between the emission and reception of the infrared light wave, and C (where $C_{light} = 3 \times 10^8$ m/s) represents the speed of light in a vacuum. Since T measures the round-trip time (i.e., the time taken for the signal to travel to the object and back).



(a) Working principle of an laser sensor **(b)** Sick LMS 200 LiDar

Figure 1.6: Laser range finder sensor.

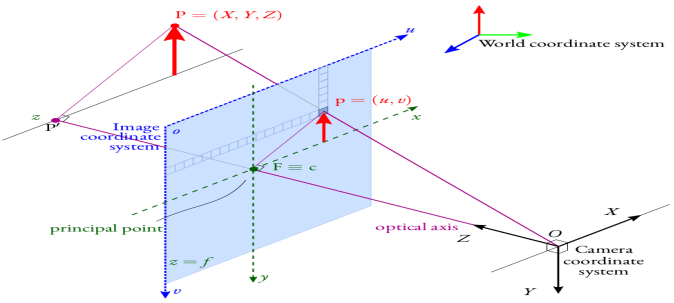
1.3.5 Camera vision

An optical sensing tool, a camera records light coming from a scene and turns it into a digital image or video. In mobile robotics, it functions as a visual sensor allowing the robot to obtain real-time visual data and thereby detect and interpret its environment. Task including object detection, localization, mapping, and navigation in both organized and unstructured

settings depend on cameras; they can be monocular (single lens), stereo (dual lens), RGB-D (giving depth information). [114].

In mobile robot navigation, camera vision denotes the utilization of cameras and image processing methodologies to facilitate a robot's perception, interpretation, and interaction with its surroundings for autonomous navigation. This vision-based technology enables the robot to perceive and interpret its surroundings, allowing it to identify and recognize obstacles, navigate through environments, determine its location within a given area, and make decisions based on visual information. [93, 115–119].

The pinhole camera model, depicted in Figure 1.7, is a prevalent mathematical abstraction in computer vision and mobile robotics. It describes the projection of a three-dimensional (3D) point onto a two-dimensional (2D) picture plane through the concepts of projective geometry. This model assumes an idealized camera without lens distortion, where light rays pass through a single point (the pinhole) to form an image on a flat image plane. [120].



(a) Pinhole camera model

(b) Active Stereo camera vision (MS Kinect v1)

Figure 1.7: Camera vision sensor.

Let a 3D point in the world coordinate system be represented as $\mathbf{P} = [X\ Y\ Z]^T$. The projection of this point onto the 2D image plane results in image coordinates $\mathbf{p} = [u\ v]^T$. Under the pinhole camera model, the projection equations are:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \end{bmatrix} \tag{1.6}$$

where:

- f_x, f_y are the focal lengths in pixels along the x and y axes,

- c_x, c_y are the coordinates of the principal point (image center),
- Z is the depth of the point from the camera along the optical axis.

The intrinsic parameters of the camera can be expressed in the form of a matrix $\mathbf{K}$:

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1.7}$$

In homogeneous coordinates, the complete projection of a 3D world point onto the image plane is given by:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1.8}$$

where:

- s is a scale factor,
- $\mathbf{R}$ is the rotation matrix,
- $\mathbf{t}$ is the translation vector,
- $[\mathbf{R} \mid \mathbf{t}]$ is the extrinsic matrix that transforms coordinates from the world to the camera frame.

In monocular systems, the depth Z cannot be calculated directly from a single image [121]. However, in stereo vision system, the depth can be estimated by triangulating the position of a point seen in both the left and right camera images [122].

The depth Z is computed using the following formula:

$$Z = \frac{f \cdot B}{d} \tag{1.9}$$

where:

- f is the focal length of the camera (in pixels),

- B is the baseline distance between the two stereo cameras (in meters),
- d is the disparity, i.e., the difference in horizontal pixel positions of the same point in the left and right images.

This equation assumes that the cameras are calibrated and rectified so that corresponding points lie on the same horizontal scanline.

1.3.6 Sensors fusion process

Sensor fusion stands as a key part in evolving navigation systems as illistrated in Figure 1.8, bringing together different data streams into one clear understanding of surroundings. By putting together strengths of various sensors, from wide views of LIDAR to detailed capture by cameras, robots obtain a fuller view of their surroundings. [123–125]

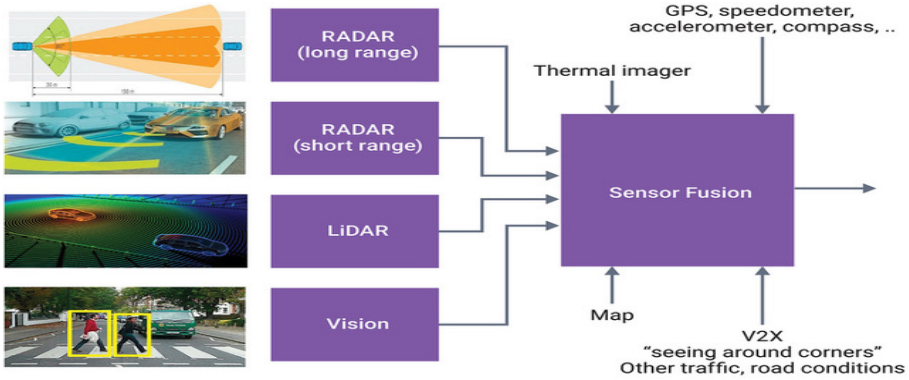


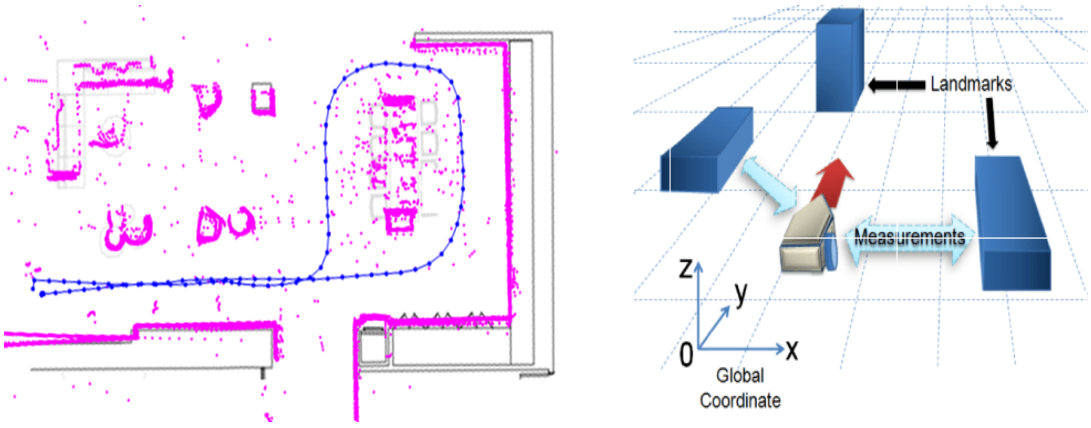
Figure 1.8: Sensor fusion process.

1.4 Localization

The fundamental principles of robotics center around mobile robots autonomously navigating, mapping, and performing tasks, making localization a critical component, as illustrated in Figure 1.9. Localization refers to the process of determining a robots position and orientation within a given local frame relative to a defined reference frame. It is a fundamental capability that ensures reliable operation and enables the robot to navigate safely and make informed decisions in both static and dynamic environments.

By considering the onboard sensors and available environmental information, establishing a frame of reference allows for the estimation of the robots spatial coordinates and orientation. This reference frame can be either a predefined environment or a global map. Reducing the ambiguity

or uncertainty associated with position estimation is essential, as it ensures a higher level of confidence in the robots ability to navigate and interact safely and effectively within its surroundings. [126].



(a) Localization with 2D scan **(b)** Localization based landmarks

Figure 1.9: Mobile robot localization.

1.4.1 Localization sensors and techniques

1.4.2 Wheel odometry

Wheel odometry refers to odometry (i.e., estimating motion and position) using rotary encoders (i.e., sensors that attach to the motors of the wheels to measure rotation). Its a useful technique for localization with wheeled robots and autonomous vehicles [111].

1.4.3 GPS/GNSS

The Global Positioning System (GPS) is a satellite-based navigation system that allows users to determine their precise location considering the latitude, longitude, and altitude anywhere on the Earth [127]. GPS trackers or GPS tracking systems utilize the technology to determine the location of the asset or vehicle in real time.GPS tracking systems function by utilizing a network of satellites to pinpoint the precise location of a device equipped with a GPS receiver. The receiver collects signals from multiple satellites, typically at least four, to triangulate its exact position using the time it takes for the signals to reach it. Simply put, GPS tracking devices or systems let you know the live location, and speed of your vehicle. It even helps with vehicle diagnostics, ascertaining how many stops or halts were taken throughout the assigned trip. It does more than you can imagine.

1.4.4 Visual odometry

Visual odometry In robotics and computer vision, visual odometry is the process of determining the position and orientation of a robot by analyzing the associated camera images. It has been used in various robotic scenarios such as indoor navigation, multiple robots systems and dynamic environments. Visual Odometry is the process of estimating the robots motion by analyzing consecutive camera images. It estimates the robots path incrementally by comparing the translation between successive frames. This approach is similar to wheel odometry but employs visual data, which is useful in a context where wheel slip or poor traction causes error in most standard odometric methods [128].

The feature involves finding distinctive parts of images (corners, edges) and tracking their displacement between frames to determine where motion is taking place. SIFT, SURF, ORB are popular feature detectors and direct methods. These methods operate directly on the intensity values of all pixels instead of relying on discrete features. Their goals are to minimize photometric inconsistency in exposure frames.

1.4.5 Pose estimation

Pose estimation is the process of determining a position and orientation of a mobile robot in an environment or map. The pose is typically represented as , where x and y are the coordinates of the robot on a 2D plane, and is its orientation θ (heading angle) with respect to some reference direction. Accurate pose estimation is a prerequisite for learning to move around autonomously, localizing, mapping, and path planning. But since odometry as well as individual sensors are inexact and noisy, their estimates of poses are also often inaccurate and drift hence accumulation of error over time. To this end filtering techniques like the Kalman Filter, Extended Kalman Filter (EKF), and Particle Filter (PF) are frequently applied. These filters integrate information from different origins, for example, the robot's dynamic model and diverse sensors (like IMU, encoders, LIDAR or camera) to obtain a more accurate and reliable estimation of the robot's pose [129].

The Kalman Filter (KF) can be described as an efficient recursive algorithm that estimates the dynamic state of a system based on noisy measurements. It makes use of two main operations: prediction, through a motion model, and correction, using sensor information. The system as-

sumes dynamics to be linear and noise to have Gaussian distribution. Constant mean squared error minimization makes continuous updating of the estimate available. This has been applied widely in the field of robotics to enhance localizing as well as pose estimation accuracy. However, this has also been applied in nonlinear systems whereby one has to apply tools such as the Extended Kalman Filter (EKF). [130].

The Particle Filter (PF) is a probabilistic method used for state estimation in nonlinear and non-Gaussian systems. It represents the robot's state using a set of weighted samples, known as particles. Each particle predicts the next state based on a motion model and subsequently updates its weight using sensor measurements. The posterior distribution is approximated through a resampling process, which concentrates computational effort on particles with higher likelihoods. This method is well-suited for handling complex noise characteristics and dynamic environments. Particle Filters are widely applied in mobile robot localization, particularly in the context of Monte Carlo Localization (MCL) [131].

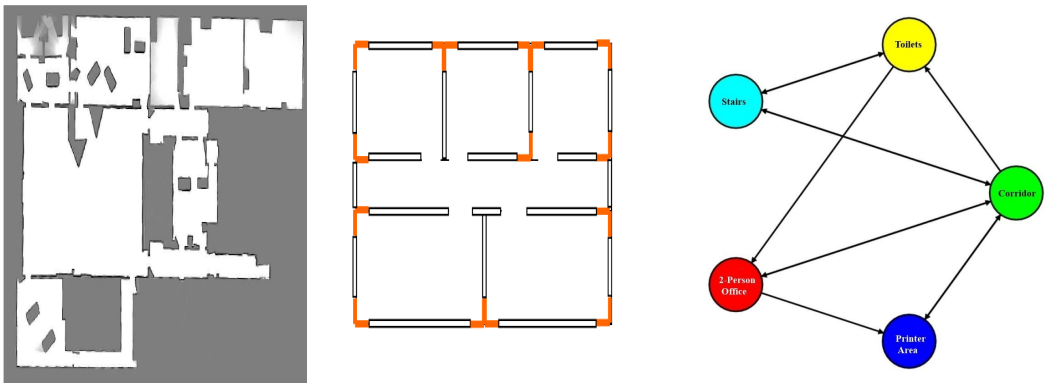
1.5 Mapping

In mobile robot navigation is the process of building a representation of the robot's surroundings from onboard sensors including LiDAR, cameras, or sonar. This representation, sometimes referred to as a , gives the robot spatial information about landmarks, free space, and obstacles so allowing it to localize itself and design effective paths. [132, 133].

The resulting map as depicted in Figure 1.10, it can take various forms depending on the application and sensor type, such as:

- **Occupancy Grid Map**: A 2D grid where each cell represents the probability that it is occupied.
- **Metric Map**: Provides precise geometric information about the environment.
- **Topological Map**: Represents the environment as a graph of connected places.
- **Semantic Map**: Includes object labels and higher-level understanding of the environment.

Mapping is a fundamental component of Simultaneous Localization and Mapping (SLAM), in which the robot builds the map while also estimating its own position within it [134].



(a) Occupancy Grid Map **(b)** Metric Map **(c)** Topological Map

Figure 1.10: Different types of mapping.

1.5.1 2D mapping based Laser scan data

2D mapping based on laser scan data involves using a 2D LiDAR (Light Detection and Ranging) sensor to acquire distance measurements from the robot to surrounding obstacles in a planar environment. As the robot moves, these distance readings are accumulated and transformed into a global reference frame using odometry or other localization techniques, enabling the construction of a two-dimensional occupancy grid map [135].

In this approach, each laser scan provides a set of range data $\{r_i, \theta_i\}$, where r_i is the distance to an obstacle and θ_i is the angle of the measurement. These polar coordinates are converted into Cartesian coordinates:

$$x_i = r_i \cos(\theta_i), \quad y_i = r_i \sin(\theta_i) \tag{1.10}$$

The robots pose p = $[x, y, \theta]^T$ is used to transform the local scan points into the global coordinate system. By integrating this information over time, the robot builds a 2D occupancy grid map $\mathscr{M}(x,y)$, where each cell in the grid holds a probability indicating whether the space is occupied, free, or unknown.

This method is widely used due to its simplicity and the accuracy of 2D LiDAR sensors. It forms the many SLAM algorithms, such as ROS packages: 2D GMapping, Hector SLAM, and Cartographer [135] .

3D mapping using RGB-D cameras involves capturing both color (RGB) images and depth (D) information from the environment. An RGB-D camera, such as the Microsoft Kinect, Intel RealSense, or ASUS Xtion, provides a dense depth map aligned with the RGB image, enabling the reconstruction of 3D environments in real time [136], the difference between 2D

and 3D mapping as shown in Figure 1.11.

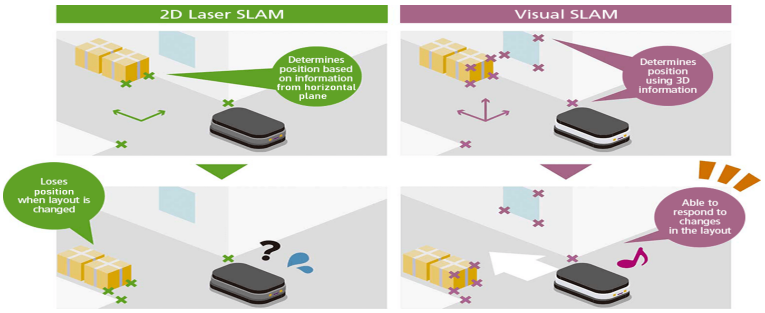


Figure 1.11: 2D and 3D Mapping.

Each pixel in the RGB image is associated with a depth value Z, allowing the recovery of 3D coordinates (X,Y,Z) of points in the scene using the pinhole camera model:

$$X = \frac{(u - c_x) \cdot Z}{f_x}, \quad Y = \frac{(v - c_y) \cdot Z}{f_y}, \quad Z = Z$$

The pixel coordinates are (u, v); the main point is (c_x, c_y); the focal lengths in the x and y directions respectively are f_x, f_y. Dense 3D map of the environment can be created by combining consecutive frames with accurate pose estimation (visual odometry or SLAM techniques). One can visualize this map as point clouds, voxel grids, truncated signed distance fields applied in volumetric mapping. In indoor robotics, 3D mapping with RGB-D cameras is especially helpful since it allows thorough scene awareness, object recognition, and obstacle avoidance.

1.6 Path Planning

1.6.1 Motion planning

Motion planning is a fundamental aspect of robotics and autonomous systems, involving the determination of a feasible and safe motion for a robot or vehicle. It broadly encompasses several sub-tasks, among which path planning and trajectory tracking are key components.

1.6.2 Path planning: Definition and types

Path planning is the process of determining a collision-free trajectory from a start position to a goal position within a known or unknown environments. The path planning module computes a series of waypoints or a

continuous trajectory that guides the robot to its destination while avoiding static and dynamic obstacles [95].

Path planning algorithms are generally can be classified into two categories: Global and local path planning. Global path planning that operates on a complete map (known) of the environment. Local path planning that reacts to local sensor data to avoid obstacles in real time. The path planning classification is illustrated in Figure 1.12

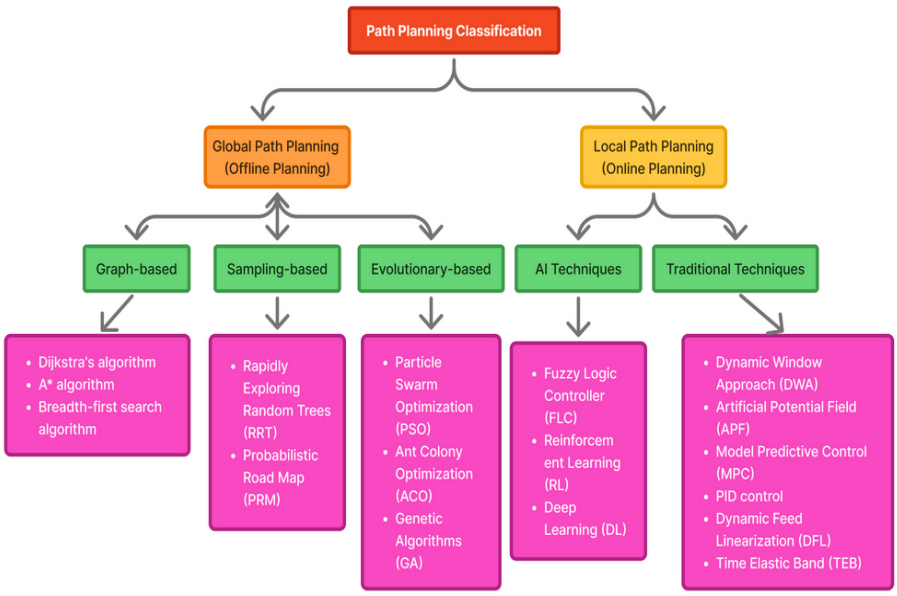


Figure 1.12: Path Planning Types.

1.6.3 Global path planning

Global path planning refers to the computational process of determining an optimal or feasible route for a robot to navigate from a given start location to a specified goal within a known or previously mapped environment. This approach assumes complete and static knowledge of the environment, typically represented as a stationary map, and does not account for dynamic obstacles or real-time sensor updates during execution [137].

Global path planners operate on either graph-based or continuous representations of the environment, aiming to minimize a cost function that typically includes path length, travel time, or energy consumption while avoiding obstacles. Common global path planning algorithms include Dijkstras Algorithm, which finds the shortest path through a uniform cost search strategy; A*, an informed search algorithm that employs a heuristic function to efficiently identify the shortest path; and Weighted A*, a variant of A* that accelerates computation by deliberately overestimating the

heuristic.

Other notable approaches include Rapidly-exploring Random Trees (RRT), a sampling-based method well-suited for high-dimensional continuous spaces, and Probabilistic Roadmaps (PRM), which randomly sample the configuration space and connect collision-free paths.

Population-based optimization techniques inspired by natural phenomena have also been applied to global path planning. Particle Swarm Optimization (PSO) mimics the social behavior of bird flocks or fish schools, where particles iteratively update their positions based on their own experience and that of their neighbors. Similarly, Ant Colony Optimization (ACO) draws inspiration from the foraging behavior of ants, using collective agent interactions to discover optimal paths.

Typically, global planners compute an initial trajectory based on the static environment, which is then passed to a local planner responsible for real-time path adjustments and obstacle avoidance in response to dynamic environmental changes.

1.6.4 Local path planning

Local path planning, also called online planning, is responsible for generating a safe and feasible trajectory in real time based on the robot's current position, velocity, and sensor readings. The local path planning in a limited region around the robot dynamically reacts to changes in the environment, such as the appearance of moving obstacles [42].

The local planner try to follow the global path as a reference and continuously adjusts the robots trajectory to follow it while avoiding collisions. It takes into account the robot's kinematic and dynamic constraints, real-time obstacle detection from onboard sensors such as LiDAR or an RGB-D camera, the goal direction and distance, safety margins, and cost maps.

Local path planning algorithms and approaches include the Dynamic Window Approach (DWA), which evaluates velocity command pairs (v,ω) within the robot's dynamic limits and selects the best trajectory based on multiple criteria such as heading to the goal, obstacle distance, and robot velocity. The Vector Field Histogram (VFH) constructs a polar histogram of obstacles and navigates through gaps in the obstacle field. Elastic Band and Timed-Elastic Band (TEB) methods optimise a flexible trajectory using physical-like forces or time-based constraints. The Artificial Potential Field (APF) approach guides the robot using attractive forces toward the goal and repulsive forces from obstacles.

Local path planning plays a crucial role in ensuring safe and adaptive navigation, especially in dynamic and partially known environments.

1.6.5 Hybrid path planning

Hybrid path planning combines the strengths of both global and local planning strategies to achieve efficient and safe robot navigation in dynamic environments. This approach addresses the limitations of each method when used independently for example, the lack of real-time reactivity in global planners and the short planning horizon in local planners [138].

In a hybrid algorithm, a global planner computes an initial, collision-free path using a complete or partial map of the environment. It provides high-level guidance from the start to the goal. Meanwhile, a local planner follows this path while continuously adjusting the robots trajectory in response to sensor data and dynamic obstacles. This layered architecture allows the robot to: avoid static and dynamic obstacles in real time; respect the robots motion constraints; react to unexpected environmental changes; and ensure progress toward the goal even in partially unknown or cluttered environments. Popular hybrid systems include combinations such as A* or Dijkstra + DWA (Dynamic Window Approach), PRM or RRT + TEB (Timed Elastic Band), and Global APF + Local fuzzy control or MPC.

Hybrid path planning is widely adopted in mobile robotics applications such as autonomous navigation in indoor environments, search and rescue, and service robots operating in human-populated areas.

1.7 Trajectory tracking in mobile robots

Trajectory tracking refers to control strategies that enable a mobile robot to follow a predefined path or trajectory over time with high accuracy. These methods aim to reduce both the pose error by continuously adjusting the robot's control inputs. Ensure that the robot not only reaches its target but also follows the exact shape and timing of the desired path [139]. These approaches are critical for real-time navigation tasks in environments where precision and safety are important, such as autonomous driving, industrial automation, and search-and-rescue missions.

The famous techniques are PID control, feedback linearization, backstepping, sliding mode control, model predictive control (MPC), and fuzzy

logic -based controllers. In other hand it's exist advanced approaches such as machine learning in dynamic or unceratin envirenemtns. The main goal is stability, robustness, and smooth control while reacting to disturbances and obstacle avoidance.

1.7.1 Trajectory tracking based on MPC control

Trajectory tracking based on Model Predictive Control (MPC) is an advanced control strategy used to guide a mobile robot along a predefinded path while respecting system constraints [140]. MPC predicts the future behavior of the robot over a finite time horizon by solving an optimization problem at each control step . The controller selects control actions that minimize a cost function , typically based on tracking errors and control effort . MPC takes into account the robot's kinematic or dynamic model, envirenments constraints

MPC takes into account the robots kinematic or dynamic model, environmental constraints (like obstacles), and input limitations (e.g., maximum velocity or acceleration). Only the first control action from the optimal sequence is applied at each step, and the process is repeated at the next time step with updated state information [141].

This approach offers several advantages such as handling multivariable systems, incorporating constraints naturally, and producing smooth, stable trajectories. In dynamic or cluttered settings where real-time adaptability and safety are paramount, MPC is especially successful. Because of its predictive and constraint-aware character, trajectory tracking with MPC is extensively applied in autonomous cars, mobile robots [142], and UAVs. The general MPC Cost Function for Trajectory Tracking is defined as follows:

$$J(k) = \sum_{j=1}^{N} \underbrace{\|\mathbf{x}(k+j|k) - \mathbf{x}_{\text{ref}}(k+j)\|^2_{\mathbf{Q}}}_{\text{State tracking error}} + \sum_{j=0}^{N-1} \underbrace{\|\mathbf{u}(k+j|k) - \mathbf{u}_{\text{ref}}(k+j)\|^2_{\mathbf{R}}}_{\text{Control effort penalty}} \tag{1.11}$$

where:

- $J(k)$: Cost function value at discrete time step k.
- N: Prediction horizon length.
- $\mathbf{x}(k+j|k)$: Predicted state vector at time $k+j$ based on information available at time k.

- $\mathbf{x}_{\text{ref}}(k+j)$: Reference (desired) state at time $k+j$.
- $\mathbf{u}(k+j|k)$: Predicted control input vector at time $k+j$ based on information available at time k.
- $\mathbf{u}_{\text{ref}}(k+j)$: Reference (desired) control input at time $k+j$ (often zero).
- $\mathbf{Q}$: Positive semi-definite weighting matrix penalizing state tracking error.
- $\mathbf{R}$: Positive definite weighting matrix penalizing control effort.

1.7.2 Trajectory tracking based on PID control

A classic method used to steer a mobile robot along a desired path by minimizing tracking errors is trajectory tracking based on PID (Proportional-Integral-Derivative) control. Based on the current error (proportional term), the accumulation of past errors (integral term), and the rate of change of the error (derivative term) the PID controller computes control inputs [143]. To minimize the distance and orientation error between the robot's actual and reference positions, this control method continuously changes the robot's linear and angular velocities. Simple to use, it requires little model knowledge, and is extensively applied in robotics and automation.

Proportional-Integral-Derivative (PID) control is widely recognized for its effectiveness in mobile robot control due to its simplicity and ease of implementation. However, its performance tends to degrade when dealing with under-actuated robots or highly nonlinear systems, especially in dynamic and unstructured environments. Achieving an optimal trade-off between speed, accuracy, and stability critically depends on the proper tuning of the three controller gains (K_p, K_i, and K_d). While PID controllers can offer satisfactory tracking performance in structured or well-modeled scenarios, their effectiveness diminishes in the presence of complex dynamics or environmental uncertainties.

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau)\, d\tau + K_d \cdot \frac{de(t)}{dt} \tag{1.12}$$

1.7.3 Trajectory tracking based on LQR control

Trajectory tracking using Linear Quadratic Regulator (LQR) control involves designing a feedback controller that steers a system along a desired

trajectory while minimizing a cost function that considers both state deviations and control effort [144]. LQR is particularly effective for linear, time-invariant systems and provides an optimal control solution based on a quadratic cost function.

In control theory, a well-known method is trajectory tracking grounded on Linear Quadratic Regulator (LQR) control. LQR is meant to minimize a quadratic cost function while guaranteeing stability of the system. Here is a thorough analysis of how LQR might be used to track trajectories in mobile robots [145]. The robot's motion model is typically represented as a linear state-space system:

$$\dot{x}(t) = A \cdot x(t) + B \cdot u(t) \tag{1.13}$$

where:

- $x(t)$ is the state vector, representing the robots position, velocity, and orientation,
- $u(t)$ is the control input vector (e.g., linear and angular velocities),
- A is the system dynamics matrix, and B is the input matrix.

The objective of LQR is to minimize the following quadratic cost function given by:

$$J = \int_0^\infty \left[\mathbf{x}^T \mathbf{Q}\mathbf{x} + \mathbf{u}^T \mathbf{R}\mathbf{u}\right] dt, \quad \mathbf{Q} = \mathbf{Q}^T \succeq 0,\ \mathbf{R} = \mathbf{R}^T \succ 0. \tag{1.14}$$

The optimal cost function $J^*(\mathbf{x})$

$$\forall \mathbf{x}, \quad 0 = \min_{\mathbf{u}} \left[\mathbf{x}^T \mathbf{Q}\mathbf{x} + \mathbf{u}^T \mathbf{R}\mathbf{u} + \frac{\partial J^*}{\partial \mathbf{x}}(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u})\right]. \tag{1.15}$$

where:

- Q is a positive-definite weighting matrix that penalizes deviations from the desired state (i.e., the tracking error),
- R is a positive-definite weighting matrix that penalizes excessive control inputs (i.e., energy consumption).

The optimal control law that minimizes the cost function is given by the state feedback law:

$$u(t) = -Kx(t) \tag{1.16}$$

where K is the state-feedback gain matrix. The matrix K is computed by solving the continuous algebraic Riccati equation (CARE):

$$A^T P + PA - PBR^{-1}B^T P + Q = 0 \tag{1.17}$$

The matrix P is the solution to the CARE, and it is used to compute the optimal gain matrix:

$$K = R^{-1}B^T P \tag{1.18}$$

1.7.4 Trajectory tracking using sliding mode control

For trajectory tracking of a wheeled mobile robot (WMR) using Sliding Mode Control (SMC), the essential equations typically include the error dynamics, sliding surface definition, and the control law that ensures robust tracking despite uncertainties and disturbances [146]. Consider a wheeled mobile robot (WMR) with state vector $\mathbf{x} = [x, y, \theta]$

where: (x, y) represent the planar position and θ the orientation. The kinematic model is given by

$$\begin{cases} \dot{x} = v\cos\theta, \\ \dot{y} = v\sin\theta, \\ \dot{\theta} = \omega, \end{cases} \tag{1.19}$$

where v and ω are the linear and angular velocities, respectively.

The tracking error expressed in the robots local frame is

$$\begin{cases} e_1 = \cos\theta(x_r - x) + \sin\theta(y_r - y), \\ e_2 = -\sin\theta(x_r - x) + \cos\theta(y_r - y), \\ e_3 = \theta_r - \theta, \end{cases} \tag{1.20}$$

where (x_r, y_r, θ_r) is the reference trajectory. To induce first-order sliding behavior (ensuring that both error and its rate converge), construct the surface:

$$s = \dot{e} + \Lambda e \tag{1.21}$$

where

$$s = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} \dot{e}_1 + \lambda_1 e_1 \\ \dot{e}_2 + \lambda_2 e_2 \\ \dot{e}_3 + \lambda_3 e_3 \end{bmatrix} \tag{1.22}$$

and

$$\Lambda = \text{diag}(\lambda_1, \lambda_2, \lambda_3) \tag{1.23}$$

with positive constants λ_i.

The control law that will force the sliding surface to zero:

$$\begin{cases} v = v_r \cos e_3 + k_1 \,\text{sat}\left(\frac{s_1}{\phi_1}\right), \\ \omega = \omega_r + k_2 \,\text{sat}\left(\frac{s_2}{\phi_2}\right) + k_3 \,\text{sat}\left(\frac{s_3}{\phi_3}\right), \end{cases} \tag{1.24}$$

where $k_i > 0$ are control gains, $\phi_i > 0$ define boundary layer thickness to reduce chattering, and $\text{sat}(\cdot)$ is the saturation function. Consider the Lyapunov function

$$V = \frac{1}{2} s^T s, \tag{1.25}$$

which satisfies

$$\dot{V} \leq -\eta \|s\|, \quad \eta > 0, \tag{1.26}$$

ensuring finite-time convergence of the sliding surface $s \to 0$ and thus tracking errors converge. The related Control Approaches of SMC are list as follows:

- **Dynamic Feedback Linearization (DFL):** Transforms nonlinear dynamics into linear form for trajectory tracking with guaranteed stability.
- **Model Predictive Control (MPC):** Optimizes a cost function balancing tracking error and control effort over a prediction horizon.
- **Linear Quadratic Regulator (LQR):** Minimizes a quadratic cost on state and control deviations, often applied to linearized error dynamics.

1.7.5 Motion control

Motion control is the crucial task for autonomous navigation that is responsible to excute the optimal path planned from diffrents algorithms and approaches by generating control commands as velocitiy (linear and angular velocities) to drive the robot's actuators This task ensures that the robot follows and track the desired path while preserve stability and satisfyning its kinematic and dynamic contraints [7].

Various approaches, algorithms, and techniques have been developed to address the problem of motion control in mobile robotics. AI-based, such

as reinforcement learning and deep learning, and fuzzy logic controller , classical appraoches such as model predictive control and linear quadratic regularization. Aditionly advanced techniques such as Dynamic feedback linearization and sliding control are used to addess nonlinearity. Path planning and motion control are the core of an autonomous navigation system, enabling the robot to move safely and efficiently in its environment.

1.8 Reasoning and Decision making

Intelligent systems including robots interpret and act upon the environment by means of fundamental cognitive processes called reasoning and decision-making. By means of available information, the processes enable robots to make informed decisions, so guaranteeing effective navigation, task completion, and interaction with dynamic environments

1.8.1 Reasoning

The cognitive process by which an intelligent system that the capabilty to draws conclusions from a collection of observations is known as reasoning. Robotics uses a variety of reasoning techniques. Deductive reasoning is the process of inferring particular conclusions from general premises. For example, a robot that is familiar with its surroundings and navigational rules can infer the best action. Conversely, inductive reasoning enables the robot to draw generalizations from specific observations. For instance, it can deduce more general patterns of obstacle distribution by observing obstacles in a localized area. When a robot encounters unexpected events or has limited data, abductive reasoning is especially useful because it can be used to generate the most plausible explanation for incomplete or uncertain information [148].

1.8.2 Decision making

Decision making refers to the process by which an intelligent agent selects an action from a set of available options, with the goal of achieving specific objectives . Several decision-making approaches are commonly employed in robotics [149].

In classical decision theory, which stems from utility theory, the goal is to choose options that optimize expected utility. The robot looks at various actions it could take and chooses the one that seems most likely to yield the

best results, considering where it currently is and what it aims to achieve. In uncertain decision-making, Markov Decision Processes (MDPs) serve as mathematical frameworks that represent the robot's environment as a series of states, with actions prompting shifts from one state to another. The aim here is to enhance the total expected rewards we can gather over time. With reinforcement learning (RL), robots can figure out the best ways to act by engaging with their surroundings and getting feedback that comes in the form of rewards or penalties. The robot learns by trying things out and figuring out what works best for making decisions. When faced with vague or unclear information, fuzzy logic comes into play, helping make decisions by relying on language-based variables and specific rules. For example, a robot could apply fuzzy logic to figure out a safe speed, taking into account how close obstacles are and the surrounding environment. Decision making includes the need to reflect on future actions, often requiring careful planning and optimization. Algorithms like A* or Rapidly-exploring Random Trees (RRT) help in mapping out the steps needed to hit a target, whereas optimization algorithms tweak these plans to meet specific goals, like conserving energy or cutting down on time.

1.8.3 Integration of reasoning and decision making

Integration of Reasoning and Decision Making involves combining various reasoning methods and decision-theoretic models to enable intelligent, context-aware, and robust decision processes. This integration supports handling uncertainty, learning from experience, and optimizing actions in complex environments. It is applied in areas such as industrial control, AI systems, and decision support, improving adaptability, efficiency, and ethical decision-making.

1.8.4 Reasoning and decision-making: Applications and challenges

Reasoning and decision-making are essential components of numerous robotic applications, particularly in autonomous navigation, enabling robots to assess their surroundings and select path that facilitate efficient movement while circumventing obstacles. In task planning, robots ascertain the sequence of actions required to accomplish certain objectives, such as item retrieval or assembly tasks. In multi-robot coordination, decision-making is crucial for synchronizing the actions of several robots to attain common goals or to mitigate potential conflicts. In human-robot inter-

action, robots depend on reasoning on human behavior and intentions to make judgments that facilitate safe and productive collaboration.

Numerous challenges in the field of robotics, regardless of advancements in decision-making and reasoning methodologies. Uncertainty is a significant obstacle. Reasoning and decision-making systems must be capable of functioning with incomplete or chaotic data, as real-world environments are frequently unpredictable. Another obstacle is complexity, as machines may be required to make decisions based on complex environmental models, which can result in substantial computational demands. Furthermore, real-time constraints are essential in numerous robotics applications, necessitating the rapid and efficient execution of decisions, such as during obstacle avoidance or navigation. Finally, ethical considerations must be addressed, particularly in scenarios where robots are compelled to make decisions that affect human safety or interact with humans.

1.8.5 Fuzzy Logic Controller

A fuzzy logic controller (FLC) is a control system that applies fuzzy logic principles to make decisions. Fuzzy logic, unlike traditional Boolean logic, allows for reasoning with imprecise, ambiguous, or uncertain information. In a typical FLC, the control action is decided by a group of fuzzy rules that use everyday language terms (like "high," "low," or "medium") to explain the system's conditions. [160]. The FLC typically consists of four main components: fuzzification, rule base, inference engine, and defuzzification.

Fuzzification is the method of generating fuzzy sets from crisp input values including error values. This is achieved in part by membership functions mapping the crisp input into degrees of membership within specified fuzzy sets. Should the input be an error term (e), for example the fuzzification process assigns it a degree of membership in fuzzy sets including "negative," "zero," and "positive." The degree of membership determines the intensity of the link between the input and any set. The rule base consists of a set of if-then rules defining the behavior of the system. Usually developed by human experts, these rules mirror in language terms the connection between inputs and outputs. If the error is "positive" and the change in error is "negative," for instance, the output is "decrease," meaning, in the case of a mobile robot, lower speed or alter the motor. If the error is "zero" and it changes to be "zero," the output is "maintain," or so preserve the existing position or state. Every rule evaluates a specific link

between the outputcontrol actionand a given relationship between inputserror and change in error. The component of the controller handling the fuzzy input values and using the fuzzy rules to produce fuzzy output values is the Inference Engine. Combining the guidelines, it computes the outcome depending on the input values. Usually including Mamdani Inference, a widely used fuzzy inference technique, the inference process uses fuzzy operators to combine input fuzzy sets so determining the output fuzzy sets. Often in optimization issues, another form of fuzzy inference is Sugeno Inference, in which the output is a crisp function of the input values. The Defuzzification procedure turns fuzzy output values created by the inference engine into clear control actions. The most often used technique computes the center of gravity of the fuzzy output set by use of the centroid approach.

In robotics, FLCs are extensively applied in tasks including speed control (adjusting a robot's speed depending on sensor readings), position control (keeping a robot in a desired position by means of velocity and direction), and obstacle avoidance (using fuzzy logic to control a robot's motions in response to obstacles in the environment).

Among their several advantages are their simplicity in which one may adjust or tweak them to match new conditions and their ability to control flawed sensor data. Among the main advantages are simplicity (especially for complex, nonlinear systems where exact models are not available), resilience (FLCs are robust to uncertainty and noise in the system), and intuitive design (fuzzy rules are based on human knowledge and intuition, thus they are accessible and interpretable).

On the other hand, FLCs have various disadvantages including computational complexity (as the number of rules raises the computational needs for the FLC also expand) and rule base design (creating the FLC takes expert knowledge, which may not always be available or may need considerable work).

1.8.6 Reinforcement Learning

Reinforcement learning (RL) is a class of machine learning techniques by which an agent learns to generate sequential decisions by means of interactions with an environment. The main goal of the RL agent is to choose best actions depending on the states it comes across so maximising cumulative rewards over time. Based on the consequences of actions without explicit supervision, RL differs from supervised learning in that correct

outputs are not given. [5].

The RL framework is composed of several core components. The agent is the entity that learns and makes decisions. The environment is the external system with which the agent interacts, providing feedback in the form of rewards or penalties. The state represents the current situation or configuration of the environment as perceived by the agent. An action is a decision taken by the agent, influencing the environment and causing a transition to a new state. The reward is a scalar feedback signal received after taking an action, indicating the desirability of the resulting state. The policy is a strategy or mapping from states to actions, guiding the agent's behavior. The value function estimates the expected cumulative reward from a given state, reflecting how beneficial it is for the agent to be in that state. The Q-function, or action-value function, evaluates the expected return of taking a specific action in a specific state and then following the optimal policy. The discount factor is a parameter ranging from 0 to 1 that balances the importance of immediate versus future rewards.

The learning process in RL generally follows these steps: the agent starts in an initial state, selects an action based on its current policy, receives a reward and transitions to a new state, and then uses this feedback to improve its policy. This process continues iteratively. The agent's goal is to maximize the cumulative discounted reward, known as the return, which is defined as the sum of rewards over time discounted by a factor γ and it defined as follows::

$$Return = \sum_{t=0}^{\infty} \gamma^t r_t \tag{1.27}$$

Reinforcement Learning methods can be categorized into several types. Model-free RL methods learn optimal policies directly from interactions with the environment without building a model of it. Examples include Q-learning and Policy Gradient methods. Model-based RL approaches involve constructing a model of the environments dynamics and using it for planning, such as in Dyna-Q. In on-policy RL, the agent learns the value of the policy it is currently using, as in SARSA. In contrast, off-policy RL methods, like Q-learning, learn the value of a different policy than the one used for exploration.

Some widely used RL algorithms include Q-learning, which is a model-free and off-policy method based on the Bellman equation:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)] \tag{1.28}$$

where α is the learning rate and γ is the discount factor. Deep Q-Networks (DQN) extend Q-learning by using neural networks to approximate the Q-values. Policy Gradient methods directly optimize the policy by adjusting it in the direction that increases the expected reward. Actor-Critic methods combine both policy-based and value-based approaches, with the actor updating the policy and the critic evaluating the action values. Reinforcement Learning is applied in various real-world domains such as robotics for motion planning and control, game playing for mastering complex games like Go or chess, autonomous vehicles for navigation and control, and recommendation systems for personalizing content delivery. Among its advantages, RL offers flexibility in learning optimal policies across diverse tasks, does not require labeled data, and is particularly well-suited for sequential decision-making. However, RL also faces challenges, including high sample inefficiency, the need to balance exploration and exploitation, and difficulties in handling delayed rewards.

1.8.7 Deep Learning

Deep Learning (DL) is a subset of machine learning that utilizes algorithms inspired by the structure and function of the human brain, known as artificial neural networks, as shown in Figure 1.13. These models, composed of multiple layers of interconnected processing units, are capable of learning from vast amounts of data by identifying complex patterns and hierarchical representations. Deep learning has shown remarkable success in outperforming traditional machine learning techniques in a wide range of applications, including image recognition, natural language processing, and game playing [151].

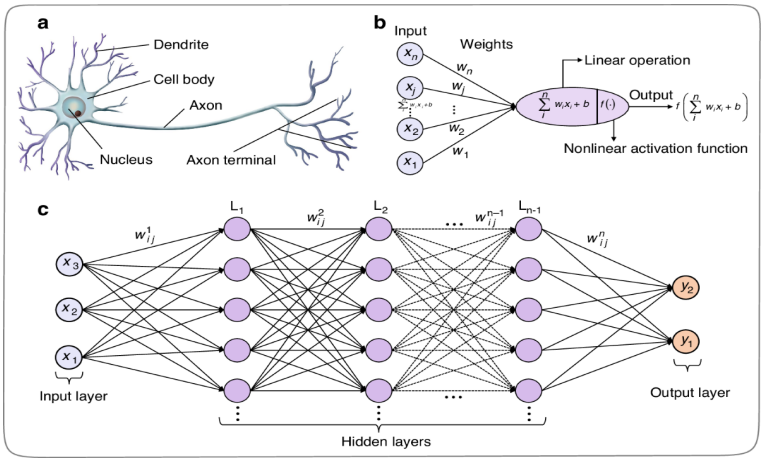


Figure 1.13: Biological and Artificial Neural Networks.

A typical deep learning model includes several fundamental components. At the core are neural networks, which consist of layers of neurons. Each neuron processes input data using a mathematical function and transmits the result to the next layer. These layers include an input layer that receives the raw data, hidden layers that transform the input through a series of computations to extract features, and an output layer that provides the final prediction. Activation functions are applied at each neuron to introduce non-linearity into the model, enabling it to learn complex patterns. Common activation functions include ReLU (Rectified Linear Unit), Sigmoid, and Tanh. The network's parameters consist of weights and biases, which are adjusted during training to improve performance. Weights represent the strength of connections between neurons, while biases allow flexibility in the activation.

The models performance is evaluated using a loss function, which quantifies the difference between predicted outputs and the actual values. This loss is minimized using optimization algorithms such as Stochastic Gradient Descent (SGD), Adam, or RMSprop, which iteratively update the model parameters.

There are several types of deep learning architectures, each suited to specific tasks. Feedforward Neural Networks (FNNs) are the simplest, where data flows in one direction from input to output. Convolutional Neural Networks (CNNs) are widely used in image processing due to their ability to capture spatial hierarchies. Recurrent Neural Networks (RNNs) and their variant Long Short-Term Memory (LSTM) networks are effective for sequential data like text or time series. Generative Adversarial Networks (GANs) consist of two networks that compete to generate realistic data, while Autoencoders are used for unsupervised learning and data compression. The training process in deep learning involves several stages. First, in forward propagation, input data passes through the network to produce an output. Next, the loss is computed by comparing the output with the actual target. Backpropagation then calculates how much each parameter contributed to the error, allowing the gradients to be computed. These gradients are used by the optimizer to update the weights and biases in a way that reduces the loss. This process is repeated over multiple epochs until the model achieves satisfactory performance.

Deep learning has been successfully applied in various domains such as computer vision, natural language processing, speech recognition, autonomous vehicles, healthcare, and gaming. It offers several advantages, including high accuracy, the ability to perform end-to-end learning, and

scalability to large datasets. However, it also presents certain challenges. Deep learning models typically require large datasets and significant computational resources. Additionally, they are often considered black-box models, making them difficult to interpret. There is also the risk of overfitting, where the model performs well on training data but poorly on unseen data.

To support the development and deployment of deep learning models, several powerful frameworks have been developed. TensorFlow, PyTorch, Keras, Caffe, and MXNet provide tools and libraries that simplify the process of building, training, and evaluating neural networks.

1.9 Conclusion

This chapter has provided a detailed survey of the fundamental components of autonomous mobile robot navigation. Beginning with an overview of mobile robot navigation principles, it explored key perception techniques essential for environment understanding. Robust localization and mapping methods were discussed as critical enablers for reliable navigation in unknown or dynamic environments. The chapter then examined path planning strategies that generate feasible and optimal routes, followed by motion control and trajectory tracking approaches that ensure precise execution of planned paths. Finally, the integration of reasoning and decision-making frameworks was highlighted as vital for adaptive and intelligent navigation in complex scenarios. Together, these interconnected modules form the foundation of modern autonomous navigation systems. Continued research and development in these areas, especially in advanced control and decision-making, will drive further improvements in robot autonomy, safety, and operational efficiency. In the next chapter, we present the fundamental concepts of fuzzy logic control and feedback linearization control, focusing on their application to mobile robot trajectory tracking and obstacle avoidance behaviors.

2

Fuzzy Logic and Dynamic Feedback Linearization Control for Mobile Robots Navigation

This chapter presents the theoretical foundations of trajectory tracking and obstacle avoidance for mobile robots using a hybrid control strategy. It explores Dynamic Feedback Linearization for precise path tracking and Fuzzy Logic Control for handling uncertainty and unknown environments. The integration of both methods is analyzed to ensure robust and intelligent autonomous navigation.

Contents

2.1 Introduction

Ensuring accurate trajectory tracking while avoiding obstacles is an essential task for autonomous mobile robots. Dynamic Feedback Linearization (DFL) effectively handles nonlinear robot dynamics, but its performance may degrade in uncertain environments. To improve adaptability and robustness, fuzzy logic is integrated with DFL, allowing the controller to adjust in real time based on environmental changes. This chapter presents a hybrid control approach combining fuzzy logic and DFL for reliable and intelligent navigation.

2.2 Modeling of differential drive mobile robots

Differential-drive mobile robots are one of the most common types of wheeled robots used in research and industry due to their simple mechanical structure and efficient maneuverability. They consist of two independently driven wheels placed on either side of the robot, allowing it to move forward, backward, and turn by varying the velocities of the wheels. A castor wheel or ball support is typically used for balance [152].

2.2.1 State-space representation

State-space representation is a mathematical framework used to model and analyze dynamic systems by describing them with a set of first-order differential (or difference) equations. Instead of focusing on input-output relations directly, the state-space approach models the system's internal behavior through a collection of variables called state variables.

A general continuous-time linear time-invariant (LTI) system can be expressed in the following form:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \tag{2.1}$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t), \tag{2.2}$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ is the state vector, $\mathbf{u}(t) \in \mathbb{R}^m$ is the input vector, and $\mathbf{y}(t) \in \mathbb{R}^p$ is the output vector. The matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ is the system matrix defining the dynamics of the state, $\mathbf{B} \in \mathbb{R}^{n\times m}$ is the input matrix, $\mathbf{C} \in \mathbb{R}^{p\times n}$ is the output matrix, and $\mathbf{D} \in \mathbb{R}^{p\times m}$ is the feedthrough or direct transmission matrix. The state-space representation provides a compact and flexible way to model multi-input multi-output (MIMO) systems and can easily

accommodate complex systems with multiple interacting components. It is also suitable for modern control design techniques, such as pole placement, optimal control (e.g., LQR), and observer design (e.g., Luenberger observers and Kalman filters). In the discrete-time case, where the system is sampled at discrete time intervals, the state-space equations become

$$\mathbf{x}[k+1] = \mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k], \tag{2.3}$$

$$\mathbf{y}[k] = \mathbf{C}\mathbf{x}[k] + \mathbf{D}\mathbf{u}[k], \tag{2.4}$$

where k denotes the discrete time step.

The main advantages of the state-space approach are its ability to handle multiple states and inputs simultaneously, to represent systems of any order in a unified manner, and to extend naturally to nonlinear systems by generalizing the equations:

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)), \tag{2.5}$$

$$\mathbf{y}(t) = g(\mathbf{x}(t), \mathbf{u}(t)), \tag{2.6}$$

where $f(\cdot)$ and $g(\cdot)$ are nonlinear functions. Overall, the state-space formulation is a fundamental tool in modern system analysis and control theory, providing a systematic method for modeling, simulation, and design.

2.2.2 Kinematic model and non-holonomic constraints

Figure 2.1 illustrates a simplified schematic representation of a differential-drive mobile robot, where (OX_gY_g), represents a global inertial reference, and (OX_lY_l) is a local coordinates frame attached to the robot. The two wheels with a common radius denoted by r are separated by a distance L.

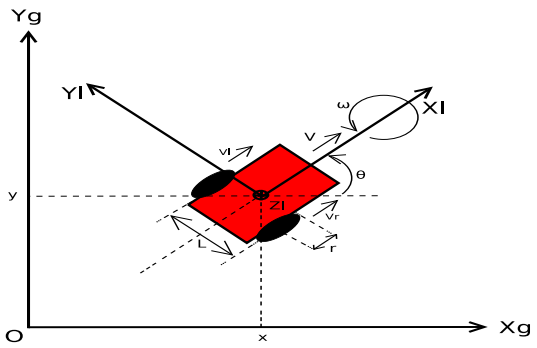


Figure 2.1: Unicycle-type differential drive robot.

The robot's pose in the global frame is defined by the position (x,y) in the plane and the orientation angle θ with respect to the $OX-axis$ of the inertial frame. The complete state of the robot can be represented by the vector $\mathbf{q} = [x \;\; y \;\; \theta]^T$, where the superscript T indicates the transpose. Let the linear velocities of the left and right wheels be represented by v_L and v_R, respectively. and angular velocities of the left and right wheels be represented by $\dot{\phi}_L$ and $\dot{\phi}_R$, respectively. From these, the robot's forward linear velocity v and angular velocity ω can be computed as follows [63]:

$$\begin{cases} v = \dfrac{v_R + v_L}{2} = \dfrac{R}{2}(\dot{\phi}_R + \dot{\phi}_L), \\ \omega = \dfrac{v_R - v_L}{L} = \dfrac{R}{2L}(\dot{\phi}_R - \dot{\phi}_L). \end{cases} \tag{2.7}$$

Based on these values, the kinematic model of the robot, which governs its motion in the plane, is defined as follows:

$$\begin{cases} \dot{x} = v\cos\theta = \dfrac{v_R + v_L}{2}\cos\theta, \\ \dot{y} = v\sin\theta = \dfrac{v_R + v_L}{2}\sin\theta, \\ \dot{\theta} = \omega = \dfrac{v_R - v_L}{L}. \end{cases} \tag{2.8}$$

These equations describe the motion of a differential drive robot in a two-dimensional space and serve as a foundation for both simulation and control strategies.

Additionally, to ensure realistic motion without slippage, the robot must adhere to non-holonomic constraints. These constraints are as follows:

$$\begin{cases} \dot{y}\cos\theta - \dot{x}\sin\theta = 0, \\ \dot{x}\cos\theta + \dot{y}\sin\theta + \dfrac{L}{2}\dot{\theta} - v_R = 0, \\ \dot{x}\cos\theta + \dot{y}\sin\theta - \dfrac{L}{2}\dot{\theta} - v_L = 0. \end{cases} \tag{2.9}$$

This constraints ensures that the robot does not move sideways (perpendicular to its main direction of travel), while the other two express the no-slip condition for each wheel.

2.2.3 Dynamic model of DWMR

Differential wheeled mobile robots (DWMRs) are a class of non-holonomic systems characterized by two independently driven wheels and typically

one or more passive caster wheels for balance. The dynamic modeling of such robots is essential for control design, simulation, and real-world implementation, particularly in tasks involving trajectory tracking and obstacle avoidance. We consider the robot moving on a flat horizontal plane with the following parameters:

- m: mass of the robot body
- r: radius of the wheels
- L: half the distance between the two drive wheels (i.e., wheelbase is $2L$)
- I_z: moment of inertia around the vertical axis (yaw)
- τ_R, τ_L: torques applied to the right and left wheels, respectively

The robots configuration is described by the pose vector: $\mathbf{q} = \begin{bmatrix} x & y & \theta \end{bmatrix}^T$ where:

- x, y: position of the robot in the global frame
- θ: orientation (heading angle) of the robot

The velocities of the robot are:

- v: linear velocity of the robot's center
- ω: angular velocity about the vertical axis

Using NewtonEuler or Lagrangian mechanics, the dynamic equations of motion for a DWMR can be expressed as:

$$\begin{cases} m\dot{v} = \dfrac{1}{r}(\tau_R + \tau_L), \\ I_z\dot{\omega} = \dfrac{L}{r}(\tau_R - \tau_L) \end{cases} \tag{2.10}$$

The robot's position and orientation evolve according to:

$$\begin{cases} \dot{x} = v\cos(\theta) \\ \dot{y} = v\sin(\theta) \\ \dot{\theta} = \omega \end{cases} \tag{2.11}$$

Combining the kinematic and dynamic equations, we obtain the state-space model:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ \omega \\ \frac{1}{mr}(\tau_R + \tau_L) \\ \frac{L}{I_z r}(\tau_R - \tau_L) \end{bmatrix} \tag{2.12}$$

This nonlinear model captures the essential dynamic behavior of a differential drive robot and serves as the foundation for control design techniques that consider physical properties and constraints, such as Model Predictive Control (MPC), Feedback Linearization, and Fuzzy Logic Control. Dynamic modeling is crucial for developing robust control strategies that address actuator limitations, disturbances, and system uncertainties. By considering the effects of mass, inertia, and applied torque, it enables accurate prediction of the robots behavior and facilitates the design of controllers for stable and precise motion.

2.2.4 Motion control of mobile robot systems

In mobile robotics, motion control plays a fundamental role in ensuring that a robot moves safely, efficiently, and accurately to achieve its navigation and interaction tasks. The main objectives of motion control in mobile robots are closely related to their autonomous behavior in dynamic and uncertain environments. A primary objective is trajectory tracking, where the robot must follow a desired path with precision while maintaining appropriate velocities and orientations. Trajectory tracking is critical for mobile robots navigating through structured environments, such as corridors, warehouses, or outdoor pathways. Another important goal is position regulation, where the mobile robot must reach a specified target point, minimizing the error in both position and orientation. Position regulation is essential for docking maneuvers, waypoint navigation, and target reaching tasks. Obstacle avoidance is a vital objective in mobile robotics, ensuring that the robot can detect and safely circumvent static and dynamic obstacles in its environment. Effective obstacle avoidance strategies require integrating motion planning with real-time control adjustments. Stability and robustness are crucial to maintain reliable robot behavior in the presence of modeling uncertainties, sensor noise, and external disturbances.

Mobile robots often operate in unpredictable conditions, making it necessary for the motion controller to ensure stable trajectories and recover from disturbances. Energy efficiency is another significant objective, especially for battery-operated mobile robots. Motion control algorithms must optimize trajectories and velocity profiles to reduce energy consumption, thereby extending operational autonomy. Finally, safety must be guaranteed, particularly when robots share environments with humans. Safe motion control ensures that robots react appropriately to unexpected obstacles or changes in the environment, maintaining human-robot coexistence without incidents.

In summary, the main motion control objectives in mobile robotics include trajectory tracking, position regulation, obstacle avoidance, stability, robustness, energy efficiency, and safety. Achieving these objectives requires the integration of kinematic and dynamic models, real-time perception, adaptive control methods, and intelligent decision-making strategies.

2.2.5 Common issues in path tracking and collision avoidance

In mobile robotics, path tracking and collision avoidance are two critical tasks that ensure safe and efficient navigation. However, several common issues arise in their implementation, often affecting the robot's performance and reliability. One major issue is tracking errors, which occur when the robot deviates from the desired trajectory due to model inaccuracies, actuator limitations, or external disturbances. Precise modeling of the robot's kinematics and dynamics, as well as robust control design, are necessary to minimize tracking errors.

Dynamic obstacles represent another challenge. Mobile robots must adapt their path in real time to avoid moving obstacles such as pedestrians or other robots. Predicting the behavior of dynamic obstacles and planning safe avoidance maneuvers remains a complex problem, especially in cluttered or unpredictable environments. Latency and sensor noise can degrade both path tracking and collision avoidance performance. Delays in sensor measurements or control signals, along with inaccuracies in perception data, can cause the robot to react late or incorrectly to environmental changes, leading to collisions or unstable motion. Limited field of view and occlusions in the sensing system can prevent the robot from detecting obstacles in time. This issue is particularly critical in indoor environments with tight spaces or outdoor environments with complex structures.

Nonholonomic constraints also pose a difficulty for path tracking. Differential-

drive or car-like robots cannot move directly sideways and must plan feasible trajectories that respect their motion limitations, which complicates accurate following of arbitrary paths. Local minima are common in potential field-based or reactive obstacle avoidance strategies, where the robot can become trapped in areas with no clear path toward the goal. Overcoming local minima often requires global planning strategies or hybrid methods that combine local and global information. Trade-offs between safety and optimality must be carefully managed. Conservative avoidance maneuvers ensure safety but can lead to inefficient paths and longer travel times. Conversely, aggressive strategies can optimize speed but increase collision risk.

In summary, common issues in path tracking and collision avoidance include tracking errors, handling dynamic obstacles, dealing with latency and noise, limited sensor capabilities, nonholonomic motion constraints, local minima problems, and balancing safety with efficiency. Addressing these challenges requires the integration of robust control techniques, predictive planning, sensor fusion, and adaptive decision-making strategies.

2.3 Control of DWMR using the Dynamic Feedback Linearization concept

The technique of feedback linearization constitutes a powerful tool in the control theory of nonlinear systems [153–156]. This method is based on the principle of transforming the control problem from a complex nonlinear form into a simple and controllable linear form defined as:

$$\dot{z} = Az + Bu, \quad z \in \mathbb{R}^{n'}, \quad u \in \mathbb{R}^{m} \tag{2.13}$$

through a nonsingular (dynamic or static) feedback u and a diffeomorphism z of the extended state (or output) space. In this framework, nonlinear systems appear as analogues of linear systems. This approach simplifies the analysis, planning, and control of nonlinear systems that can be transformed into the form in (2.13). The systematic procedure for formulating a feedback linearization controller involves the following steps [157]:

- An appropriate set of flat outputs must be chosen. The number of flat outputs, must precisely match the number of system control inputs.
- These chosen flat outputs are then iteratively differentiated with respect to time until all control inputs; or their derivatives; explicitly

manifest within the resulting expressions. If it's possible to express all inputs (or their derivatives) solely in terms of the flat outputs and their derivatives, the analysis can proceed to the subsequent phase.

- The derived system of equations is subsequently solved to isolate the highest-order derivatives of each control input. To translate these into the actual physical system inputs, a series of integrators must be strategically cascaded to reverse the differentiation process. Conversely, the derivatives of the outputs serve as the new, virtual inputs to the linearized system.
- With the system now successfully transformed into a linear representation, a diverse array of established linear control laws can be effectively designed and applied to these newly defined inputs.

For a wheeled mobile robot employing a differential drive mechanism, the vector of Cartesian coordinates of the robot can be selected as the system flat outputs, i.e. $[(x(t) \;\; y(t)]^T$.
Considering the robot's kinematic model, the first time derivative of the chosen flat output vector is given as:

$$\begin{cases} \dot{x} = v\cos\theta, \\ \dot{y} = v\sin\theta \end{cases} \tag{2.14}$$

Upon first differentiation in (2.14), only the linear velocity v is explicitly present. Further differentiation then becomes necessary:

$$\begin{cases} \ddot{x} = \dot{v}\cos\theta - v\omega\sin\theta \\ \ddot{y} = \dot{v}\sin\theta + v\omega\cos\theta \end{cases} \tag{2.15}$$

Both control velocities, linear velocity v and angular velocity, now appear in these second derivatives. The system of equations can be conveniently rearranged to express the second derivatives of the flat outputs as a function of the highest derivatives of the individual inputs (in this instance, $\dot{v}$ and ω):

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} \cos\theta & -v\sin\theta \\ \sin\theta & v\cos\theta \end{bmatrix} \begin{bmatrix} \dot{v} \\ \omega \end{bmatrix} = \mathbf{F} \begin{bmatrix} \dot{v} \\ \omega \end{bmatrix} \tag{2.16}$$

Provided that the linear velocity $v \neq 0$, the matrix $\mathbf{F}$ in (2.16) is non-singular and the corresponding unique solution for this system is:

$$\begin{bmatrix} \dot{v} \\ \omega \end{bmatrix} = \mathbf{F}^{-1} \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\frac{\sin\theta}{v} & \frac{\cos\theta}{v} \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} \tag{2.17}$$

From (2.17), the derived value for ω directly corresponds to an input command for the robot. Conversely, the value for $\dot{v}$ necessitates an integration step before it can be supplied as a control input. The newly formed linear system operates with inputs $[u_1, u_2]^T = [\ddot{x}, \ddot{y}]^T$ and an augmented state vector $\mathbf{z} = [x, \dot{x}, y, \dot{y}]^T$. Note that while the kinematic model inherently possesses three states, the fourth state emerges due to the additional integration required for $\dot{v}$. The dynamics of this transformed system can be concisely expressed in the state-space representation as:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{y} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{2.18}$$

This can be more compactly written as in (2.13). This system is inherently controllable because its controllability matrix, defined as:

$$\mathbf{Q}_c = [\mathbf{B}, \mathbf{AB}] \tag{2.19}$$

possesses full rank. This inherent controllability ensures the existence of a state feedback controller capable of arbitrarily placing the closed-loop poles within the left half-plane of the complex s-plane.

A further objective is to design a suitable control law that force the robot to accurately track the specified reference trajectories $x_{ref}(t)$ and $y_{ref}(t)$. Consequently, the corresponding reference for the flat system state $\mathbf{z}_{ref}(t) = [x_{ref} \;\; \dot{x}_{ref} \;\; y_{ref} \;\; \dot{y}_{ref}]^T$ and the system input $\mathbf{u}_{ref} = [\ddot{x}_{ref} \;\; \ddot{y}_{ref}]^T$ can be readily obtained as in (2.20):

$$\dot{\mathbf{z}}_{ref} = \mathbf{A}\mathbf{z}_{ref} + \mathbf{B}\mathbf{u}_{ref} \tag{2.20}$$

The error between the actual states and the reference states $\tilde{\mathbf{z}} = \mathbf{z} - \mathbf{z}_{ref}$ can then be obtained as :

$$\dot{\tilde{\mathbf{z}}} = \mathbf{A}\tilde{\mathbf{z}} + \mathbf{B}(\boldsymbol{u} - \boldsymbol{u}_{ref}) \tag{2.21}$$

The dynamics of the state error in (2.21) must be both stable and sufficiently fast. One common approach to define the desired closed-loop dynamics is through the strategic placement of the system's poles as defined in (2.21) can be manipulated into the following form:

$$\dot{\tilde{\mathbf{z}}} = (\mathbf{A}-\mathbf{BK})\tilde{\mathbf{z}} + \mathbf{BK}\tilde{\mathbf{z}} + \mathbf{B}(\mathbf{u}-\mathbf{u}_{ref}) = (\mathbf{A}-\mathbf{BK})\tilde{\mathbf{z}} + \mathbf{B}(\mathbf{K}\tilde{\mathbf{z}} + \mathbf{u} - \mathbf{u}_{ref}) \tag{2.22}$$

If the final term in (2.22) is driven to zero, the state errors will converge to zero with the desired dynamics, governed by the closed-loop system matrix $(\mathbf{A}-\mathbf{BK})$. Setting this term to zero directly defines the control law for this methodology:

$$\mathbf{u}(t) = \mathbf{K}(\mathbf{z}_{ref}(t) - \mathbf{z}(t)) + \mathbf{u}_{ref}(t) \tag{2.23}$$

A schematic representation of the complete control system is provided in Figure. 2.2. Due to the specific configuration of matrices $\mathbf{A}$ and $\mathbf{B}$ in (2.18), where u_1 exclusively influences states z_1 and z_2, and u_2 only impacts states z_3 and z_4, the control gain matrix $\mathbf{K}$ adopts a simplified structure:

$$\mathbf{K} = \begin{bmatrix} k_1 & k_2 & 0 & 0 \\ 0 & 0 & k_3 & k_4 \end{bmatrix} \tag{2.24}$$

Consequently, the control law given by (2.23) can be entirely decomposed into two independent components:

$$\begin{cases} u_1(t) = \ddot{x}(t) = k_1(x_{ref}(t) - x(t)) + k_2(\dot{x}_{ref}(t) - \dot{x}(t)) + \ddot{x}_{ref}(t) \\ u_2(t) = \ddot{y}(t) = k_3(y_{ref}(t) - y(t)) + k_4(\dot{y}_{ref}(t) - \dot{y}(t)) + \ddot{y}_{ref}(t) \end{cases} \tag{2.25}$$

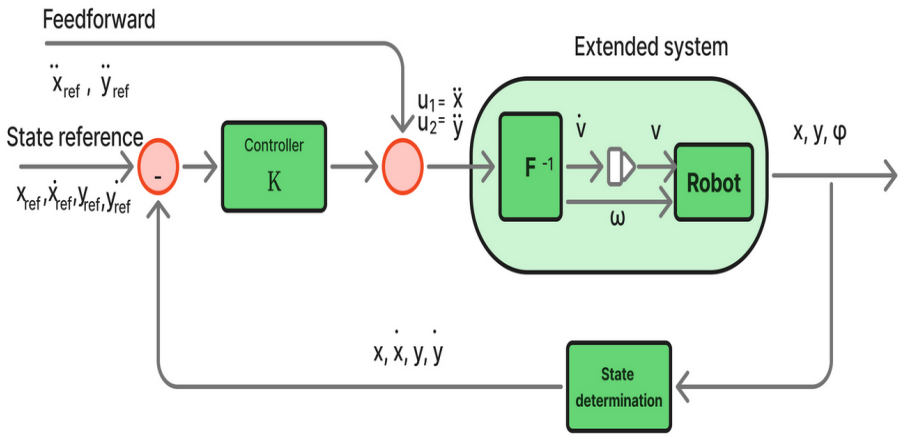


Figure 2.2: Feedback linearization for reference tracking.

The efficacy of this proposed method relies on the availability of all system states. While x and y coordinates are typically measurable, their

derivatives are often not directly accessible. The use of numerical differentiation to estimate these derivatives is generally discouraged in practical applications due to its susceptibility to noise amplification. Two primary solutions can be considered to address this challenge:

- Unmeasured states can be effectively estimated using state observers, such as Luenberger observers or Kalman filters.
- When the robot's orientation θ and linear velocity v are known, the derivatives of x and y can be computed directly using the kinematic relations defined in (2.14) :

2.4 Fuzzy Logic Control (FLC)

2.4.1 Fundamental Concepts of Fuzzy Logic

Fuzzy logic is a form of approximate reasoning that enables the modeling of complex behaviors, phenomena, or processes by handling imprecision and uncertainty. It is based on the concept of fuzzy sets, which establish a bridge between numerical (quantitative) data and linguistic (qualitative) variables. These linguistic variables are typically defined over a specific range known as the universe of discourse. To process these variables, fuzzy logic uses membership functions that map each element to a value between 0 and 1, representing the degree of membership (or truth value) of the element within different fuzzy subsets of a given class. In addition, fuzzy logic systems rely on fuzzy rulestypically expressed as a collection of IF-THEN statementsto describe the relationships between input and output variables in linguistic terms. These fuzzy rules form the core of the decision-making mechanism, allowing the system to infer conclusions and take actions based on approximate, rather than exact, information. Through this framework, fuzzy logic provides a powerful and flexible approach for representing and manipulating knowledge that is inherently vague or uncertain, closely resembling human reasoning.

A fuzzy set is a mathematical structure that allows each element to have a degree of membership between 0 and 1, rather than being simply included or excluded as in classical (crisp) sets. This degree of membership represents how strongly or weakly an element belongs to the set, enabling the modeling of uncertainty and imprecision [158]. Given a fuzzy set A defined over a universe of discourse X, its mathematical representation can

be expressed as a set of ordered pairs, where each pair associates an element $x \in X$ with its corresponding membership degree $\mu_A(x)$, as follows:

$$A = \{(x, \mu_A(x)) \mid x \in X\} \tag{2.26}$$

Where, $\mu_A(x)$ is the membership function associated with each element $x \in X$, the membership function $\mu_A(x)$ is defined by:

$$\mu_{\mathrm{A}}(x) = \begin{cases} valeur \in [0,1], & \text{if } x \in A \\ 0, & \text{if } x \notin A \end{cases} \tag{2.27}$$

A fuzzy set A in the universe of discourse X can have the following properties:

1. **Membership Function**: Every fuzzy set is characterized by a membership function $\mu_A(x)$ that maps each element x of the universe of discourse X to a value in the interval $[0,1]$. The value $\mu_A(x)$ represents the degree to which the element x belongs to the fuzzy set A.

2. **Fuzziness**: Unlike classical sets, where an element either belongs or does not belong to the set, fuzzy sets allow partial membership. This means that an element can belong to a fuzzy set to a certain degree, with values ranging from 0 (no membership) to 1 (full membership).

3. **Cardinality**: The cardinality of a fuzzy set is not just the number of elements in the set, but rather the sum of the degrees of membership of all the elements in the set. The total cardinality can be defined as:

$$Card(A) = \sum_{x \in X} \mu_A(x) \tag{2.28}$$

4. **Support**: The support of a fuzzy set A is the set of elements in X that the degree of membership is between 0 and 1. It is defined as:

$$\mathrm{Supp}(A) = \{x \in X \mid 0 < \mu_A(x) \leq 1\} \tag{2.29}$$

5. **Core**: The core of a fuzzy set A is the set of elements that have a membership degree equal to 1. It is defined as:

$$\mathrm{Core}(A) = \{x \in X \mid \mu_A(x) = 1\} \tag{2.30}$$

   The fuzzy set A is normalized if Core(A) $\neq 0$

6. α **-Cut**: For a given threshold $\alpha \in [0,1]$, the α-cut of a fuzzy set is the crisp set containing all elements whose membership degree is greater than or equal to α. It is defined as:

$$A_\alpha = \{x \in X \mid \mu_A(x) \geq \alpha\} \tag{2.31}$$

A linguistic variable is a variable whose values are words or sentences in a natural or artificial language rather than numerical values. Each linguistic value is associated with a fuzzy set defined over a universe of discourse.

A linguistic variable is typically characterized by five components:

- **Name**: The name of the variable (e.g., *Distance*).
- **Term set**: A set of linguistic values (e.g., *Near*, *Medium*, *Far*).
- **Universe of discourse**: The range of possible values (e.g., from 0 m to 4 m).
- **Syntactic rule**: A rule that defines how the linguistic values are generated.
- **Semantic rule**: A method to interpret the meaning of the terms via fuzzy membership functions.

The use of linguistic variables enables fuzzy logic to model imprecise and vague concepts in a way that closely resembles human reasoning [158].

In fuzzy logic, a membership function (MF) defines how each element in the universe of discourse is mapped to a degree of membership between 0 and 1. It characterizes the fuzziness of a fuzzy set by associating a real value (membership degree) with each element [158].

Mathematically, for a fuzzy set A defined over a universe X, the membership function $\mu_A(x)$ is expressed as:

$$\mu_A : X \to [0,1]$$

Common types of membership functions include:

- **Triangular** The triangular membership function is defined by three parameters, a, b, and c with $a < b < c$. It is given by

$$\mu_A(x) = \begin{cases} \frac{x-a}{b-a} & \text{si } x \in [a,b] \\ \frac{c-x}{c-b} & \text{si } x \in [b,c] \\ 0 & \text{others} \end{cases} \tag{2.32}$$

  where $\mu_A(x)$ is the degree of membership of x in the fuzzy set A.

- **Trapezoidal** The trapezoidal membership function is defined by four parameters a, b, c, and d with $a < b \leq c < d$. It is given by:

$$\mu_A(x) = \begin{cases} \frac{x-a}{b-a} & \text{si } x \in [a,b] \\ 1 & \text{si } x \in [b,c] \\ \frac{d-x}{d-c} & \text{si } x \in [c,d] \\ 0 & \text{others} \end{cases} \tag{2.33}$$

- **Gaussian** The Gaussian membership function is defined by two parameters, the center m and the width (standard deviation)σ, and is given by

$$\mu_A(x) = \exp\left(-\frac{(x-m)^2}{2\sigma^2}\right) \tag{2.34}$$

- **Singleton** The singleton membership function represents a fuzzy set that contains exactly one element with a membership degree equal to 1, and 0 elsewhere. It is mathematically expressed as:

$$\mu_A(x) = \begin{cases} m & \text{si } x = a \\ 0 & \text{si } x \neq a \end{cases} \tag{2.35}$$

  where a is the unique element with full membership

The graphical representation of the membership functions is shown in Figure 2.3. The selection of a membership function depends on the specific application and the level of modeling precision required.

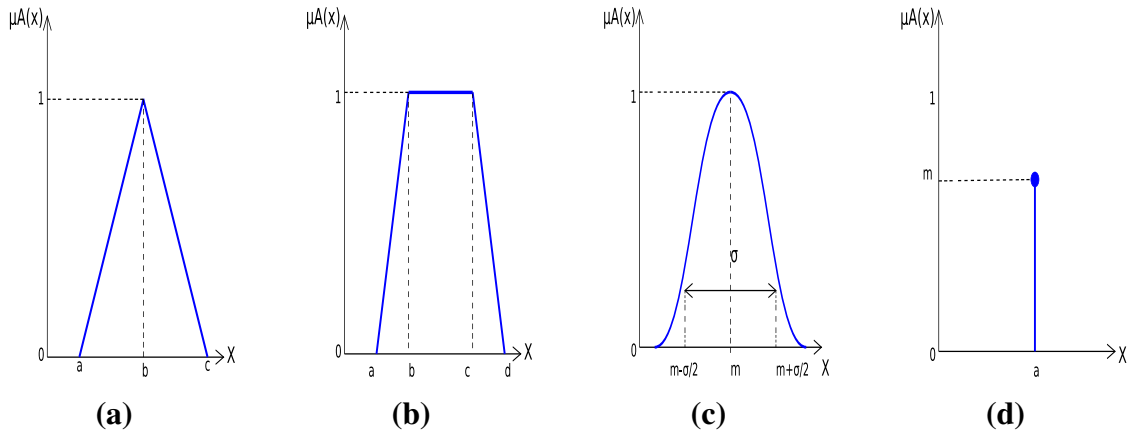


Figure 2.3: Membership functions shape: (a) Triangular, (b) Trapezoidal, (c) Gaussian, (d) Singleton.

In fuzzy logic, fuzzy operators are used to combine or modify fuzzy sets. The most common fuzzy operators are: Fuzzy AND (Intersection), Fuzzy OR (Union), Fuzzy NOT (Complement) as follows [158]:

$$\begin{aligned} \mu_{A\cap B}(x) &= \min\left(\mu_A(x), \mu_B(x)\right) \quad \text{(Fuzzy AND)} \\ \mu_{A\cup B}(x) &= \max\left(\mu_A(x), \mu_B(x)\right) \quad \text{(Fuzzy OR)} \\ \mu_{\bar{A}}(x) &= 1 - \mu_A(x) \quad \text{(Fuzzy NOT)} \end{aligned} \tag{2.36}$$

where, $\mu_{A\cap B}(x)$ is the minimum value between the degrees with which x belongs to set A and set B, $\mu_{A\cup B}(x)$ This operator corresponds to the maximum value between the degrees with which x belongs to set A and set B, and $\mu_{\bar{A}}(x)$ it gives the degree with which x does not belong to the set A:

Fuzzy rules are a core component of fuzzy logic systems. They represent knowledge in a way that can handle uncertainty and approximate reasoning. These rules are typically in the form of if-then statements. Here's an outline of fuzzy rules and how they work: A fuzzy rule takes the form of a conditional statement: If condition, then conclusion. The condition typically describes the state of the input variables (e.g., "distance is far"), while the conclusion provides an action or decision (e.g., "the robot moves straight forward to the goal"). The establishment of these fuzzy rules is based on knowledge derived from human expertise, which represents the main factor in building a base of fuzzy rules capable of describing the evolution of the system [158].

A fuzzy rule can be defined as a fuzzy implication between two fuzzy propositions. It has the form 'If "premise" Then "conclusion"' where the premise part is compatible with the first proposition and the conclusion part matches the second proposition. The premise and conclusion of a fuzzy rule can consist of several propositions combined by different operators, such as conjunction and disjunction. A set of fuzzy rules forms what we call a fuzzy rule base, which allows the inclusion of the maximum amount of information needed for a detailed representation of the system A basic fuzzy rule is of the form:

"If x is A, then y is B."

"If x is A and y is B and $\ldots$, then z_1 is C and z_2 is D and $\ldots$"

$$\begin{cases} R_1: \text{ If } x \text{ and } A_1 \text{ and } y \text{ and } B_1 \text{ and } \ldots \text{ Then } z_1 \text{ and } C_1 \text{ and } z_2 \text{ and } D_1 \text{ and } \ldots \\ R_2: \text{ If } x \text{ and } A_2 \text{ and } y \text{ and } B_2 \text{ and } \ldots \text{ Then } z_1 \text{ and } C_2 \text{ and } z_2 \text{ and } D_2 \text{ and } \ldots \\ \vdots \\ R_n: \text{ If } x \text{ and } A_n \text{ and } y \text{ and } B_n \text{ and } \ldots \text{ Then } z_1 \text{ and } C_n \text{ and } z_2 \text{ and } D_n \text{ and } \ldots \end{cases}$$

Each fuzzy rule R_i (i=1,..., n) in the rule base is considered during the inference step if the truth degree of the propositions in its predicate is non-zero. It has a truth value R_i , which depends on the membership degrees of the linguistic variables associated with the premise and conclusion parts. The case of a fuzzy implication between two propositions is defined as follows:

$$\mu_R(x,y) = imp\left(\mu_A(x), \mu_B(y)\right) \tag{2.37}$$

In other case of a fuzzy implication that includes several propositions is defined as follows:

$$\mu_{R_i}(x,y,\ldots) = imp\left(\{\mu_{A_i}(x) \; and \; \mu_{B_i}(y) \; and \ldots\}, \{\mu_{C_i}(z_1) \; and \; \mu_{D_i}(z_2) \; and \ldots\}\right) \tag{2.38}$$

2.4.2 Principles of fuzzy logic in control systems

Fuzzy logic control is based on the principles of fuzzy set theory, where system variables are represented by linguistic terms rather than precise numerical values. Unlike conventional control methods, which require accurate mathematical models, fuzzy control systems are designed using a set of heuristic rules derived from expert knowledge or observed system behavior [159].

The key advantage of fuzzy logic in control systems is its ability to model and manage systems that are complex, nonlinear, or poorly understood, using approximate reasoning similar to human decision-making processes. This flexibility makes fuzzy controllers particularly robust, easy to interpret, and capable of functioning effectively without requiring high computational power.

2.4.3 Structure of an FLC system:

A fuzzy control system operates through three fundamental stages [160]:

- **Fuzzification**: Crisp input values are mapped to fuzzy sets using predefined membership functions.
- **Inference Mechanism**: A set of fuzzy rules (typically of the form "If *condition*, then *action*") is evaluated to determine the fuzzy output sets.
- **Defuzzification**: The resulting fuzzy output sets are converted back into crisp control actions, typically using methods such as the centroid or weighted average.

A fuzzy control system consists mainly of four blocks as shown in Figure 2.4 .

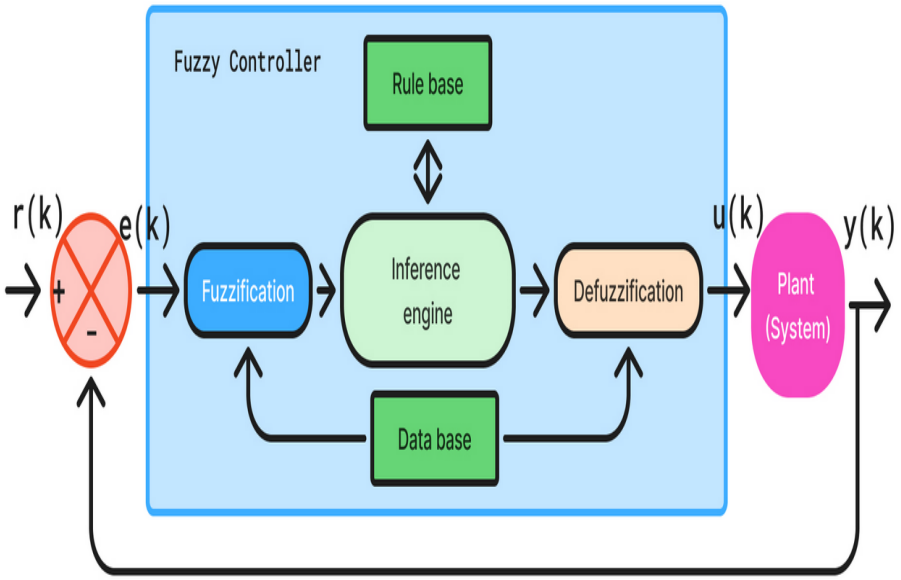


Figure 2.4: Structure of Fuzzy Control.

Fuzzification is the first step in a fuzzy logic system, where crisp numerical inputs are transformed into fuzzy sets. This process involves mapping each input value to one or more linguistic terms through predefined membership functions. The result of fuzzification is a degree of membership between 0 and 1, representing how strongly the input belongs to each fuzzy set [160]. Mathematically, if x is a crisp input and A is a fuzzy set, the fuzzification process determines the membership value $\mu_A(x)$, where: $\mu_A(x) \in [0,1]$. Fuzzification allows the system to handle uncertainty and

imprecision by working with qualitative descriptions of input data, making it possible to apply fuzzy inference rules in subsequent stages. Two types of fuzzification can be distinguished: Singleton fuzzification is a simplified fuzzification method used in many fuzzy logic systems, particularly in fuzzy control applications. In this approach, each crisp input is directly associated with a fuzzy set that has a membership degree of 1 at a single point and 0 elsewhere. The membership function of a singleton fuzzy set centered at x_0 is given by:

$$\mu_A(x) = \begin{cases} 1, & \text{if } x = x_0, \\ 0, & \text{otherwise.} \end{cases} \tag{2.39}$$

In singleton fuzzification, the crisp input value is treated as a "spike" or a "delta function" over the input universe. This simplifies the inference and computation processes since the fuzzified input does not involve overlapping membership functions and complex evaluations. Singleton fuzzification is particularly useful for real-time control systems where computational efficiency is critical. Non-singleton fuzzification is an advanced method where the crisp input is associated with a fuzzy set that has a spread over a range of values rather than being concentrated at a single point. Unlike singleton fuzzification, where the input is treated as a point with full certainty, non-singleton fuzzification models the uncertainty or imprecision inherent in real-world measurements. In this case, the input is represented by a fuzzy set $I(x)$ rather than a single value. The degree of membership between the input and the fuzzy sets in the rule base is computed through the fuzzy intersection, typically using operations like the minimum or product:

$$\begin{aligned} \mu_A(I) &= \sup_x \left(\min\left(\mu_A(x), I(x)\right)\right) \\ \text{or} \qquad \mu_A(I) &= \sup_x \left(\mu_A(x) \times I(x)\right) \end{aligned} \tag{2.40}$$

where:

- $\mu_A(x)$ is the membership function of the fuzzy set A,
- $I(x)$ is the membership function of the input fuzzy set,
- sup denotes the supremum (maximum) operator over all x.

Non-singleton fuzzification is particularly beneficial when inputs are noisy, uncertain, or imprecise, providing a more realistic modeling of sensor data and improving the robustness of fuzzy inference.

2.4.4 Knowledge base in fuzzy logic systems

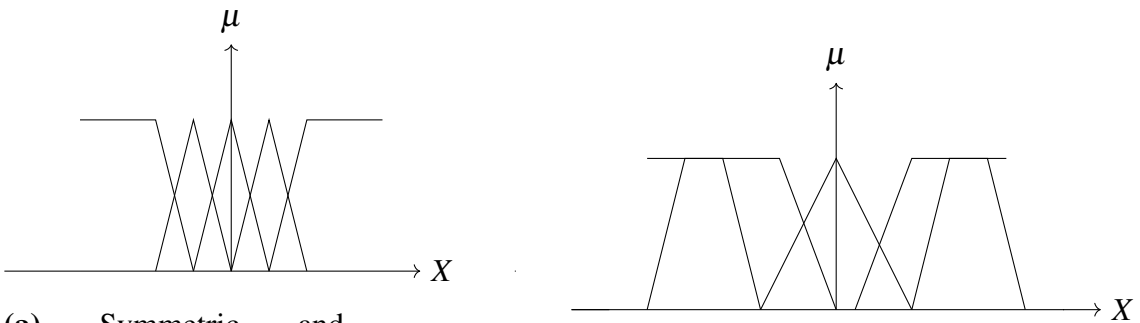
The knowledge base in a fuzzy logic system is a fundamental component that contains all the necessary information to perform fuzzy reasoning. It consists of two main parts:

- **Data base**: Defines the linguistic variables, their corresponding linguistic terms, and the shapes and parameters of the membership functions used for fuzzification and defuzzification.
- **Rule base**: Contains a collection of fuzzy if-then rules that describe the behavior of the system, based on expert knowledge or empirical observations.

The knowledge base serves as the system's memory, allowing it to interpret input information, perform inference, and generate appropriate control actions or decisions. Its proper design directly influences the performance, accuracy, and robustness of the fuzzy system. The database includes the determination of fuzzy sets of input and output variables, the distribution of these fuzzy sets in the universe of discourse and the specification of membership functions that express them. When performing these three tasks, it is best to consider the following tips: The set of possible values for a variable constitutes its universe of discourse. Regarding the number of fuzzy sets, it should be noted that initially, a minimum number of fuzzy rules is best. Consider 3 or 5 fuzzy sets, that is, each variable has 3 or 5 membership functions. Preferably, choose membership functions that are triangular and/or trapezoidal in shape, which are distributed symmetrically and equidistantly (see Figure 2.5).

The number, shape and distribution of membership functions can be modified during the execution of the process in order to achieve the desired results. However, it must be taken into account that insufficient overlap of membership functions. Will create areas where the controller does not intervene, and excessive overlap must be prevented, especially when is equal to 1 (see Figure 2.5).

Fuzzy rules are conditional if-then statements, as shown that relate fuzzy input variables to fuzzy output variables within a fuzzy inference system. They use linguistic terms to express expert knowledge and allow reasoning under uncertainty by mapping fuzzy inputs to fuzzy outputs based on degrees of membership rather than precise values. The structure and number of these rules, respectively, depend on the number of input and output variables and the number of fuzzy sets describing these variables



(a) Symmetric and equidistant

(b) Overlap of membership functions

Figure 2.5: Comparison of membership function arrangements.

The inference mechanism in fuzzy logic is the process of applying fuzzy rules to given input data to generate fuzzy outputs. It involves evaluating the degree to which each rule applies (rule evaluation), combining the results of all activated rules (aggregation), and preparing the fuzzy output set for defuzzification. The inference mechanism allows decision-making in environments characterized by uncertainty and imprecision [158]. The Max-Min inference method is a widely used fuzzy reasoning technique where the minimum operator is used to evaluate the degree of match between the antecedent (IF part) of each fuzzy rule and the input data, and the maximum operator is used to aggregate the outputs of all the rules. Specifically, for each rule, the degree of fulfillment is calculated as the minimum membership value among the inputs, and the final output fuzzy set is obtained by taking the maximum of the outputs from all activated rules. The Max-Product inference method is a fuzzy reasoning technique where the logical "AND" operation within a fuzzy rule is performed using the product (multiplication) of membership degrees, and the aggregation of multiple rule outputs is performed using the maximum operator. In this method, for each rule, the degrees of membership are combined by multiplication, and the outputs from all rules are aggregated by taking the maximum membership value at each point. The Sum-Product inference method is a fuzzy reasoning approach where the logical "AND" operations in the rule evaluation are replaced by multiplication (product) and the aggregation of multiple rule outputs is performed by addition (sum). This method uses the product operator to combine the degrees of membership within a rule and sums the contributions from all rules to form the final fuzzy output.

Defuzzification is the process of converting a fuzzy output set, produced by the inference mechanism, into a single crisp numerical value.

It translates the fuzzy conclusions derived from fuzzy rules into a precise action or decision. Defuzzification is essential in fuzzy logic systems to provide real-world control signals or responses based on fuzzy reasoning. Many defuzzification techniques can be distinguished, the most common of which are [158]

- **Center of Gravity**: The Center of Gravity (COG) defuzzification method, also called the centroid or centroid of area (COA) method, is one of the most widely used techniques for converting a fuzzy output set into a crisp value. As illustrated in Figure. 2.6, this method determines the crisp output by calculating the centroid (center of mass) of the aggregated fuzzy set's membership function. In essence, it finds the balance point of the area under the curve, taking into account both the possible output values and their corresponding degrees of membership.The disadvantages of this method lie in the complexity and duration of the calculation

  The COG method operates by evaluating the following formula:

$$X_R^* = \frac{\int_{X_{R1}}^{X_{R2}} X_R \cdot \mu_{res}(X_R)\, dX_R}{\int_{X_{R1}}^{X_{R2}} \mu_{res}(X_R)\, dX_R} \tag{2.41}$$

  where: x^* is the defuzzified crisp output, $\mu(x)$ is the membership function of the fuzzy output set, X is the domain of the output variable.

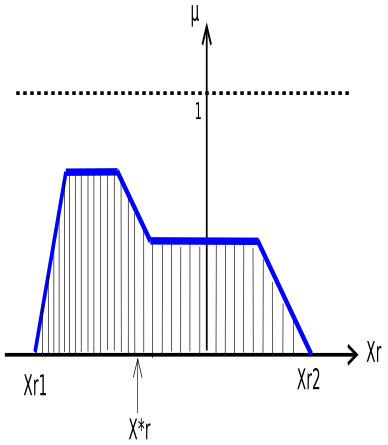
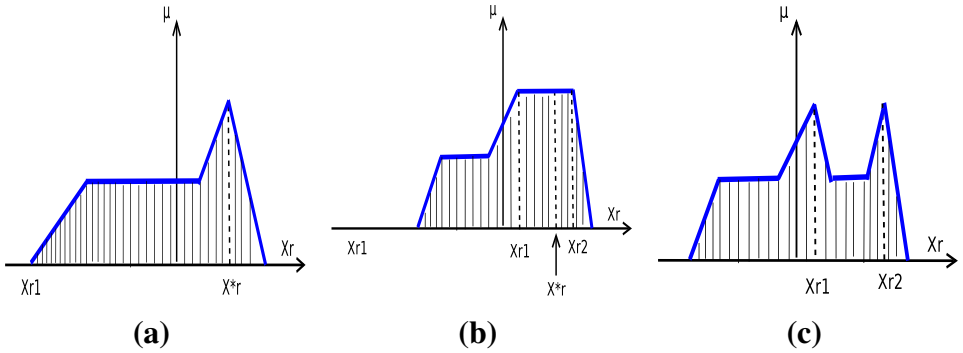


Figure 2.6: Center of gravity defuzzification.

- **Maximum Value Defuzzification** The Maximum Value defuzzification method selects the crisp output by choosing the value corresponding to the highest membership degree in the fuzzy output set

As illustrated in Figure. 2.7 This method is simple and computationally efficient but may not always reflect the most representative or balanced value when the fuzzy output has multiple peaks.



(a) **(b)** **(c)**

Figure 2.7: Maximum value defuzzification.

Defuzzification by the average of the maxima" (also called Mean of Maxima, or MoM) is a simple and common defuzzification method used in fuzzy logic systems As illustrated in Figure

The procedure is as follows:

- Find all points x_i where the membership function $\mu(x)$ attains its maximum value $\mu_{\max}$.
- Compute the arithmetic mean of these points.

Let M be the set of all points where the membership function is maximal, is given by

$$M = \{x_i \mid \mu(x_i) = \mu_{\max}\}, \tag{2.42}$$

Then the defuzzified crisp output x^* is given by:

$$x^* = \frac{1}{|M|} \sum_{x_i \in M} x_i, \tag{2.43}$$

where $|M|$ is the number of maxima points, x_i are the points.

2.4.5 Different types of fuzzy systems

Generally, all fuzzy systems follow the same design steps, as presented above. The only difference lies in the inference mechanism, at the level of fuzzy rules, particularly in the structure of consequent propositions. Following this structure, two types of fuzzy models are proposed: Mamdani fuzzy systems and Takagi-Sugeno fuzzy systems.

The Mamdani fuzzy inference system, proposed by Ebrahim Mamdani in 1975, is one of the most widely used fuzzy models for decision-making and control problems. It is a rule-based system where both the antecedents (IF parts) and consequents (THEN parts) are represented by fuzzy sets. Each fuzzy rule takes the form:

$$\text{“IF } x_1 \text{ is } A_1 \text{ AND } x_2 \text{ is } A_2 \text{ AND } \ldots \text{ THEN } y \text{ is } B,\text{”}$$

Where $x_1, x_2, \ldots$ are input variables, y is the output variable $A_1, A_2, \ldots$, and B are fuzzy sets defined by their membership functions.

The fuzzy inference process in a Mamdani system involves four main steps. First, during fuzzification, crisp inputs are converted into fuzzy values by computing their degrees of membership in the corresponding input fuzzy sets. Second, in the rule evaluation (inference) step, the activation level of each rule is computed, typically using fuzzy logical operations such as minimum (for AND) and maximum (for OR). Third, during aggregation, the fuzzy outputs of all rules are combined into a single fuzzy set, often using the maximum operator. Finally, the defuzzification step transforms the aggregated fuzzy output into a crisp value. Common defuzzification methods include the centroid (center of gravity) method, the mean of maxima, and the bisector of area [161].

In general, if there are n fuzzy rules, the aggregated fuzzy output $B'(y)$ can be expressed as:

$$B'(y) = \max_{i=1,\ldots,n} \left(\min \left(\mu_{A_{i1}}(x_1), \mu_{A_{i2}}(x_2), \ldots \right), \mu_{B_i}(y) \right), \tag{2.44}$$

where $\mu_{A_{ij}}(x_j)$ is the membership degree of input x_j in fuzzy set A_{ij}, and $\mu_{B_i}(y)$ is the membership function of the output fuzzy set B_i.

The Mamdani fuzzy system is especially intuitive because it uses linguistic rules that closely resemble human reasoning. It is commonly applied in various fields such as control systems (for motors, robots, and climate control), decision support systems, pattern recognition, and medical diagnosis. However, the computational complexity, particularly during defuzzification using the centroid method, can be higher compared to other fuzzy models like the Sugeno system.

The Takagi-Sugeno fuzzy inference system, introduced by Takagi and Sugeno in 1985, is a type of fuzzy system where the consequent of each rule is a mathematical function of the input variables rather than a fuzzy set. It is designed to provide precise outputs and is particularly suitable for control applications and system modeling where accurate and computationally efficient results are needed. Each fuzzy rule in a Takagi-Sugeno

system has the form:

$$\text{IF } x_1 \text{ is } A_1 \text{ AND } x_2 \text{ is } A_2 \text{ AND } \ldots \text{ THEN } y = f(x_1, x_2, \ldots, x_n), \quad (2.45)$$

where $x_1, x_2, \ldots$ are the input variables, $A_1, A_2, \ldots$ are fuzzy sets associated with the inputs, and $f(x_1, x_2, \ldots, x_n)$ is a crisp function, usually linear or constant, describing the output.

The inference process in a Takagi-Sugeno system involves three main steps. First, during fuzzification, the input values are evaluated against the antecedent fuzzy sets to compute their membership degrees. Second, in the rule evaluation step, the firing strength of each rule is computed, typically by applying the product (multiplication) or the minimum operator to the membership degrees. Third, the overall output is calculated by taking a weighted average of the individual rule output, where the weights are the firing strengths.

In general, if there are n rules and the firing strength of the i-th rule is w_i, and its output function is $f_i(x_1, \ldots, x_n)$, the final crisp output y is given by:

$$y = \frac{\sum_{i=1}^{n} w_i f_i(x_1, \ldots, x_n)}{\sum_{i=1}^{n} w_i}. \quad (2.46)$$

Takagi-Sugeno fuzzy systems offer several advantages. They are computationally efficient, which makes them well-suited for real-time control. Moreover, because the consequents are functions rather than fuzzy sets, defuzzification is not required, simplifying the inference process. These systems are widely used in areas such as nonlinear system modeling, adaptive control, optimization, and prediction tasks. However, they are often considered less intuitive than Mamdani systems because the rules involve mathematical expressions instead of linguistic descriptions [161].

2.4.6 Application to obstacle avoidance using fuzzy logic in mobile robots

Fuzzy logic has been widely applied to the obstacle avoidance problem in mobile robots due to its ability to handle uncertainty, imprecision, and nonlinearities in dynamic environments. It provides a flexible framework for decision making based on linguistic rules, without requiring an exact mathematical model of the environment or the robot's dynamics [162]. In a typical fuzzy logic-based obstacle avoidance system, the robot's sensors, such as laser range finders, ultrasonic sensors, or vision systems, provide information about the surrounding obstacles. This sensor information is

fuzzified into linguistic variables such as *"Near"*, *"Medium"*, and *"Far"* distances. The fuzzy inference system then processes these fuzzy inputs using a set of heuristic rules. A typical rule might be as follows: The output of the fuzzy inference system typically includes linear and angular velocities that dictate the robot's motion. These outputs are then defuzzified into crisp control signals using methods such as *center of gravity* or *average of maxima* techniques. Fuzzy logic controllers can effectively generate smooth and adaptive motions, allowing the robot to avoid both static and dynamic obstacles without abrupt maneuvers. Moreover, fuzzy systems can integrate multiple objectives simultaneously, such as maintaining a safe distance from obstacles while still progressing towards a goal.

One of the main advantages of using fuzzy logic for obstacle avoidance is its robustness to sensor noise and its ability to generalize well in uncertain or partially known environments. However, designing an effective fuzzy system requires careful tuning of the membership functions and rule bases, often relying on expert knowledge or learning algorithms. In conclusion, the application of fuzzy logic to obstacle avoidance enables mobile robots to operate more intelligently and autonomously in complex, dynamic environments by mimicking human-like decision-making processes.

2.4.7 Strengths of FLC

Fuzzy Logic Controllers (FLCs) have been widely used in control systems, including mobile robotics, due to their ability to handle complex and uncertain environments. They offer several notable strengths, but also present specific challenges that must be addressed for effective application.

- **Robustness to Uncertainty and Noise:** FLCs can tolerate imprecise, noisy, or incomplete input information, which makes them well suited for real-world environments.
- **Model-Free Design:** FLCs do not require an accurate mathematical model of the system. They can be designed based on expert knowledge and intuitive reasoning.
- **Flexibility and Adaptability:** Fuzzy controllers can be easily modified or extended by adjusting membership functions and rule bases without redesigning the entire control structure.

- **Nonlinear Control Capability:** FLCs naturally handle non-linearities in system behavior, which are often difficult to manage using classical linear control techniques.
- **Human-Like Reasoning:** FLCs mimic human decision-making processes, making them intuitive to understand and implement, especially for tasks like navigation and obstacle avoidance.

2.4.8 Typical challenges of FLC

- **Rule Explosion:** As the number of input variables increases, the number of fuzzy rules can grow exponentially, leading to a complex and difficult-to-manage rule base.
- **Tuning Complexity:** Designing appropriate membership functions and tuning fuzzy rules require expert knowledge or sophisticated optimization techniques.
- **Lack of Formal Stability Analysis:** Unlike classical control systems, providing formal guarantees of stability and performance for FLCs is generally difficult.
- **Computational Load:** In real-time applications, especially with high-dimensional input spaces, FLCs can impose significant computational burdens.
- **Scalability Issues:** Extending an FLC to handle more complex tasks or larger environments often requires significant redesign efforts.

In summary, while Fuzzy Logic Controllers offer powerful and flexible solutions for controlling systems in uncertain and dynamic environments, their design and implementation require careful attention to scalability, tuning, and computational efficiency to overcome their inherent challenges.

2.5 Integration of FL concept and DFL approach for hybrid control

Hybrid control strategies combine the strengths of different control techniques to achieve robust and adaptive behavior in mobile robots. One such powerful combination is the integration of Fuzzy Logic Control (FLC) with Dynamic Feedback Linearization (DFL). Dynamic Feedback Linearization

is a nonlinear control method that transforms a nonlinear system into an equivalent linear system through nonlinear coordinate transformation and input-state feedback. It provides precise trajectory tracking for systems with known and well-modeled dynamics. However, DFL can be sensitive to model uncertainties and external disturbances, which are common in real-world robotic environments. To address this limitation, Fuzzy Logic is introduced as a complementary control layer. FLC is a rule-based approach that handles uncertainties, imprecise data, and nonlinear behaviors by mimicking human reasoning through fuzzy if-then rules. When integrated with DFL, fuzzy logic can adaptively tune control parameters, compensate for modeling errors, or act as a supervisory controller.

The hybrid controller thus leverages the high accuracy of DFL for nominal system behavior, while fuzzy logic ensures robustness and adaptability in uncertain or dynamic conditions. This integration enhances the overall stability, precision, and autonomy of mobile robots navigating in complex environments.

In autonomous navigation, a mobile robot often requires both precise tracking of a predefined trajectory and real-time obstacle avoidance. This can be achieved by fusing two control paradigms: Dynamic Feedback Linearization (DFL) for trajectory tracking and Fuzzy Logic Control (FLC) for obstacle avoidance. The key challenge is deciding when to switch between these controllers based on the robots environment.

Let $\mathbf{z}(t)$ denote the flat output state vector of the robot:

$$\mathbf{z}(t) = \begin{bmatrix} x(t) & \dot{x}(t) & y(t) & \dot{y}(t) \end{bmatrix}^T, \tag{2.47}$$

and the reference trajectory be:

$$\mathbf{z}_{\text{ref}}(t) = \begin{bmatrix} x_{\text{ref}}(t) & \dot{x}_{\text{ref}}(t) & y_{\text{ref}}(t) & \dot{y}_{\text{ref}}(t) \end{bmatrix}^T. \tag{2.48}$$

The control objective is to track $\mathbf{z}_{\text{ref}}(t)$ using DFL:

$$\dot{\mathbf{z}} = \mathbf{A}\mathbf{z} + \mathbf{B}\mathbf{u}, \tag{2.49}$$

where $\mathbf{u} = \begin{bmatrix} u_1 & u_2 \end{bmatrix}^T$ is the control input.

To ensure obstacle avoidance, a fuzzy logic controller $\mathbf{u}_f$ is used when an obstacle is detected within a safety region. The switching logic is defined using a Boolean condition $S(t)$:

$$S(t) = \begin{cases} 1, & \text{if } d_{\min}(t) < d_{\text{safe}} \\ 0, & \text{otherwise} \end{cases}, \tag{2.50}$$

where $d_{\min}(t)$ is the minimum distance to an obstacle, and d_{safe} is a predefined threshold.

The final hybrid control input $\mathbf{u}_{\text{hybrid}}$ is then defined as:

$$\mathbf{u}_{\text{hybrid}} = (1 - S(t)) \cdot \mathbf{u}_{\text{DFL}} + S(t) \cdot \mathbf{u}_f, \tag{2.51}$$

where $\mathbf{u}_{\text{DFL}}$ is the control law derived from feedback linearization and $\mathbf{u}_f$ is computed by the fuzzy logic rule base using inputs like distance to obstacle and relative angle.

This fusion strategy ensures:

- Accurate tracking when the path is free of obstacles (DFL active).
- Smooth and reactive obstacle avoidance when hazards are detected (FLC active).
- Continuous operation without control conflict via a clean switch mechanism.

2.6 Conclusion

In this chapter, we present the fundamental concepts of fuzzy logic and its various types, followed by an overview of dynamic feedback linearization. Subsequently, we investigate a hybrid control strategy that integrates Dynamic Feedback Linearization (DFL) with fuzzy logic to effectively address trajectory tracking and obstacle avoidance in mobile robots. DFL provides a solid mathematical framework to handle nonlinear robot dynamics, ensuring accurate path tracking. However, its sensitivity to model uncertainties and environmental changes necessitates a more adaptive layer. The integration of fuzzy logic enhances system robustness and flexibility by enabling real-time adjustments based on environmental feedback. The proposed approach demonstrates improved performance in static and dynamic environments, offering a balance between precision and adaptability. In the next chapter, we will introduce a hybrid path planning algorithm that combines the Efficient A* (EA*) algorithm for global path planning with the Fuzzy Dynamic Window Approach (FDWA) for local path planning, targeting navigation in static and known environments.

3

An Intelligent Navigation Based on Efficient A-Star Algorithm and Fuzzy Dynamic Window Approach in Static Environments

This chapter presents an intelligent navigation system for mobile robots in static environments by combining an Efficient A-Star (EA) algorithm for global path planning and a Fuzzy-Dynamic Window Approach (FDWA) for local motion control. The Adaptive A-Star algorithm reduces exploration overhead by dynamically adjusting heuristic weights. The fuzzy logic controller improves obstacle avoidance and trajectory smoothness by tuning DWA parameters based on environmental and robot state inputs.*

Contents

3.1 Introduction

Efficient navigation in static environments necessitates the integration of a reliable global path planner with a responsive local obstacle avoidance strategy to ensure both safety and performance. Among global planning methods, the A* algorithm remains a popular choice due to its completeness and ability to find optimal paths. Nevertheless, the conventional A* approach often yields paths that are geometrically inflexible and less practical for real-world navigation, particularly when the robots maneuverability is constrained. To overcome these limitations, the Adaptive A* algorithm enhances classical A* by updating the heuristic function dynamically during the search process, which accelerates replanning and improves adaptability in complex static environments. On the local level, the Dynamic Window Approach (DWA) is frequently employed for its ability to incorporate the robots kinematic and dynamic constraints while enabling real-time obstacle avoidance. Despite its effectiveness, DWAs performance heavily depends on proper parameter tuning, which can reduce its generalization across different settings. To mitigate this, integrating fuzzy logic into DWA enhances its decision-making capability by adaptively adjusting velocity commands based on obstacle proximity and goal orientation. This chapter introduces a hybrid navigation strategy that unites Adaptive A* for efficient global trajectory generation with a Fuzzy Logic-augmented DWA for adaptive local control. The combination of these two methods results in a robust and intelligent navigation system tailored for mobile robots operating in static indoor or outdoor environments.

3.2 Traditional A-Star algorithm

3.2.1 Environment model description

Before initiating any path planning process, a mobile robot must first perceive and interpret its environment by constructing an internal representation of the surroundings. This environmental model plays a crucial role in enabling autonomous navigation and informed decision-making, particularly in complex or cluttered settings. Several mapping methodologies have been developed for this purpose, including grid-based maps, Voronoi diagrams, and topological representations [163, 164]. In this research, the environment is represented using a grid map, which divides the workspace into a matrix of equally sized cells. Each cell corresponds to a specific

area within the environment, where free space is denoted in white and obstacles are shown in black. This discretized structure reduces environmental complexity and enables more efficient path planning by minimizing the search space for algorithms such as A*. The resolution of the griddefined by the size of individual cellsplays a critical role in determining the accuracy of environmental representation. Smaller cells allow for a more precise depiction of obstacles and spatial features [165]. However, increasing the resolution also raises computational and memory requirements due to the larger number of cells. Therefore, selecting an appropriate cell size involves balancing the level of detail captured with the available computational resources to ensure both reliable obstacle detection and efficient path planning. In this experiment, a grid map measuring 20×20 cells was implemented using MATLAB, with each cell corresponding to a 1 square meter area, as illustrated in Figure 3.1.

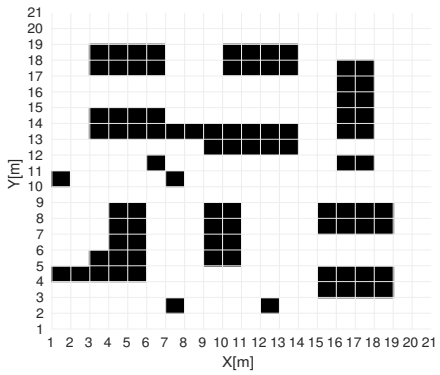


Figure 3.1: Grid map: black and white cells represents occupied and free spaces respectively.

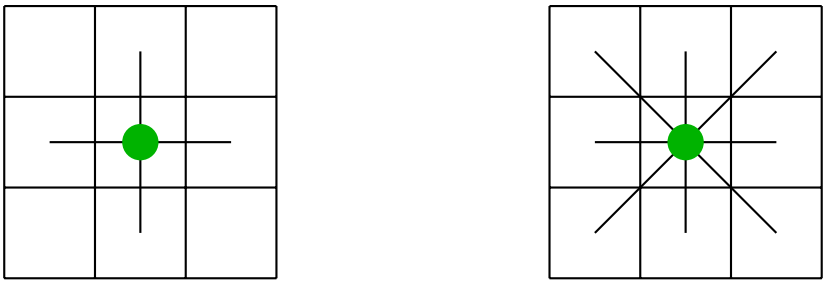
3.2.2 Neighborhood search strategy

In the implementation of the A* algorithm, the strategy for selecting neighboring nodes is crucial to the overall efficiency and accuracy of the pathfinding process [166, 167]. This strategy determines which surrounding cells of the current node are evaluated during each iteration of the search.

Two widely used neighborhood models are the 4-connected and 8-connected configurations. The 4-connected model considers only the immediate vertical and horizontal neighborsthose directly above, below, to the left, and to the right of the current node. Although this approach reduces computational complexity, it may limit the algorithms ability to identify op-

timal paths in environments with intricate structures. In contrast, the 8-connected model expands the search to include diagonal neighbors as well, allowing for greater path flexibility and often resulting in shorter, more natural trajectories.

Considering these benefits, this study employs the 8-connected neighborhood configuration to improve the A* algorithms capability in navigating through the mapped environment.



(a) 4-connected grid **(b)** 8-connected grid

Figure 3.2: Grid connectivity: (a) 4-connected and (b) 8-connected neighborhood.

The A* algorithm is a well-established heuristic-based search method frequently utilized in mobile robotics for global path planning, particularly in environments that are static and fully known. Its primary objective is to determine an optimal or near-optimal path from a designated start node to a target node by minimizing a total cost function. When applied to grid-based maps, A* initiates the search from the cell corresponding to the start location and systematically explores neighboring cells in all eight directions, including diagonals.

During each iteration, the algorithm selects the node with the lowest estimated total cost, which is computed using an evaluation function. This function combines the actual cost accumulated to reach the current node with a heuristic estimate of the remaining cost to the goal. The algorithm continues this process of expansion until the goal node is reached and identified as the current node [168, 169].

The evaluation function $F(n)$ used in A* is defined as:

$$F(n) = G(n) + H(n) \tag{3.1}$$

where $G(n)$ represents the accumulated cost from the start node to the current node n, and $H(n)$ denotes a heuristic estimate of the cost from node n to the goal node [168]. The combination of these two terms enables

the algorithm to balance exploration and exploitation, directing the search towards the most promising path.

An illustrative example of a global path generated using the A* search algorithm on a grid map is shown in Figure 3.3.

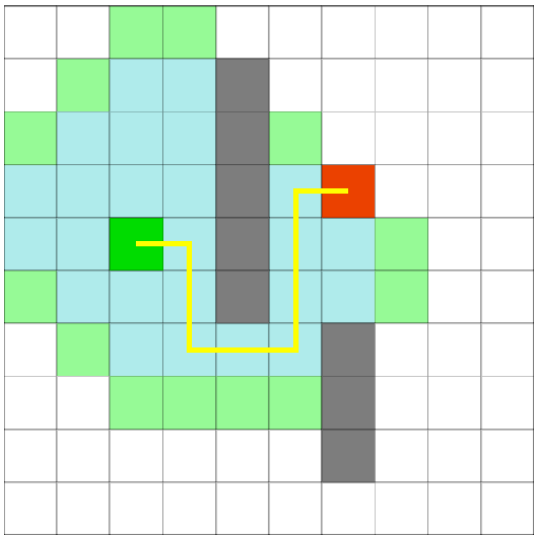


Figure 3.3: Path generated by the A* grid search algorithm.

The cost $G(n)$ is typically calculated as the Euclidean distance between the start node (x_s, y_s) and the current node (x_n, y_n):

$$G(n) = \sqrt{(x_n - x_s)^2 + (y_n - y_s)^2} \tag{3.2}$$

Similarly, the heuristic cost $H(n)$ is the estimated Euclidean distance from the current node to the goal node (x_g, y_g):

$$H(n) = \sqrt{(x_g - x_n)^2 + (y_g - y_n)^2} \tag{3.3}$$

where (x_n, y_n), (x_s, y_s), and (x_g, y_g) denote the coordinates of the current node, the start node, and the goal node, respectively.

The core operational steps of the classical A* algorithm are summarized in Algorithm 3.1 [168].

3.3 Efficient A-Star algorithm

3.3.1 Node optimization

In the field of mobile robotics, the conventional A* algorithm has been widely adopted for path planning due to its simplicity and reliability. However, it exhibits limitations when dealing with dynamic or temporary obsta-

Algorithm 3.1 A* Algorithm for path planning

Input: Grid map *map*, start node *start*, goal node *goal*
Output: Path *path*, boolean *goal_reached*, cost *cost*, list *EXPAND*
Initialize $OPEN \leftarrow \{start\}$, $CLOSED \leftarrow \emptyset$, $EXPAND \leftarrow \emptyset$
Initialize $cost \leftarrow 0$, $goal\_reached \leftarrow$ false, define *motion* for 8 directions
while $OPEN \neq \emptyset$ **do**
    Compute $f(n) = g(n) + h(n, goal)$ for all $n \in OPEN$
    Select and remove node n with lowest f from $OPEN$, add n to $CLOSED$
    if $n == goal$ **then**
        $goal\_reached \leftarrow$ true, $cost \leftarrow g(n)$, **break**
    end if
    for each motion in *motion* **do**
        Compute neighbor n', skip if n' in $CLOSED$ or obstacle
        if $n' \notin OPEN$ or better $g(n')$ **then**
            Update $g(n')$, $f(n')$, parent of n', add to $OPEN$ if needed
        end if
    end for
    Add n to $EXPAND$ if not already present
end while
Reconstruct *path* from *goal* to *start* via parent links
Return *path*, *goal_reached*, *cost*, *EXPAND*

cles. Additionally, its exhaustive node exploration strategy often results in redundant and unnecessary node expansions, which may include so-called infection points that contribute little to optimal navigation.

The traditional A* procedure begins by placing the start node in the open list and moving it to the closed list as the parent node. Subsequently, the algorithm expands the neighboring nodes in all eight directions, progressing iteratively from the start toward the goal. While this comprehensive search ensures completeness, it lacks adaptability when confronted with varying environmental complexities. In relatively simple environments with sparse obstacles, the full 8-directional search may be excessive and computationally inefficient.

To address this limitation, this study introduces a reduced-direction strategy based on five directional sub-node searches. This approach dynamically aligns the search direction by connecting the current node to the goal node and determining the angle θ between this vector and the positive y-axis, as defined in (3.4). Using this angle, a search sector centered on the current node is established, wherein only five directional expansions are considered, provided the region is obstacle-free. This reduction in search directions improves efficiency without compromising path quality.

The concept is illustrated in Figure 3.4, and the corresponding direc-

tional strategy is summarized in Table 3.1.

$$\theta = \arctan \frac{g_y - n_y}{g_x - n_x} \tag{3.4}$$

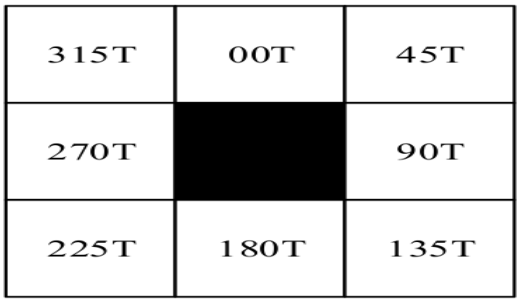


Figure 3.4: Node search direction.

Table 3.1: Child node search direction strategy.

Θ	Keep 5 Directions	Abandon Direction
$[337.5°, 360°) \cup [0°, 22.5°)$	000T, 045T, 090T, 270T, 315T	135T, 180T, 225T
$[22.5°, 67.5°)$	000T, 045T, 090T, 135T, 315T	180T, 225T, 270T
$[67.5°, 112.5°)$	000T, 045T, 090T, 135T, 180T	225T, 270T, 315T
$[112.5°, 157.5°)$	045T, 090T, 135T, 180T, 225T	270T, 315T, 000T
$[157.5°, 202.5°)$	090T, 135T, 180T, 225T, 270T	000T, 045T, 315T
$[202.5°, 247.5°)$	135T, 180T, 225T, 270T, 315T	000T, 045T, 090T
$[247.5°, 292.5°)$	180T, 225T, 270T, 315T, 000T	045T, 090T, 135T
$[292.5°, 337.5°)$	225T, 270T, 315T, 000T, 045T	090T, 135T, 180T

In Table 3.1, θ represents the angle between the line connecting the current node to the target node and the due north (positive y-axis) direction.

3.3.2 Heuristic function model improvement

An important consideration in the A* algorithm is the selection of the heuristic function $H(n)$. When $H(n) = 0$, the evaluation function $f(n)$ depends solely on the actual cost $G(n)$, effectively reducing A* to Dijkstras algorithm, which performs a uniform-cost search from the start node. If $H(n)$ underestimates the true cost to the goal (i.e., $H(n) <$ actual cost), the algorithm is guaranteed to find an optimal path. However, setting $H(n)$ too

low results in the exploration of many nodes, thereby decreasing search efficiency.

In the ideal case where $H(n)$ exactly equals the true cost, the search becomes optimally efficient, rapidly converging on the shortest path. Conversely, if $H(n)$ overestimates the true cost, the search process accelerates but sacrifices optimality, potentially producing suboptimal paths. When $H(n)$ significantly exceeds $G(n)$, the evaluation function $F(n)$ is dominated by the heuristic term, causing A* to behave similarly to a breadth-first search algorithm. These observations highlight the critical role of the heuristic function in influencing both the search behavior and performance of A*.

Traditionally, the Euclidean distance is employed as the heuristic function in A*, but its computational complexity can lead to an increased number of expanded nodes and paths that exhibit unnecessary turns, ultimately reducing search efficiency. In this study, we adopt the Manhattan distance as the heuristic function for the A* algorithm, which is computed as follows:

$$D = |x_i - x_j| + |y_i - y_j| \tag{3.5}$$

3.3.3 Heuristic function weight optimization

The heuristic function $H(n)$ in the A* algorithm significantly influences its search performance. In environments containing obstacles, the algorithm often generates many redundant nodes, increasing the total number of node expansions and thereby reducing search efficiency. Static obstacles particularly impact the environmental distance informationnamely, the distances from the current node to the start and goal nodescausing them to vary dynamically during the search process. Consequently, it becomes necessary to incorporate this changing distance information to adaptively modify the heuristic function $H(n)$, mitigating the negative effects obstacles impose on the A* search.

In the conventional A* algorithm, the choice of nodes for expansion is governed by the cost function $G(n)$ and the heuristic $H(n)$, without dynamically weighting $H(n)$ based on the scale of the nodes position relative to the environment. This fixed weighting can limit the search space and reduce algorithm efficiency, as it does not adjust according to the current nodes location within the grid. Specifically, when the current node is closer to the start node and farther from the goal, increasing the weight on $H(n)$

(making it greater than the default value of one) can accelerate the search and reduce unnecessary exploration. Conversely, if the current node is nearer the goal and farther from the start, lowering the weight toward one allows for more precise path evaluation.

By dynamically adjusting the heuristic weighting in response to the relative distances between the current node, start, and goal positions, the A* algorithms search space can be effectively minimized, leading to enhanced search efficiency.

To formalize this concept, this study introduces a distance-scale parameter P that quantifies the spatial relationship between the current node and the start and goal nodes. Given that the initial node is at $S(x_s, y_s)$ and the goal node at $G(x_g, y_g)$, the distance-scale parameter P for the current node $N(x_n, y_n)$ within the grid map is defined as follows:

$$P = \frac{|(x_n - x_g)| + |(y_n - y_g)|}{|(x_s - x_g)| + |(y_s - y_g)|}, P \in [0, 1] \tag{3.6}$$

The search space of the A* algorithm should adapt according to the distance-scale parameter P. The improved A* algorithm incorporates this parameter P into the evaluation function, as shown in (3.7), adjusting the relative weights of the movement cost $G(n)$ and the heuristic estimate $H(n)$ based on P. This modification enhances the flexibility of the evaluation function $F(n)$, enabling the A* algorithm to plan paths more quickly and accurately.

To prevent the heuristic weight from becoming excessively large, a new dynamic weighting parameter is introduced in this study that leverages the environmental distance-scale parameter P. The updated evaluation function of the improved A* algorithm is defined as follows:

$$F(n) = G(n) + e^P \cdot H(n) \tag{3.7}$$

This improvement allows the A* algorithm to adaptively modify the influence of the heuristic function in response to the environmental obstacle information. From (3.5) and (3.7), it is clear that by increasing the distance-scale parameter P, the algorithm can more accurately estimate the cost associated with the current node relative to the start and goal nodes. This dynamic adjustment optimizes the search strategy, resulting in a higher operational efficiency and an improved ability to find better paths.

In this formula, the coordinates of the initial position are (x_s, y_s), and the coordinates of the target position are (x_g, y_g)

3.3.4 Redundant jump point removal strategy

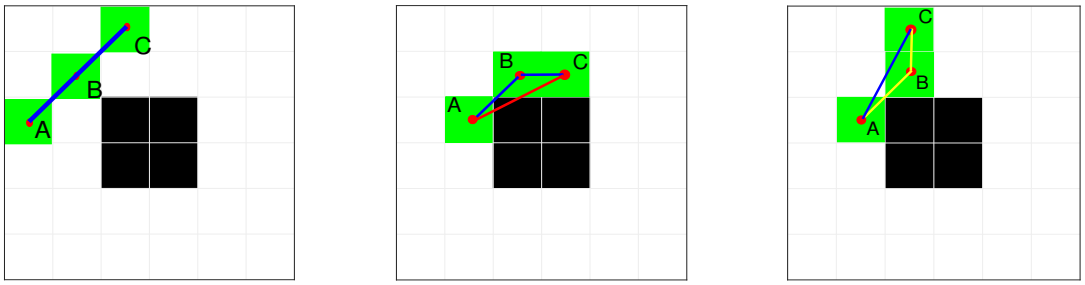
This work proposes a key node selection strategy, based on the following principles: Suppose the green nodes $\{A,B,C\}$ are three contiguous nodes on the global path, If $\lambda\overrightarrow{AB} = \lambda\overrightarrow{BC}$ is satisfied, meaning that the three nodes are collinear, then node B is considered redundant and can be removed. Consequently, path set is updated to $\{A,C\}$, as illustrated in Figure. 3.5a

If the green nodes $\{A,B,C\}$ are not collinear, that is,

$$\lambda\overrightarrow{AB} \neq \lambda\overrightarrow{BC},$$

then it must be determined whether there are obstacles between nodes A and C. If obstacles exist, node B is a key node that must be retained, and the path set remains $\{A,B,C\}$, as shown in Figure 3.5b.

If no obstacle exists between A and C, then node B is redundant and can be removed, resulting in the path set $\{A,C\}$, as illustrated in Figure 3.5c.



(a) Three nodes collinear

(b) Three nodes non-collinear with intersect with obstacle

(c) Three nodes non-collinear without intersect with obstacle

Figure 3.5: redundant nodes removing.

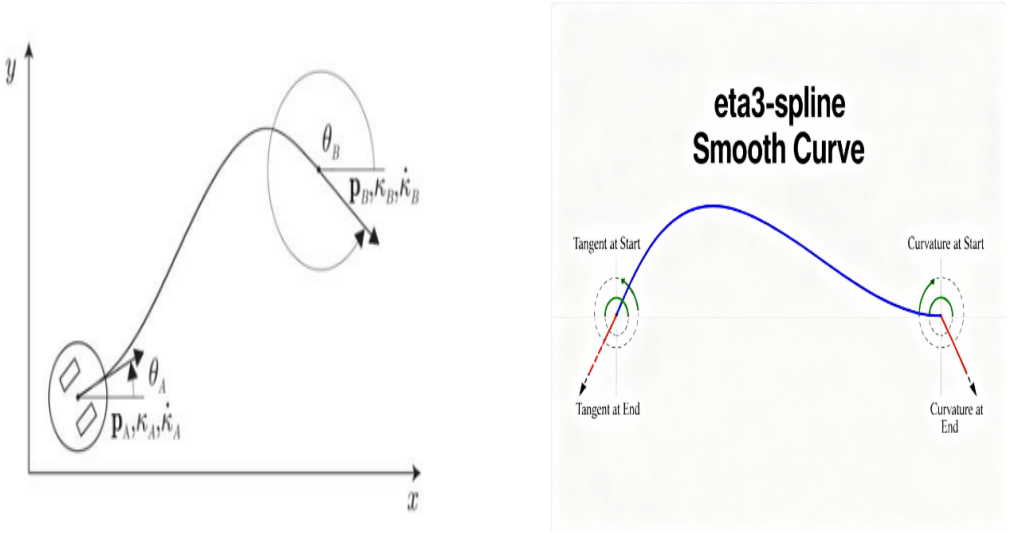
3.3.5 Path smoothing

Path smoothing using η^3-Spline: The η^3-spline is a seventh-order polynomial spline defined by:

$$\mathbf{r}(t) = \sum_{i=0}^{6} \mathbf{a}_i t^i, \quad t \in [0,1] \tag{3.8}$$

with boundary conditions on position, robot orientation, curvature, and curvature derivative:

η^3-splines generate trajectories with continuous curvature and its derivatives, ensuring smooth, feasible motion for wheeled mobile robots. The shaping parameter η offers precise control over path curvature, enabling adaptation to diverse navigation and dynamic constraints.



(a) **(b)**

Figure 3.6: Illustrations of η^3-spline path smoothing.

Given any two sequential points $\mathbf{A} = [X_A\ Y_A\ \theta_A\ k_A\ k'_A]$ and $\mathbf{B} = [X_B\ Y_B\ \theta_B\ k_B\ k'_B]$ (where X, Y are coordinates, θ is orientation, k is curvature, and k' is curvature rate), these points are interpolated using seventh-order polynomials. The coefficients for the X-coordinate polynomial are α_i, and for the Y-coordinate polynomial are β_i, ensuring all boundary conditions are satisfied. The initial conditions at point A are defined by the first few coefficients as follows:

$$\begin{cases} \alpha_0 = X_A \\ \alpha_1 = \eta_1 \cos(\theta_A) \\ \alpha_2 = \frac{1}{2}\eta_2 \cos(\theta_A) - \frac{1}{2}\eta_1^2 k_A \sin(\theta_A) \\ \beta_0 = Y_A \\ \beta_1 = \eta_1 \sin(\theta_A) \\ \beta_2 = \frac{1}{2}\eta_2 \sin(\theta_A) + \frac{1}{2}\eta_1^2 k_A \cos(\theta_A) \end{cases} \tag{3.9}$$

The higher-order coefficients (α_3 to α_7 and β_3 to β_7) then ensure the satisfaction of remaining boundary conditions at both points A and B, including position, orientation, curvature, and curvature rate, for a smooth

trajectory.

Path smoothing using cubic B-Spline: Considering the kinematic characteristics of the mobile robot, it is necessary to constrain the smoothness of the path after removing redundant points and using either of interpolation or approximation techniques. Approximation and interpolation are both techniques used to estimate values, but they differ in their approach and purpose. Interpolation involves constructing a function that passes exactly through a given set of known data points. In contrast, approximation seeks to find a function that closely follows the trend of the data but does not necessarily pass through every data point.
The B-spline generates a smooth path with continuous curvature by using the path points, obtained after redundancy removal by the greedy algorithm, as control points of the B-spline basis functions [10]. This approach not only allows specifying the curve order but also enables local modification of the curve shape by adjusting control points. This retains the advantages of Bezier curves while overcoming their limitation of lacking local control over the path shape [168, 170, 171].

Let there be $n+1$ control points P_i with $i = 0, 1, 2, \ldots, n$, a knot vector

$$T = [t_0, t_1, \ldots, t_{n+k+1}],$$

and a B-spline curve of order k, which can be defined as follows:

$$P(t) = \sum_{i=0}^{n} P_i B_{i,k}(t) \tag{3.10}$$

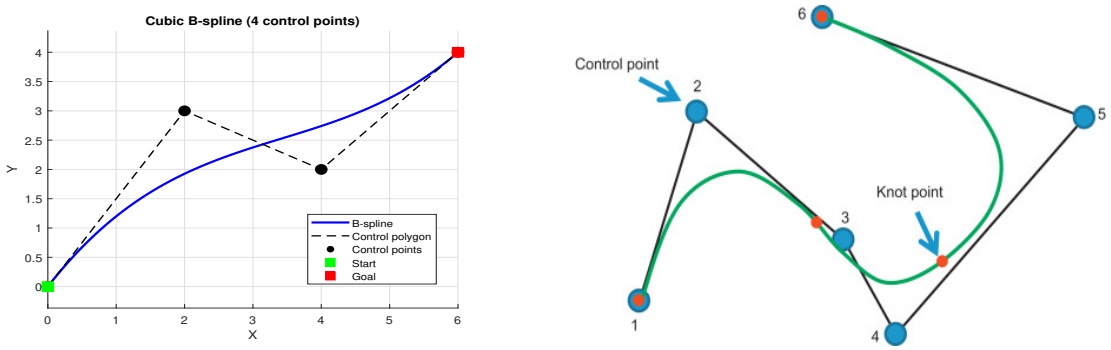
$$B_{i,k}(t) = \frac{1}{k!} \sum_{j=0}^{k-1} (-1)^j C_{k+1}^j (t+k-i-j)^k \tag{3.11}$$

where $P(t)$ denotes the B-spline curve function and $B_{i,k}(t)$ represents the k-order B-spline basis function. The path, with redundant points removed, is smoothed by constraining it with a cubic B-spline curve.

The B-spline curve can also be expressed in matrix form as follows:

$$P(t) = \frac{1}{6} \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \begin{bmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & 3 & 1 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix} \tag{3.12}$$

where P_0, P_1, P_2, and P_3 are control points. In this work, a B-spline curve is employed to smoothly fit the path after removing redundant points. The resulting smoothed path is illustrated in Figure 3.7a.



(a) Two control points **(b)** Four control points

Figure 3.7: Path smoothing using B-spline curve.

The main steps of the Efficient A-star global path as illustrated in Algorithm 3.2

Algorithm 3.2 Efficient A* global path planning

Input: Start s, goal g, grid map **Output:** Smoothed path P_{smooth}
Init open list with s, $g(s) \leftarrow 0$, $f(s) \leftarrow g(s) + w \cdot h(s,g)$
while open list $\neq \emptyset$ **do**
  $n \leftarrow$ node with lowest $f(n)$ in open list; move n to closed list
  if $n == g$ **then** Reconstruct path $s \rightarrow g$, **break**
  end if
  for neighbor n' of n **do**
    if n' in closed list or is obstacle **then continue**
    end if
    $g' \leftarrow g(n) + cost(n,n')$
    if $n' \notin$ open list **or** $g' < g(n')$ **then**
      parent(n') $\leftarrow n$; update $g(n')$, $f(n') \leftarrow g(n') + w \cdot h_{enhanced}(n',g)$
      if $n' \notin$ open list **then** add n' to open list
      end if
    end if
  end for
end while
Apply smoothing to path $\Rightarrow P_{smooth}$; **Return** P_{smooth}

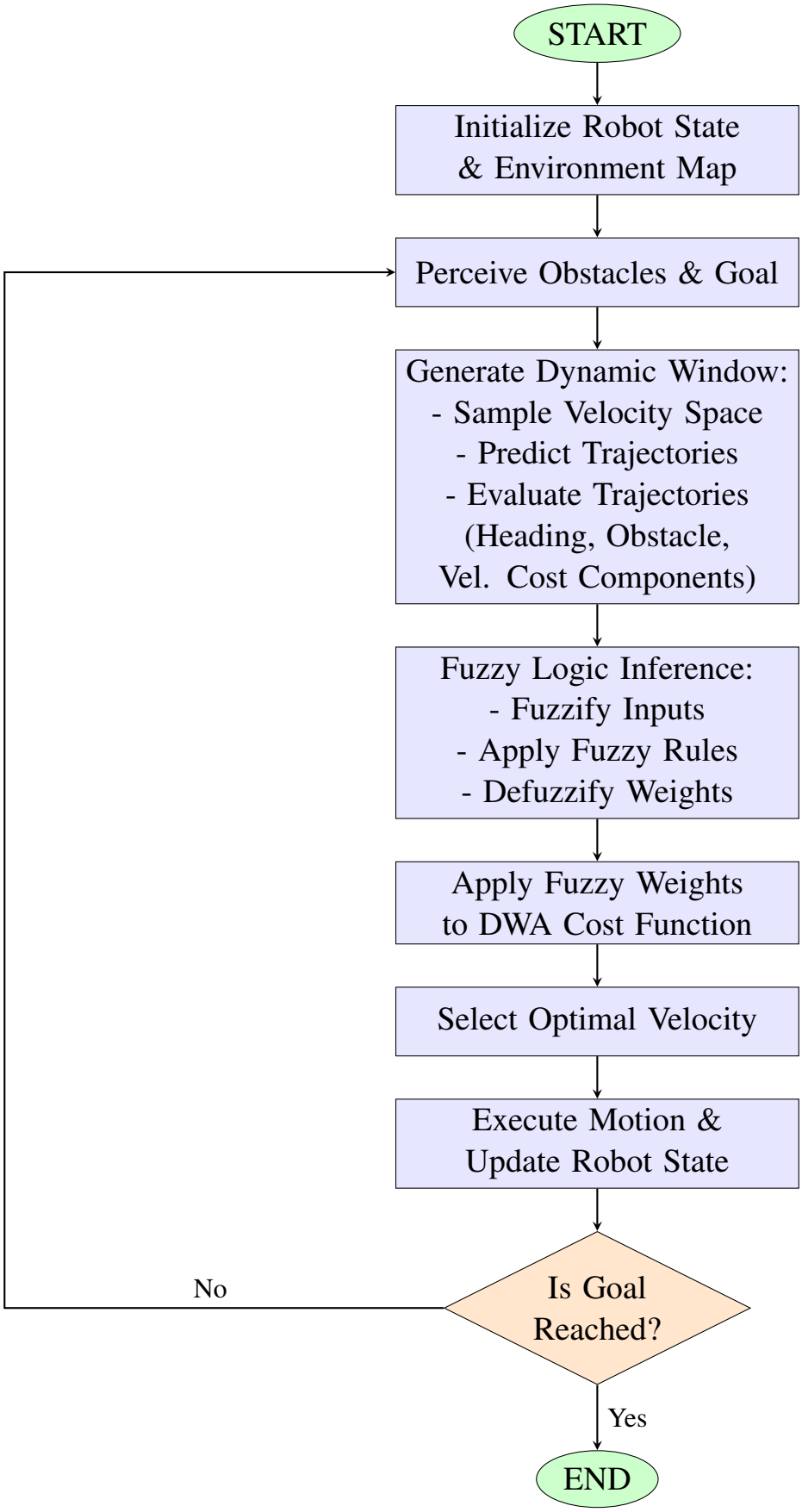


Figure 3.8: Flowchart of the Fuzzy Dynamic Window Approach (FDWA) for mobile robot local path planning.

3.4 Traditional Dynamic Window Approach

3.4.1 Kinematic model of mobile robot

A conceptual diagram of a differential-drive wheeled mobile robot is illustrated in Figure 2.1. In this figure, the coordinate frame (OX_gY_g) denotes a fixed global inertial reference, whereas (OX_lY_l) represents a local frame attached to the robots chassis. The parameter L defines the distance between the robots left and right wheels, and r corresponds to the radius of these wheels.

The robots pose within the plane is described by the tuple (x, y, θ), where (x, y) specifies the Cartesian position relative to the global frame and θ denotes the robots orientation angle with respect to the same frame. Collectively, these parameters form the state vector $\mathbf{q} = [x \quad y \quad \theta]^T$, where $[\,\cdot\,]^T$ indicates the transpose operation.

The individual angular velocities of the left and right wheels, denoted as v_L and v_R, can be translated into the robots linear velocity v and angular velocity ω by means of the following relations: [63]:

$$\begin{cases} v = \dfrac{v_R + v_L}{2} \\ \omega = \dfrac{v_R - v_L}{L} \end{cases} \tag{3.13}$$

The differential kinematic equations describing the motion of the robot are expressed as follows: [168]

$$\begin{cases} \dot{x} = v\cos\theta = \dfrac{v_R + v_L}{2}\cos\theta \\ \dot{y} = v\sin\theta = \dfrac{v_R + v_L}{2}\sin\theta \\ \dot{\theta} = \omega = \dfrac{v_R - v_L}{L} \end{cases} \tag{3.14}$$

These equations form the basis for controlling and simulating the motion of differential drive mobile robots.

3.4.2 Sampling speed space

The Dynamic Window Approach (DWA) plays a crucial role in determining the feasible ranges of linear velocity $[v_{\min}, v_{\max}]$ and angular velocity $[\omega_{\min}, \omega_{\max}]$ that the mobile robot can achieve during the current simulation interval. This determination is a prerequisite for simulating possible motion trajectories corresponding to various sampled velocity pairs (v, ω).

The selection of sampled velocities within the DWA must take into account three key factors: the physical velocity limits of the robot, the achievable velocity range constrained by the robots mechanical dynamics within a given sampling period, and the requirement to avoid collisions by allowing sufficient time and distance to decelerate safely when obstacles are detected.

In summary, the velocity sampling space in the DWA framework is bounded by the following components [168]:

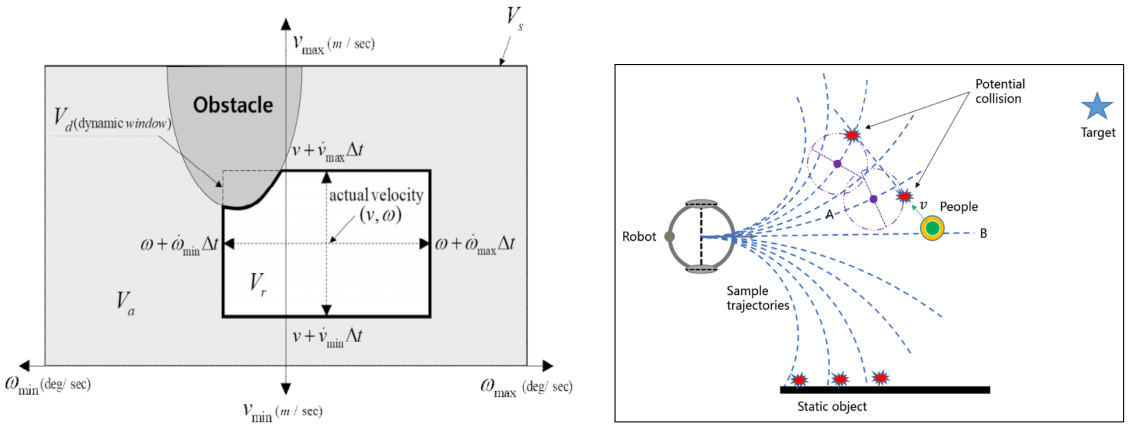
1. The absolute maximum and minimum velocities physically attainable by the robot.
2. The velocity limits attainable during the next sampling period, dictated by the robot's acceleration capabilities.
3. The velocity constraints necessary to ensure the robot can stop in time to avoid collisions with detected obstacles.

Thus, the Dynamic Window Approach is formally characterized by three constraints expressed as:

$$
\begin{aligned}
V_{\mathrm{s}} &= \{\ (v,\omega) \mid v \in [v_{\min}, v_{\max}], \omega \in [\omega_{\min}, \omega_{\max}]\} \\
V_{\mathrm{a}} &= \left\{ (v,\omega) \;\middle|\; \begin{array}{l} v \le \sqrt{2 \cdot \mathrm{dist}(v,\omega) \cdot \dot{v}_b} \\ \omega \le \sqrt{2 \cdot \mathrm{dist}(v,\omega) \cdot \dot{\omega}_b} \end{array} \right\} \\
V_{\mathrm{d}} &= \left\{ (v,\omega) \;\middle|\; \begin{array}{l} v \in [v_c - \dot{v}_b \cdot \Delta t, v_c + \dot{v}_a \cdot \Delta t] \\ \omega \in [w_c - \dot{w}_b \cdot \Delta t, \omega_c + \dot{\omega}_a \cdot \Delta t] \end{array} \right\}
\end{aligned}
\tag{3.15}
$$

- **Kinematic constraints:** These constraints define the inherent velocity limits of the mobile robot, representing the minimum and maximum linear and angular speeds the robot can achieve. Denoted as the set V_{s} in (3.15), the parameters $v_{\min}$ and $v_{\max}$ correspond to the minimum and maximum linear velocities, while $\omega_{\min}$ and $\omega_{\max}$ represent the minimum and maximum angular velocities, respectively [93].

- **Dynamic constraints:** These constraints stem from the physical limitations of the robot's actuators, specifically the maximum acceleration and deceleration capabilities for both linear and angular motions. The velocity set V_{d} in (3.15) accounts for these limits within a single sampling period. Here, v_{c} and ω_{c} denote the current linear and



(a) Velocity spaces of the DWA algorithm.

(b) Schematic diagram of the DWA method.

Figure 3.9: Dynamic Window Approach (DWA).

angular velocities, v_a and v_b indicate the maximum linear acceleration and deceleration, and ω_a and ω_b are the corresponding angular acceleration and deceleration bounds.

- **Safety constraints:** To prevent collisions, the sampled velocities must ensure that the robot can stop safely before encountering an obstacle. The set of velocities V_a in (3.15) incorporates these braking distance requirements. Specifically, dist(v, ω) represents the distance from the endpoint of the simulated trajectorygenerated at velocity pair $(v, \omega)$to the nearest obstacle, considering the previously mentioned velocity constraints.

The final velocity selection for the robot is given by the intersection of these sets:

$$V_r = V_s \cap V_d \cap V_a \tag{3.16}$$

Figure 3.9b illustrates a simplified overview of the Dynamic Window Approach, demonstrating how the velocity sampling space integrates the kinematic space V_s, dynamic space V_d, and obstacle avoidance space V_a, as also depicted in Figure 3.9a [40].

3.4.3 Evaluation function

The evaluation function aims to select the optimal trajectory for the mobile robot by assessing candidate velocities. Different velocity inputs generate distinct predicted trajectories, which correspond to varying values

of the evaluation function. A key principle in designing this function is to maximize obstacle avoidance during local path planning, while ensuring the robot progresses toward the goal at the highest feasible speed. The evaluation function used in this work is defined as follows: [128]:

$$G(v,\omega) = \sigma(\alpha \cdot \text{Heading}(v,\omega) + \beta \cdot \text{Dist}(v,\omega) + \gamma \cdot \text{Vel}(v,\omega)) \quad (3.17)$$

Equation (3.17): Here, $\text{Heading}(v,\omega)$ represents the target heading evaluation function, which measures the angular difference between the end point of the simulated trajectory and the goal position, given the current velocity commands.

$\text{Dist}(v,\omega)$ evaluates the distance between the robot's predicted endpoint and the nearest obstacle, serving as a measure of obstacle avoidance.

$\text{Vel}(v,\omega)$ denotes the evaluation function of the robot's linear velocity for the current motion, encouraging faster movement.

The coefficients α, β, and γ are fixed weights that balance the contribution of the three sub-functions in the overall evaluation.

Lastly, δ is a normalization factor applied to ensure the three components of the evaluation function are comparably scaled, defined as follows:

$$\text{Heading}(v,\omega) = 180^\circ - \theta \quad (3.18)$$

$$\text{Dist}(v,\omega) = \begin{cases} d, d < d_{\max} \\ d_{\max}, d \geq d_{\max} \end{cases} \quad (3.19)$$

$$\text{Vel}(v,\omega) = v(t) \quad (3.20)$$

Equation (3.18) defines the heading subfunction score, where θ denotes the angular deviation between the endpoint of the simulated trajectory and the goal position under the current velocity pair. A smaller deviation corresponds to a higher score.

Equation (3.19) describes the distance subfunction score, where d represents the shortest distance between the endpoint of the simulated trajectory and the nearest obstacle, and $d_{\max}$ is the predefined maximum threshold distance.

Equation (3.20) shows the velocity subfunction score, where v is the linear velocity of the mobile robot at the current time instant.

The normalization formula in (3.21) guarantees that each component of the trajectory evaluation function is scaled to the interval $[0,1]$. Normal-

izing the individual components before their combination promotes a balanced evaluation, helping to generate smoother trajectories by preventing any single factor from disproportionately influencing the decision-making process. The normalization is computed as follows:

$$\begin{cases} \text{normal\_heading}(i) = \dfrac{\text{heading}(i)}{\sum_{i=1}^{n} \text{heading}(i)} \\ \text{normal\_dist}(i) = \dfrac{\text{dist}(i)}{\sum_{i=1}^{n} \text{dist}(i)} \\ \text{normal\_velocity}(i) = \dfrac{\text{velocity}(i)}{\sum_{i=1}^{n} \text{velocity}(i)} \end{cases} \tag{3.21}$$

Equation. (3.21), the variable n represents the total number of sampled trajectories within a single simulation cycle, while i corresponds to the specific trajectory currently under evaluation. The Dynamic Window Approach (DWA) demonstrates strong performance in real-time obstacle avoidance and online path planning by relying on locally sensed environmental data. This enables the generation of smooth motion trajectories that accurately reflect the robots actual path in dynamic settings.

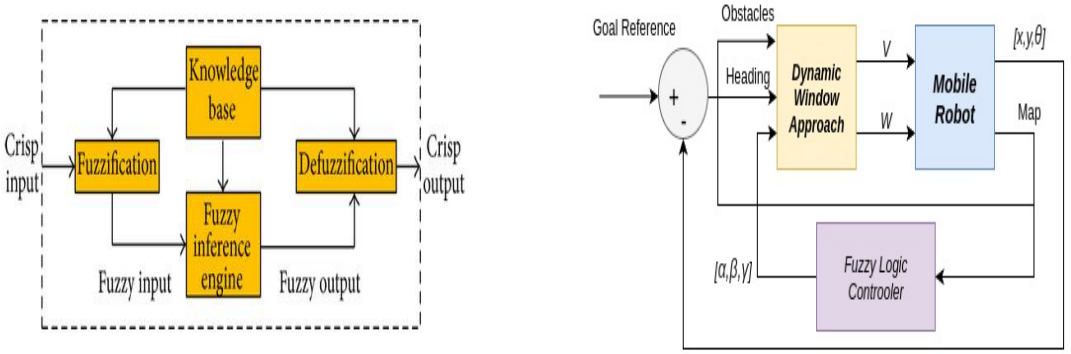
Nevertheless, a notable limitation of DWA lies in its focus on short-term sampling within a limited time window. This localized perspective restricts the algorithms ability to utilize comprehensive global information that spans from the start to the goal position. Consequently, the method is prone to becoming trapped in local minima, which can lead to suboptimal solutions when planning the global path.

3.5 Fuzzy Dynamic Window Approach

This section presents a fuzzy logic-based enhancement of the Dynamic Window Approach (DWA), where the fuzzy controller dynamically adjusts the weight coefficients in response to real-time environmental inputs. This adaptive mechanism aims to improve the algorithms robustness and flexibility across varying operational conditions.

As illustrated in Figure 3.10a, the design of the fuzzy controller the core component of fuzzy controlfollows four key steps: input fuzzification, rule base construction, fuzzy inference, and defuzzification. Among these, the formulation of the fuzzy control rules is especially critical for the controllers performance [43, 172].

Figure 3.10b illustrates the Fuzzy Dynamic Window Approach (FIDWA), a robotic navigation strategy. It depicts a closed-loop control system where



(a) Fuzzy logic controller structure

(b) Block diagram of the fuzzy dynamic window approach (FDWA)

Figure 3.10: DWA based on Fuzzy control.

a Dynamic Window Algorithm calculates optimal robot velocities (V, ω) based on perceived obstacles and a desired heading. Concurrently, a Fuzzy Logic component dynamically adjusts the algorithm's parameters ($[\alpha, \beta, \gamma]$) by leveraging map feedback, enhancing adaptability and robustness in dynamic environments.

3.5.1 Fuzzification

The fuzzy controller receives three input variables: the heading angle between the mobile robot and the goal point (`headGoal`), the distance to the nearest obstacle (`distObs`), and the robots linear velocity (`velRob`). These inputs are fuzzified as follows:

- The heading deviation `headGoal` is classified into three fuzzy sets: {Little, More, Lots}, corresponding to small (blue curve), medium (red curve), and large (orange curve) deviation angles within the domain $[0°, 180°]$. The associated membership functions are shown in Figure 3.11a.
- The obstacle distance `distObs` is divided into {Close, Medium, Far}, indicating near (blue), medium (red), and far (orange) distances within the range $[0, 15]$ meters. The membership functions appear in Figure 3.11b.
- The robots linear velocity `velRob` is grouped into {Slow, Normal, Fast}, representing slow (blue), medium (red), and fast (orange) speeds within $[0, 1]$ m/s. Its membership functions are depicted in Figure 3.11c.

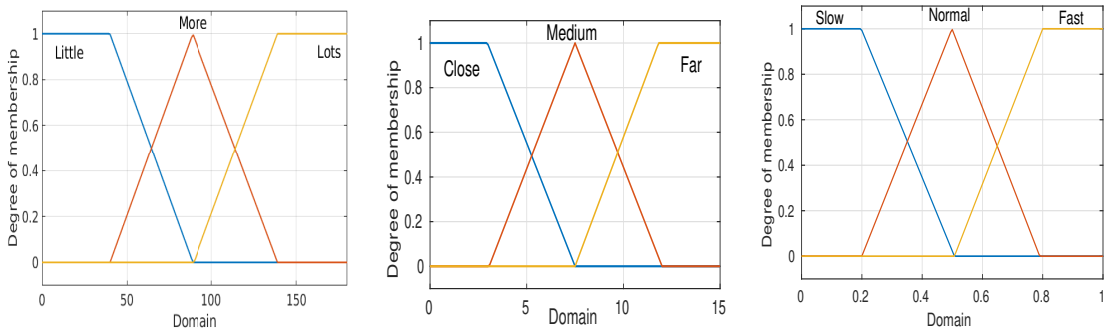
The fuzzy outputs correspond to the weight coefficients α, β, and γ, each defined with fuzzy sets {Low, Medium, High}, representing small (blue), medium (red), and large (orange) weights in the interval $[0, 1]$. Their membership functions are presented in Figure 3.12.

3.5.2 Fuzzy rule base construction

The fuzzy inference system relies on a set of control rules, which are comprehensively outlined in Table 3.2. These rules encapsulate expert knowledge to determine the appropriate weight adjustments based on varying input scenarios, thereby enhancing the DWAs adaptability to complex and dynamic environments.

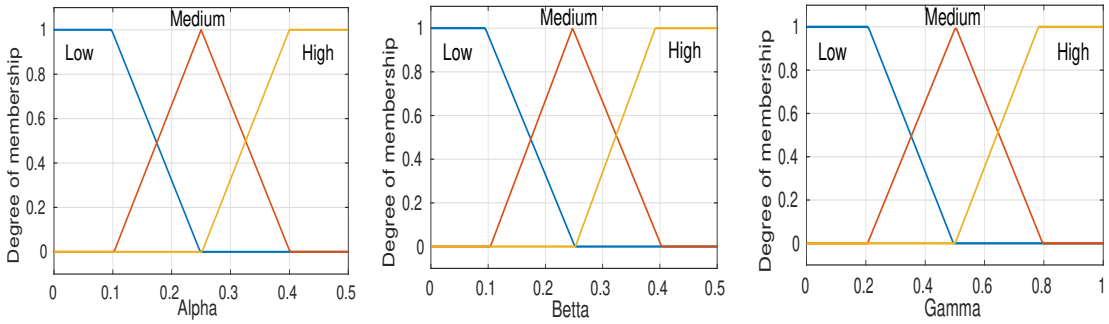
Table 3.2: Fuzzy rules of the proposed approach.

Rule	**Input**			**Output**		
	headGoal	distObs	velRob	α	β	γ
1	Little	Close	Slow	Low	High	Low
2	More	Close	Normal	Low	High	Low
3	Lots	Close	Fast	Low	High	Low
4	Little	Medium	Slow	Medium	Low	Medium
5	More	Medium	Normal	Medium	Low	Medium
6	Lots	Medium	Fast	Low	Medium	Medium
7	Little	Far	Slow	High	Low	High
8	More	Far	Normal	High	Low	High
9	Lots	Far	Fast	High	Low	High



(a) headGoal **(b)** distObs **(c)** velRob

Figure 3.11: Input membership functions.



(a) α **(b)** β **(c)** γ

Figure 3.12: Output membership functions.

3.6 Simulation Results and Discussion

3.6.1 Simulation Setup

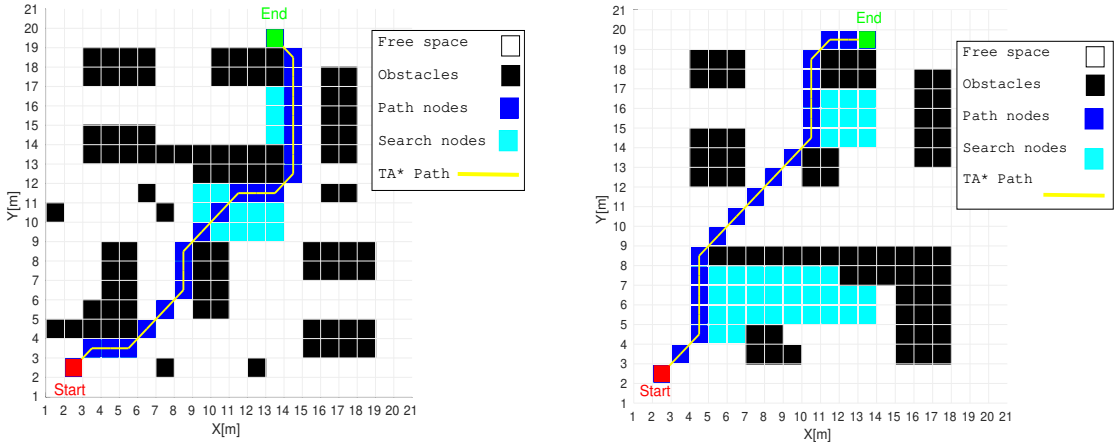
To verify the effectiveness and robustness of the proposed approach, two comparative experiments were conducted. First, the Traditional A* (TA*) algorithm was compared with the Efficient A* (EA*) algorithm in two different environments. Second, the Traditional Dynamic Window Approach (TDWA) was evaluated against the Fuzzy Dynamic Window Approach (FDWA). The simulations were executed on an Ubuntu 20.04 system using MATLAB 2020b, running on an Intel i9-12900KF CPU with 32GB RAM and a 12GB GPU.

3.6.2 Simulation Results

As illustrated in Figure 3.13, Scenarios 1 and 2 present the traditional A* approach. Here, the red square denotes the start position, the green square indicates the target, dark blue nodes form the actual path, and cyan nodes reveal the search area. The global path appears jagged (yellow line), emphasizing its lack of continuity and suitability for mobile robot navigation.

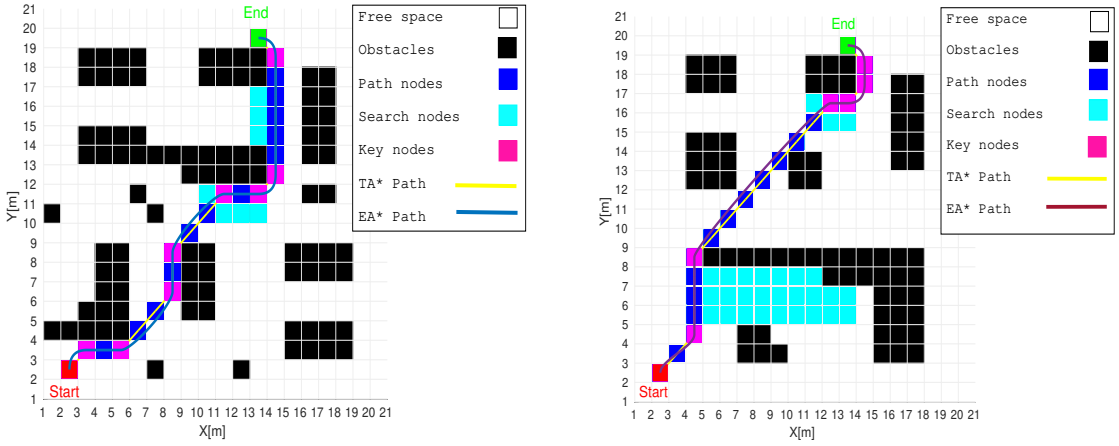
In contrast, Scenarios 1 and 2 in Figure 3.14 illustrate the results of the Efficient A* (EA*) algorithm. In these figures, the purple squares highlight key nodes along the planned path. Both the traditional global path (yellow line) and the improved global path (blue line) are depicted for comparison. The enhanced path demonstrates increased smoothness and improved safety, making it notably more suitable for the motion and kinematic constraints of mobile robots.

By comparing the results presented in Table 3.4, it can be seen that in Scenario 1, the number of path nodes and searched nodes of the efficient A* algorithm are reduced by **54.55%** and **17.14%**, respectively, while the planning time improved by **30.9%**. In Scenario 2, path nodes and searched nodes are reduced by **60%** and **16.04%**, respectively, with a small increase in path length of 1%, and planning time improved by **22.83%**.



(a) Traditional A* path in map1 **(b)** Traditional A* path in map2

Figure 3.13: Traditional A* path planning in two different environments.



(a) Efficient A* path in map 1 **(b)** Efficient A* path in map 2

Figure 3.14: Efficient A* path planning in two different environments.

To compare the environmental adaptability of the traditional DWA (TDWA) algorithm and the Fuzzy DWA (FDWA) algorithm, path planning simulation experiments were conducted in a static environment. The relevant parameter configuration is shown in Table 3.3, and the simulation experiment data are also presented in Table 3.5.

Table 3.3: Table of system parameters.

Parameter	Value	Parameter	Value
α	0.8	Maximum linear velocity	1 m/s
β	0.1	Maximum linear acceleration	0.2 m/s^2
γ	0.1	Maximum angular velocity	0.35 rad/s
Prediction time	3 s	Maximum angular acceleration	0.87 rad/s^2
Linear velocity resolution	0.1 m/s	Angular velocity resolution	0.017 rad/s

Table 3.4: Comparison of simulation results for two environments.

	Algorithm	Path Nodes	Search Nodes	Path Length (m)	Planning Time (s)	Smoothness	Safety
Scenario 1	Tradional A-star	22	35	24.73	0.0055	No	No
	Efficient A-star	**10**	**29**	24.73	**0.0038**	**Yes**	**Yes**
Scenario 2	Tradional A-star	20	56	**22.73**	0.0092	No	No
	Efficient A-star	**8**	**47**	22.97	**0.0071**	**Yes**	**Yes**

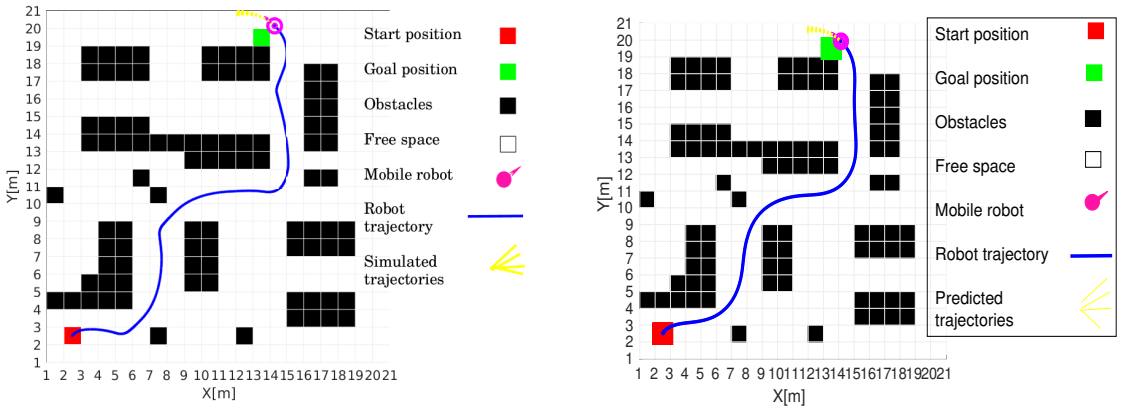
Figure 3.15a shows the local path generated by TDWA, represented by the blue curve. It can be observed that the executed trajectory lacks smoothness, contains numerous turning points, and exhibits large turning angles, which negatively impact the motion of the mobile robot.

Figure 3.15b shows the local path generated by FDWA, represented by the blue curve. It can be observed that the executed trajectory is smooth, contains fewer turning points, and has smaller turning angles, which positively impact the motion of the mobile robot.

By analyzing the results in Table 3.5 for FDWA, it can be observed that the planning time, path length, and number of iterations were reduced by **84.83%**, **6.74%**, and **57.78%**, respectively.

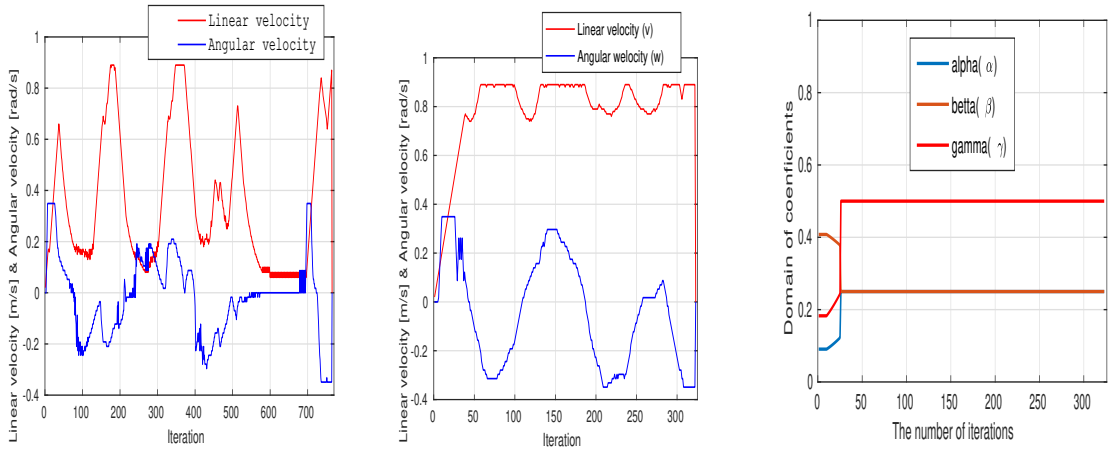
From Figure 3.16a and Figure 3.16b, it can be seen that the linear and angular velocities of the optimized DWA exhibit less variation compared to TDWA, which shows significant fluctuations in velocity.

Figure 3.16c illustrates the weight coefficients of the evaluation function in the optimized FDWA, adjusted by fuzzy logic control based on the environmental situation. Initially, the value of α is moderate, β is high, and γ is low, prioritizing obstacle avoidance and enhancing safety. Conversely, when the robots speed increases, both α and β are moderate, while γ is high.



(a) Robot trajectory in Traditional DWA **(b)** Robot trajectory in Fuzzy DWA

Figure 3.15: Comparison of robot trajectories using Traditional and Fuzzy DWA.



(a) Robot velocities in Traditional DWA **(b)** Robot velocities in Fuzzy DWA **(c)** Weight coefficients in Fuzzy DWA

Figure 3.16: Comparison of velocities and control parameters in Traditional and Fuzzy DWA.

Table 3.5: Comparison of simulation results for two environments.

	Approach	Planning Time (s)	Path Length (m)	Iteration	Success Rate (%)
Scenario 1	Tradional DWA	213.3526	27.364	765	100
	Fuzzy DWA	**32.3727**	**25.521**	**323**	100

3.7 Conclusion

This chapter presents a hybrid path planning method that integrates an efficient A* algorithm with an Fuzzy Dynamic Window Approach (DWA)

to address the limitations of traditional A*, including excessive path nodes, long planning time, unsmooth trajectories, close proximity to obstacles, and its restriction to static environments. Additionally, the proposed method aims to overcome the local minimum problem inherent in the original DWA approach. First, the efficient A* algorithm incorporates an adaptive weighting mechanism for the heuristic function, enhancing search efficiency. To further optimize the path, the Redundant jump point removal strategy is applied to remove redundant points, and B-spline curves are used for path smoothing. Furthermore, a fuzzy control system is introduced to dynamically optimize the weights of the DWA evaluation function. Finally, the adaptive A* algorithm and the optimized DWA algorithm are integrated to achieve a more efficient and reliable path planning strategy. Simulation results demonstrate that the adaptive A* algorithm reduces planning time by **30.9%**, the number of path nodes by **54.55%**, and the number of searched nodes by **17.14%** compared to the traditional A* algorithm. Similarly, the optimized DWA reduces path length by **6.74%** and planning time by **84.83%** compared to the original DWA approach. These improvements significantly enhance search efficiency and path smoothness while effectively addressing the local minimum problem. Overall, this hybrid approach provides a robust and efficient solution for autonomous mobile robot navigation, with promising applications in real-world robotics. As future work, we plan to incorporate unknown and dynamic obstacles into the environment and validate our results through real-world experiments using a physical robot under the Robot Operating System (ROS). In the next chapter, we will introduce a fusion path planning algorithm that combines the A* algorithm for global path planning with the Dynamic Window Approach (DWA) for local path planning, targeting navigation in dynamic and unknown environments.

4

A Dynamic Path Planning of Mobile Robot Based on Fusion of Enhanced A* Algorithm and Improved Dynamic Window Approach

This chapter presents a dynamic path planning method for mobile robots by fusing an enhanced A algorithm with an improved Dynamic Window Approach (DWA). The enhanced A* ensures efficient global path generation, while the improved DWA enables real-time local obstacle avoidance. Simulation results demonstrate improved path optimality and navigation safety in dynamic environments.*

Contents

4.1 Introduction

Navigation mobile robots in dynamic environments demands both efficient global and local path planning to ensure safe and optimal movement. The A algorithm is a reliable choice for global path planning; however, it typically overlooks the presence of dynamic obstacles. In contrast, the Dynamic Window Approach (DWA) excels at real-time local obstacle avoidance but may produce suboptimal trajectories. This chapter presents a fusion strategy that integrates an Enhanced A algorithm for global path generation with an Improved DWA for local navigation. The proposed approach enables more efficient, responsive, and adaptive robot navigation in dynamic settings.

4.2 Global path planning based on Enhanced A* algorithm

4.2.1 Traditional A* algorithm

The traditional A algorithm is a heuristic search method developed as an extension of Dijkstras algorithm. Unlike the blind search approach used in Dijkstras method, A introduces a heuristic function to guide the search process, thereby improving efficiency by providing direction during path exploration. With the heuristic integrated, the algorithm evaluates the neighboring nodes of the current position using a cost function, selecting the node with the lowest total estimated cost as the next point to explore. This process continues iteratively, expanding nodes toward the goal until the target is reached [44]. The total cost function used in the traditional A* algorithm is calculated as follows:

$$F(n) = G(n) + H(n) \tag{4.1}$$

In the formula, $f(n)$ is the estimated value from the starting point to the target point, $g(n)$ is the actual value from the starting point to the node n, $h(n)$ is the estimated value from node n to the target point. In the paper, we use the Euclidean distance for the actual cost value $g(n)$ and the Euclidean distance for the estimated cost value $h(n)$ .The calculation formulas are as follows: [44]

$$G(n) = \sqrt{(x_n - x_s)^2 + (y_n - y_s)^2} \tag{4.2}$$

$$H(n) = \sqrt{(x_g - x_n)^2 + (y_g - y_n)^2} \tag{4.3}$$

In the formula, (x_s, y_s) represents the coordinates of the start point, (x_n, y_n) represents the coordinates of the current node n, and (x_g, y_g) represents the coordinates of the target point. [44]

4.2.2 Enhanced A* algorithm

4.2.2.1 Improved heuristic function

The efficiency of the A* algorithm is primarily influenced by the design of its heuristic function $H(n)$. The choice of heuristic directly impacts both the quality of the resulting path and the computational cost of the search. Let $r(n)$ denote the actual cost from the current node to the goal. If the heuristic is set to zero, i.e., $H(n) = 0$, A* effectively behaves like Dijkstras algorithm, exploring all possible nodes to guarantee an optimal path. While this exhaustive search ensures accuracy, it is computationally inefficient due to the exploration of many irrelevant nodes.

When the heuristic underestimates the true cost ($H(n) < r(n)$), the algorithm expands a wider set of nodes, increasing the time required to find a path. In contrast, if the heuristic overestimates the true cost ($H(n) > r(n)$), the search becomes more directed toward the goal and faster, but it risks missing the optimal solution. The best performance is achieved when $H(n)$ precisely matches the actual remaining cost ($H(n) = r(n)$), allowing the algorithm to find the shortest path efficiently. To enhance A*s practical performance, it is essential to design a heuristic function that closely approximates the true cost to the goal. One strategy for achieving this is to incorporate a distance-based weighting factor and a logarithmic decay term, which dynamically adjust the heuristic during the search process to improve both speed and accuracy. The improved heuristic function $H'(n)$ is defined as follows:

$$H'(n) = \left(1 + \frac{d}{D}\right) \lg(H(n) + 1) H(n) \tag{4.4}$$

In the improved heuristic formulation, d represents the Euclidean distance from the current node n to the goal, while D is the total Euclidean distance between the start and goal points. At the beginning of the search, the ratio $\left(1 + \frac{d}{D}\right)$ is close to 2, giving higher emphasis to the heuristic.

As the robot progresses toward the goal, this factor gradually reduces, approaching a value of 1 when near the destination. Due to the nature of the logarithmic function, the term $\log(h(n)+1)$ has a higher value at the start, where $h(n)$ is relatively large. This logarithmic attenuation also decreases as $h(n)$ becomes smaller with path progression.

By integrating both the distance-based weighting factor and the logarithmic attenuation term into the heuristic function, the modified heuristic emphasizes goal-directed behavior in the early phases of planningthereby reducing unnecessary node expansion and accelerating convergence. Near the goal, the influence of the heuristic diminishes, allowing the algorithm to consider a broader set of candidate nodes, which helps in refining the path and ensuring optimality.

$$F'(n) = G(n) + H'(n) \tag{4.5}$$

4.2.2.2 Path optimization strategy

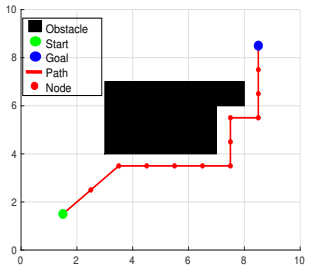
The traditional A* algorithm often produces paths containing unnecessary intermediate nodes and frequent direction changes, which negatively impact the efficiency of autonomous mobile Robots due to speed reduction during turns. To mitigate this issue, this paper introduces a post-processing strategy for path refinement. The overall optimization procedure is illustrated in Figure 4.1. The goal is to detect and eliminate redundant waypoints and non-essential turning points. Redundant nodes are identified by checking collinearity between a node and its immediate neighbors, while superfluous turns are detected by analyzing whether straight-line segments intersect obstacles. The process involves the following steps:

Step 1: Elimination of redundant nodes.
Let the initial path obtained from the enhanced A* algorithm be represented as the set $\{Q_i \mid i = 1,2,\ldots,n\}$, where Q_1 is the start and Q_n is the goal node. Beginning with the second node Q_2, check whether it lies on the same straight line as its predecessor Q_1 and successor Q_3. If collinear, Q_2 is considered redundant and can be removed. The parent of Q_3 is then updated to Q_1, and the OPEN list is refreshed accordingly. This check continues iteratively for nodes Q_3 to Q_{n-1}, producing an optimized path with fewer nodes.

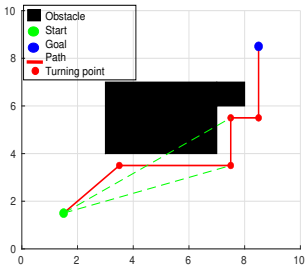
Step 2: Elimination of unnecessary turning points.

Once redundant nodes have been pruned, the resulting path, denoted as $\{N_i \mid i = 1, 2, \ldots, n\}$, undergoes further refinement. Starting from N_1, sequentially attempt to connect it directly to each of the subsequent nodes N_3, N_4, ..., N_n. For each connection N_1N_k, check whether the straight line intersects any obstacles. The first node N_k where the line N_1N_k fails the collision check indicates that the previous node N_{k-1} is a critical turning point. All intermediate nodes between N_1 and N_{k-1} are deemed redundant and are removed. The path is then updated, and the process is repeated from N_k toward N_n until the goal is reached.

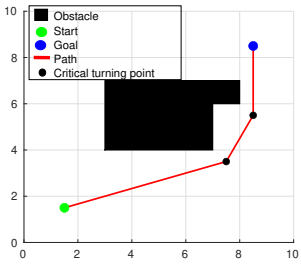
By eliminating unnecessary nodes and reducing the number of turning points in the global path, both the path length and turning angles are minimized. This simplification enhances the stability of the fusion algorithm when tracking the planned path and increases the overall efficiency of the path planning process.



(a) Original A* Path



(b) Path with Turning Points



(c) Path with Turning Points

Figure 4.1: The process of the path optimization strategy.

4.2.2.3 Smoothing the planned path

On a grid-based map, paths generated by the A* algorithm often exhibit sharp corners and lack smoothness at turning points. These abrupt transitions can lead to inefficient turning maneuvers and the accumulation of localization errors during the robot's motion. To alleviate these drawbacks, a path smoothing approach is employed.

Among the widely used smoothing techniques are B-spline and Bézier curves. B-spline curves, which generalize Bézier curves, offer better local control and flexibility due to their ability to independently adjust the number of control points and the curve order. As illustrated in Figure 4.2, B-splines provide a more refined path smoothing effect. In this work, we adopt B-spline curves to smooth the path after eliminating redundant

nodes. The general expression of a B-spline curve is given by:

$$p(t) = \sum_{i=0}^{n} P_i N_{i,k}(t) \tag{4.6}$$

where P_i $(i = 0, 1, 2, \ldots, n)$ are the control points and $N_{i,k}(t)$ denotes the k-th order B-spline basis function. The variable t is the parameter for curve evaluation, and n is the total number of control points. The B-spline basis function can be defined recursively using the de BoorCox formula:

$$N_{i,k}(t) = \begin{cases} 1 & \text{if } t_i \leq t < t_{i+1}, \quad k = 1 \\ 0 & \text{otherwise,} \\ \frac{t - t_i}{t_{i+k-1} - t_i} N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t) & \text{if } k > 1 \end{cases} \tag{4.7}$$

with the convention that $0/0 = 0$. The interval $[t_i, t_{i+k})$ defines the support for $N_{i,k}(t)$.

For practical applications, cubic or quadratic B-splines are commonly chosen due to their balance between computational cost and smoothing capability. In particular, uniform B-splines offer consistent spacing between knots and enable precise control over curve shape and length. Hence, this study utilizes cubic uniform B-spline curves to smooth the trajectory generated by the enhanced A* algorithm. The cubic uniform B-spline curve is represented as:

$$p(t) = \sum_{i=0}^{n} P_i N_{i,3}(t) \tag{4.8}$$

The basis functions for cubic uniform B-spline are explicitly defined as:

$$\begin{cases} N_{0,3}(t) = \frac{1}{6}(-t^3 + 3t^2 - 3t + 1) \\ N_{1,3}(t) = \frac{1}{6}(3t^3 - 6t^2 + 4) \\ N_{2,3}(t) = \frac{1}{6}(-3t^3 + 3t^2 + 3t + 1) \\ N_{3,3}(t) = \frac{1}{6}t^3 \end{cases} \tag{4.9}$$

where $t \in [0, 1]$.

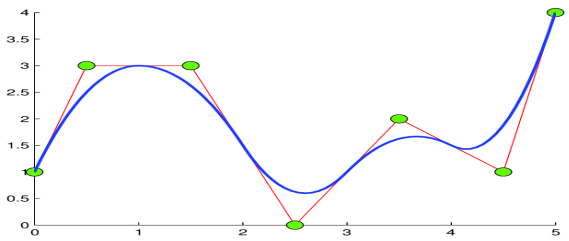


Figure 4.2: B-Spline curve.

The main steps of the Enhanced A* algorithm are defined in Algorithm 4.1.

Algorithm 4.1 Enhanced A* with safety distance and key node guidance

Input: Start s, goal g, key nodes K, grid map
Output: Smoothed path P_{smooth}
Inflate obstacles by safety radius r_{safe} to ensure clearance
Plan sequentially: $[s, k_1, ..., k_m, g]$ where $K = \{k_1, ..., k_m\}$
for each sub-goal t in $[k_1, ..., k_m, g]$ **do**
    Init open list with current start s, $g(s) \leftarrow 0$, $f(s) \leftarrow g(s) + w \cdot h(s,t)$
    while open list $\neq \emptyset$ **do**
        $n \leftarrow$ node with lowest $f(n)$; move n to closed list
        if $n == t$ **then** Reconstruct path from s to t, set $s \leftarrow t$, **break**
        end if
        for neighbor n' of n **do**
            if n' in closed list or too close to obstacle **then continue**
            end if
            $g' \leftarrow g(n) + cost(n, n')$
            if $n' \notin$ open list **or** $g' < g(n')$ **then**
                parent$(n') \leftarrow n$, update $g(n')$, $f(n') \leftarrow g(n') + w \cdot h_{enhanced}(n', t)$
                if $n' \notin$ open list **then** add n' to open list
                end if
            end if
        end for
    end while
end for
Apply smoothing to full path $\Rightarrow P_{smooth}$; **return** P_{smooth}

4.3 Local path planning based on the Improved Dynamic Window Approach

The Dynamic Window Approach (DWA) is widely used for local path planning, enabling the wheeled mobile robot (WMR) to effectively avoid both moving obstacles and previously unknown static obstacles within its environment. The implementation of the DWA involves several key steps: First, a set of candidate velocities is sampled within the feasible velocity space of the mobile robot. Second, for each sampled velocity pair, the corresponding motion trajectory of the WMR over the next time interval is simulated. Third, these candidate trajectories are evaluated using a predefined objective function, allowing the selection of the most suitable path. Finally, the velocity associated with the optimal trajectory is issued as the control command for the WMR. This process ensures real-time obstacle avoidance while guiding the robot safely and efficiently towards its goal.

4.3.1 Kinematic model of the mobile robot

The Dynamic Window Approach (DWA) relies on simulating possible trajectories over a short time horizon based on sampled velocity inputs, which necessitates an accurate kinematic model of the wheeled mobile robot (WMR). The robots motion during each sampling interval Δt can be approximated by an arc segment; however, given that Δt is very small, this arc can be effectively treated as a linear segment. Assuming constant velocity within the interval Δt, the WMRs kinematics are described by the following equations:

$$\begin{cases} x_{t+1} = x_t + v_x \Delta t \cos\theta_t - v_y \Delta t \sin\theta_t, \\ y_{t+1} = y_t + v_y \Delta t \sin\theta_t + v_x \Delta t \cos\theta_t, \\ \theta_{t+1} = \theta_t + \omega \Delta t, \end{cases} \tag{4.10}$$

where (x_t, y_t) and (x_{t+1}, y_{t+1}) denote the robot's position at times t and $t+1$, respectively. The variables θ_t and θ_{t+1} represent the heading angles at the corresponding times. The components v_x and v_y are the projections of the robots velocity along the x- and y-axes, respectively.

4.3.2 Velocity sampling

Based on the defined kinematic model, potential future trajectories can be generated by sampling linear and angular velocities. Given the large range of possible velocities, it is essential to restrict these samples within feasible limits imposed by the robots physical capabilities and environmental constraints.

1. The linear velocity and angular velocity constraints of mobile robot:

$$V_m = \{(v, \omega) \mid v \in [v_{\min}, v_{\max}],\ \omega \in [\omega_{\min}, \omega_{\max}]\} \tag{4.11}$$

In the formula, $v_{\min}$ and $v_{\max}$ represent the minimum and maximum allowable linear velocities of the mobile robot, respectively. Similarly, $\omega_{\min}$ and $\omega_{\max}$ denote the minimum and maximum permissible angular velocities.

2. Considering the constraints imposed by the motor capabilities and braking system during a time interval Δt, the robot's velocity must remain within a defined range, expressed as follows:

$$V_d = \{(v,\omega) \,|\, v \in [v_c - \dot{v}_a \Delta t,\, v_c + \dot{v}_b \Delta t],\ \omega \in [\omega_c - \dot{\omega}_a \Delta t,\, \omega_c + \dot{\omega}_b \Delta t]\} \tag{4.12}$$

In the formula, v_c and ω_c represent the current linear and angular velocities of the mobile robot, respectively. The parameters $\dot{v}_a$ and $\dot{\omega}_a$ denote the maximum allowable linear and angular decelerations, while $\dot{v}_b$ and $\dot{\omega}_b$ correspond to the maximum linear and angular accelerations that the robot can achieve.

3. To ensure safe navigation, it is crucial that the mobile robot can bring its velocity to zero before reaching any obstacle. This safety requirement imposes velocity constraints on the robot, which are expressed as follows:

$$V_a = \left\{ (v,\omega) \;\middle|\; v \leq \sqrt{2 \cdot dist(v,\omega) \cdot \dot{v}_b},\ \omega \leq \sqrt{2 \cdot dist(v,\omega) \cdot \dot{\omega}_b} \right\} \tag{4.13}$$

In the formula, $dist(v,\omega)$ denotes the nearest safe distance to an obstacle on the mobile robot trajectory projected at velocity (v,ω).

4.3.3 Evaluation function

As discussed in the previous section on the dynamic window approach, an effective evaluation function is essential for assessing multiple candidate trajectories generated by sampling various velocity pairs within the velocity space. The optimal trajectory is then chosen as the next movement for the autonomous guided vehicle (AGV). In this work, the evaluation function is designed to balance two key objectives: enabling the mobile robot to reach its target swiftly, while simultaneously maintaining a safe clearance from obstacles throughout its navigation. The proposed evaluation function is formulated as follows:

$$G(v,\omega) = \sigma\left(\alpha \cdot head(v,\omega) + \beta \cdot dist(v,\omega) + \gamma \cdot velocity(v,\omega)\right) \tag{4.14}$$

In the evaluation formula, the term $head(v,\omega)$ serves to steer the mobile robots movement direction, ensuring it remains aligned toward the target and minimizes deviation. The component $dist(v,\omega)$ helps the robot maintain a safe distance from obstacles to avoid collisions. Meanwhile,

$velocity(v,\omega)$ encourages the robot to move efficiently toward the target point. The coefficients α, β, and γ represent the relative weights assigned to each criterion, and σ acts as a smoothing factor.

It is important to note that the DWA algorithms local target points are derived from the global path. In this study, the evaluation function is enhanced by replacing the angle difference between the trajectorys final direction and the ultimate target point with the angle difference relative to the current intermediate target point. This modification aims to improve the responsiveness and accuracy of local path planning.

$$G(v,\omega) = \sigma\left[\alpha PHead(v,\omega) + \beta dist_s(v,\omega) + \gamma vel(v,\omega) + \delta dist_d(v,\omega)\right] \tag{4.15}$$

where $PHead(v,\omega)$ is the angular difference between the end direction of the trajectory and the current target point. $dist_s(v,\omega)$, $dist_d(v,\omega)$ distance between the end of the simulated trajectory and the nearest static and dynamic obstacles, respectively.

4.4 Fusion of Enhanced A* algorithm and Improved Dynamic Window Approach

The improved A* algorithm is capable of generating a globally optimal path for the mobile robot; however, it is not designed to handle dynamic obstacles or unknown static obstacles that may appear in the environment. Conversely, while the Dynamic Window Approach (DWA) is not suitable for global path planning, it excels at real-time local obstacle avoidance. To leverage the strengths of both methods, this work proposes a hybrid approach that combines the improved A* algorithm for global path planning with the DWA for dynamic obstacle avoidance. The integration procedure is detailed as follows: Planning global paths by improving the A* algorithm, Optimizing global path using the path optimization strategy, The critical turning points in the globally optimal path are made as temporary target points for the dynamic window method, Local path planning between critical turning points using the dynamic window method, Generate a global optimal path. The flowchart of fusion algorithm is shown in Figure 4.3.

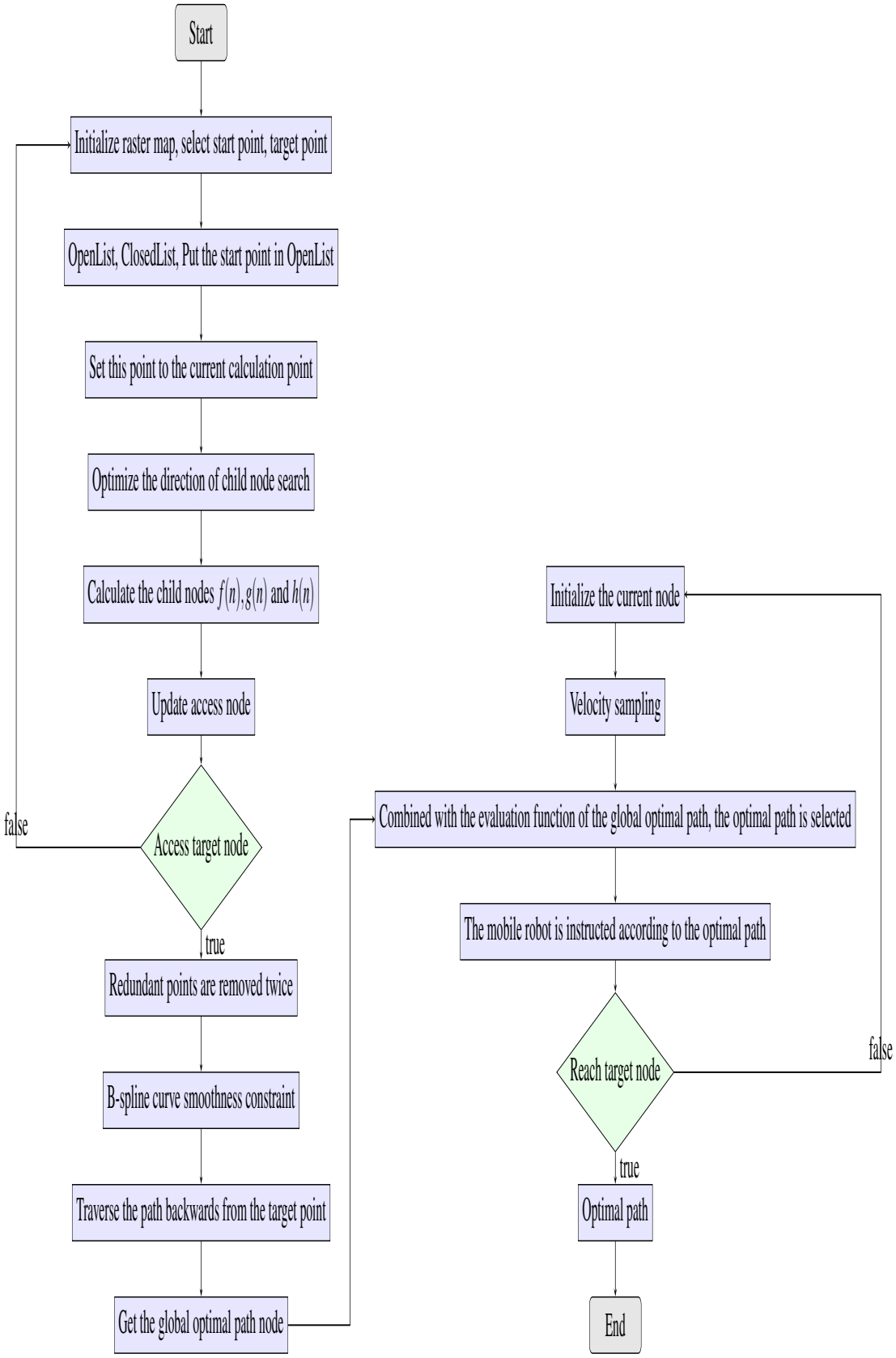
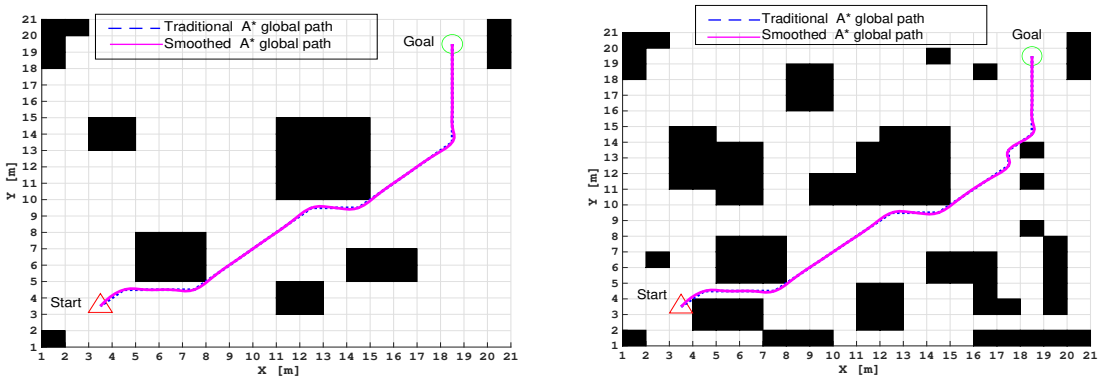


Figure 4.3: Fusion algorithm flow chart.

4.5 Simulation Results and Discussion

To evaluate the feasibility and performance of the improved algorithm proposed in this study, simulations were conducted in the following computing environment: *MATLAB R2020b* running on *Linux Ubuntu 20.04*, with an Intel i9-12900KF processor, 32 GB of RAM, and a 12 GB Nvidia GeForce graphics card.

The Differential Wheeled Mobile Robot (DWMR) was tested within a simulated 20×20 grid map, illustrated in Figures 4.4a and 4.4b. Each grid cell represents a 1-meter square, where black cells denote obstacles and white cells indicate free navigable space for the DWMR. The robot's starting position was set at coordinates (3, 3), while the goal location was fixed at (18, 19). The parameters utilized for the Dynamic Window Approach (DWA) during these simulations are detailed in Table 4.1.



(a) Map 1 (20x20) **(b)** Map 2 (20x20)

Figure 4.4: The global path generated by Traditional A*.

Table 4.1: Table of system parameters.

Parameter	Value	Parameter	Value
α	0.05	Maximum linear velocity	1 m/s
β	0.2	Maximum linear acceleration	0.2 m/s^2
γ	0.3	Maximum angular velocity	0.35 rad/s
Prediction time	3 s	Maximum angular acceleration	0.87 rad/s^2
Linear velocity resolution	0.1 m/s	Angular velocity resolution	0.017 rad/s

The traditional A-Star has a large search scope, a long search time, and low safety around obstacles, resulting in inefficient path planning. The simulation results, as shown in Figure 4.4, where the black grids and white

grids represents the obstacles and free space respectively, the red triangle and green circle represent the start and goal points, respectively,

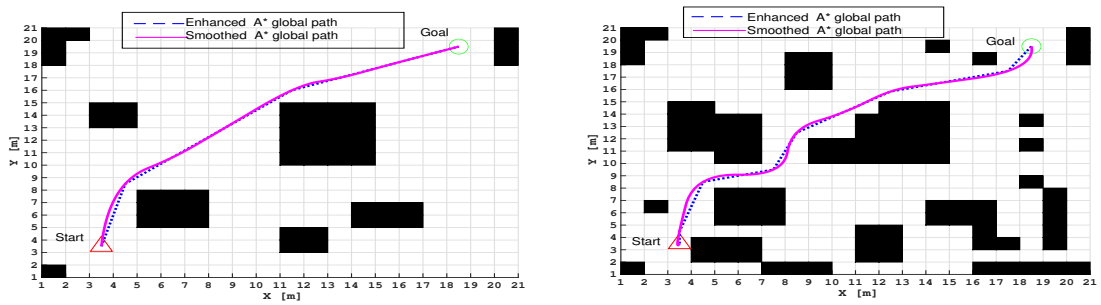
Figure 4.4a illustrates the path planning results of the traditional A-Star algorithm in map 1, with the initial global path depicted by the dashed blue line. According to the simulation experiments, the planning time for the traditional A-Star algorithm in this scenario was 0.0064 s. The length of the planned path was measured to be 25.14 m. The magenta line represents the result of smoothing the initial A-Star path using a B-spline. While the B-spline smoothing generates a visually smoother path, it is considered an inefficient path for actual mobile robot motion due to a lack of consideration for the robot's kinematic constraints and physical dimensions. This discrepancy between a geometrically smooth curve and a physically executable trajectory highlights a key limitation of simply applying post-processing smoothing without integrating the robot's motion constraints into the planning process.

Figure 4.4b illustrates the path generated by the traditional A-Star algorithm in map 2 (dashed blue line), which took 0.0033 s to compute and resulted in a path length of 25.14 m. Although B-spline smoothing yields a visually smoother path (magenta line), this approach is ultimately inefficient for mobile robots as it overlooks crucial kinematic constraints and physical dimensions.

The enhancements introduced in the heuristic function and the resulting benefits of the improved A* algorithm are demonstrated through comparative simulation experiments, as depicted in Figure 4.5.

Figure 4.5a presents the path planning outcomes of the Enhanced A* algorithm in map 1. The initial global path, indicated by the dashed blue line, represents the raw route generated by the algorithm. The planning process in this scenario required 0.0208 seconds, producing a path length of 23.18 meters. Following this, the smoothed path shown as a solid magenta lineexhibits a visually more continuous trajectory, which likely enhances safety and feasibility for real-world robotic navigation. The smoothing procedure helps to minimize abrupt turns and sudden directional changes, thereby contributing to improved stability and control for the physical robot.

Similarly, Figure 4.5b illustrates the Enhanced A* path planning results on map 2. Here, the initial global path (dashed blue line) was generated in 0.0118 seconds, with a corresponding path length of 24.16 meters. The smoothed path (solid magenta line) again offers a smoother and more practical trajectory for the robot, reducing sharp angles and promoting safer navigation.



(a) Map 1 (20x20) **(b)** Map 2 (20x20)

Figure 4.5: The global path generated by Enhanced A*.

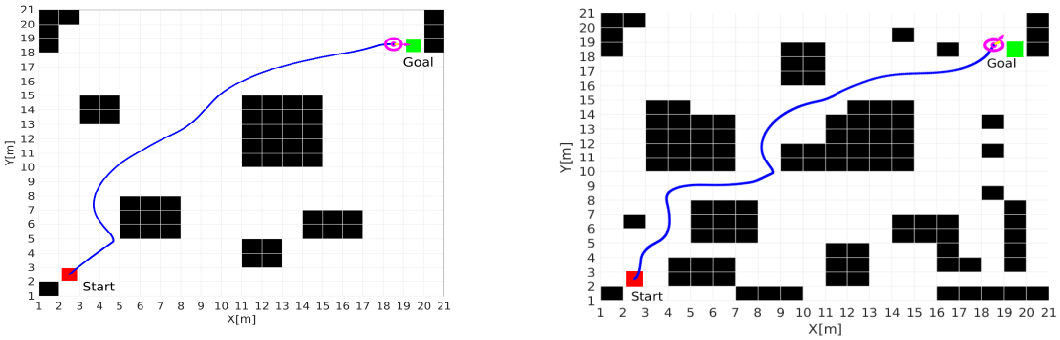
Table 4.2 summarizes the experimental results for static global path planning across both maps, highlighting an improvement in search efficiency of approximately **7.8%** and **3.9%** in path length, respectively, for the Enhanced A* algorithm.

Table 4.2: Comparison of simulation results for two environments.

	Algorithm	Path Length (m)	Planning Time (s)	Smoothness	Safety
Map 1 (20x20)	Tradional A-star	25.14	**0.0064**	Low	Low
	Enhanced A-star	**23.18**	0.0208	**High**	**High**
Map 2 (20x20)	Tradional A-star	25.14	**0.0033**	Low	Low
	Enhanced A-star	**24.16**	0.0118	**High**	**High**

The conventional Dynamic Window Approach (DWA) method exhibits several limitations, such as extended path length and prolonged execution time, as illustrated in Figure 4.6. Figure 4.6a depicts the traditional DWA trajectory on map 1. Initially, the robot moves well aligned with the goal due to a high weighting factor α, which emphasizes goal-directed motion. However, when encountering obstacles, the robot adjusts its path to avoid collisions. After navigating around the obstacles, the robot gradually reorients towards the goal while continuously evading surrounding impediments. Although the robot successfully reaches the target, the total traversal time is relatively long, taking 172.70 seconds with a path length of 25.58 meters.

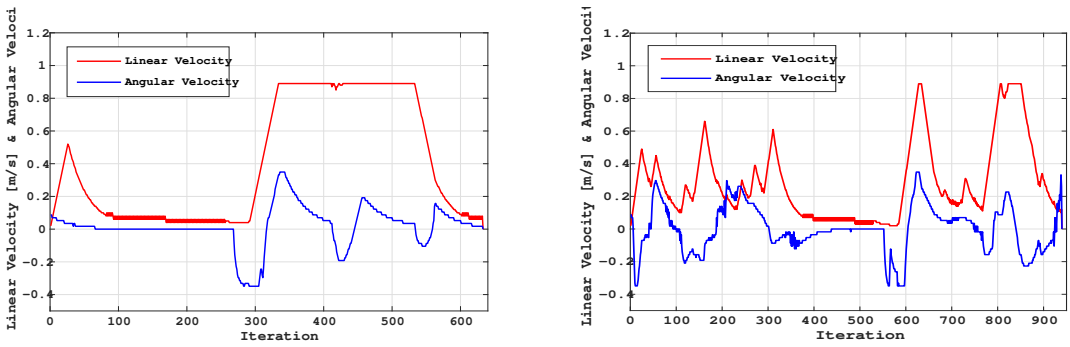
Similarly, Figure 4.6b shows the traditional DWA trajectory on map 2. Here, the robot tends to maintain a greater distance from obstacles, influenced by a larger weighting factor β that prioritizes obstacle avoidance. This leads to an increase in both path length and planning duration. As the robot encounters obstacles, it adjusts its heading accordingly and then



(a) Map 1 (20x20) **(b)** Map 2 (20x20)

Figure 4.6: The local path generated by traditional DWA in different maps.

gradually returns toward the goal direction while continuously avoiding obstacles. This behavior results in a completion time of 277.11 seconds and a total path length of 26.97 meters.



(a) Map 1(20x20) **(b)** Map 2 (20x20)

Figure 4.7: The linear and angular velocity of TDWA.

In both map 1 and map 2, abrupt changes in linear and angular velocities occur, causing the unsmooth paths generated by the traditional Dynamic Window Approach (DWA), as shown in Figure 4.7.

To further evaluate the obstacle avoidance capability and environmental adaptability of the proposed fusion algorithm, dynamic obstacle avoidance simulations were conducted in different map environments. Figures 4.8 and 4.9 illustrate the path planning results of the fusion algorithm in map 1 and map 2 respectively. In these simulations, besides the known static obstacles, additional dynamic and unknown static obstacles were introduced along potential paths to test the fusion algorithm's robustness. Black squares represent known static obstacles, yellow squares indicate dynamic obstacles, gray squares denote unknown static obstacles, the blue dotted line corresponds to the global optimal path generated by the improved A*

algorithm, and the red solid line represents the path planned by the fusion algorithm.

The path planning process on map 1 is detailed in Figure 4.8. Initially, as shown in Figure 4.8a, the fusion algorithm plans along the global optimal path. Subsequently, in Figure 4.8b, the robot follows the first subgoal. When dynamic obstacles are detected, as depicted in Figure 4.8c, the robot initiates local path planning to successfully avoid these obstacles and then resumes tracking the global path. Figures 4.8d, 4.8e, and 4.8f show the robot following successive subgoals. Figures 4.8g and 4.8h demonstrate local path planning triggered by encounters with unknown static obstacles, which the robot successfully avoids before continuing along the global optimal route. Finally, Figure 4.8h confirms that the robot reaches the target point safely. These results indicate that the fusion algorithm effectively handles dynamic and unknown static obstacles while maintaining adherence to the global path. The simulation reports a planning time of 58.29 seconds and a path length of 24.78 meters in map 1.

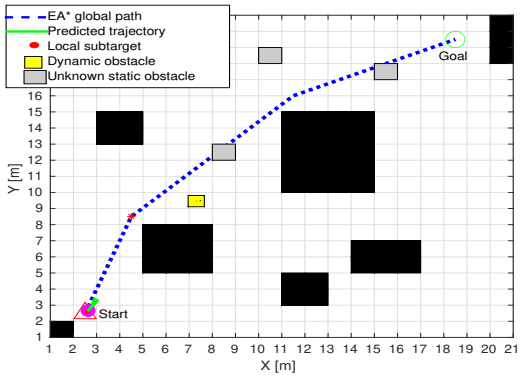
The fusion algorithms path planning on map 2 is similarly depicted in Figure 4.9, following a comparable sequence of subgoal following and obstacle avoidance maneuvers, ultimately reaching the goal with a planning time of 98.02 seconds and a path length of 25.74 meters.

To assess the effectiveness of the improved evaluation function within the DWA framework, the linear and angular velocities, as well as the robot's pose, were recorded for both the traditional and improved DWA algorithms. The improved algorithm notably reduces fluctuations in linear and angular velocities, resulting in a more stable posture. These findings are illustrated in Figure 4.10.

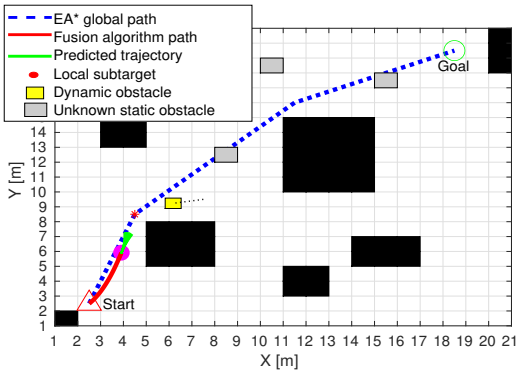
Table 4.3 presents the comparative semulation data for dynamic local path planning of the two algorithms across both scenarios. It shows improvements in path length by approximately **3.13%** and **4.57%** respectively, and reductions in planning time by **66.25%** and **64.63%** respectively.

Table 4.3: Comparison of simulation results for two environments.

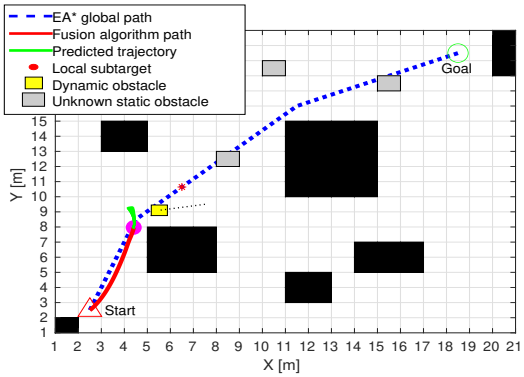
	Algorithm	**Path Length (m)**	**Planning Time (s)**	**Smoothness**	**Safety**
Map 1 (20x20)	Tradional DWA	25.58	172.6959	Low	Low
	Improved DWA	**24.78**	**58.2862**	**High**	**High**
Map 2 (20x20)	Tradional DWA	26.97	277.1121	Low	Low
	Improved DWA	**25.74**	**98.0189**	**High**	**High**



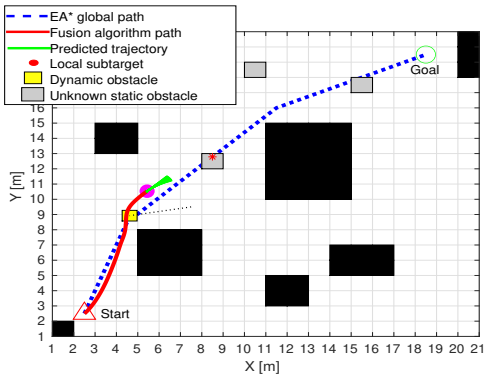
(a) ti



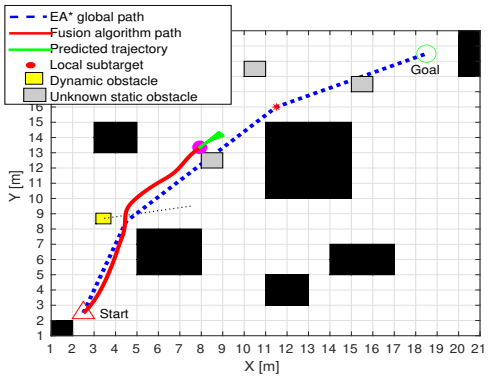
(b) t1



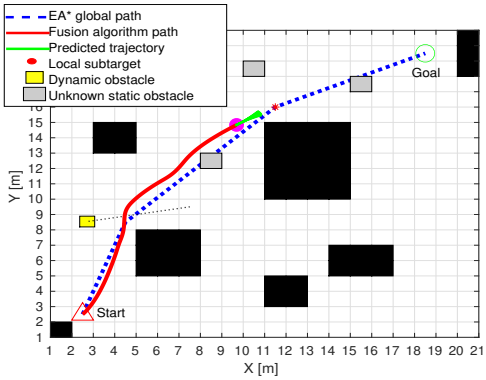
(c) t2



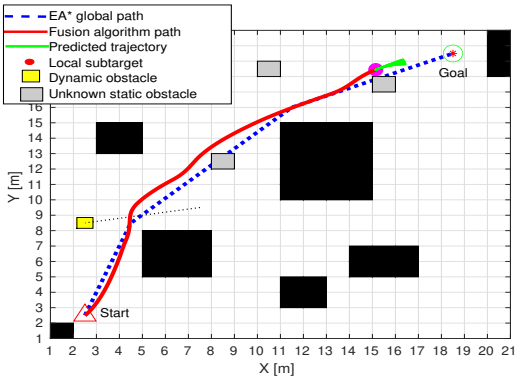
(d) t3



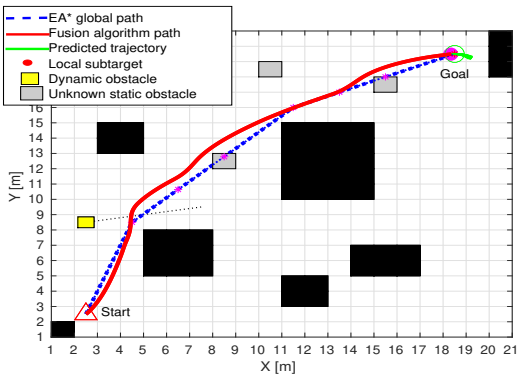
(e) t4



(f) t5

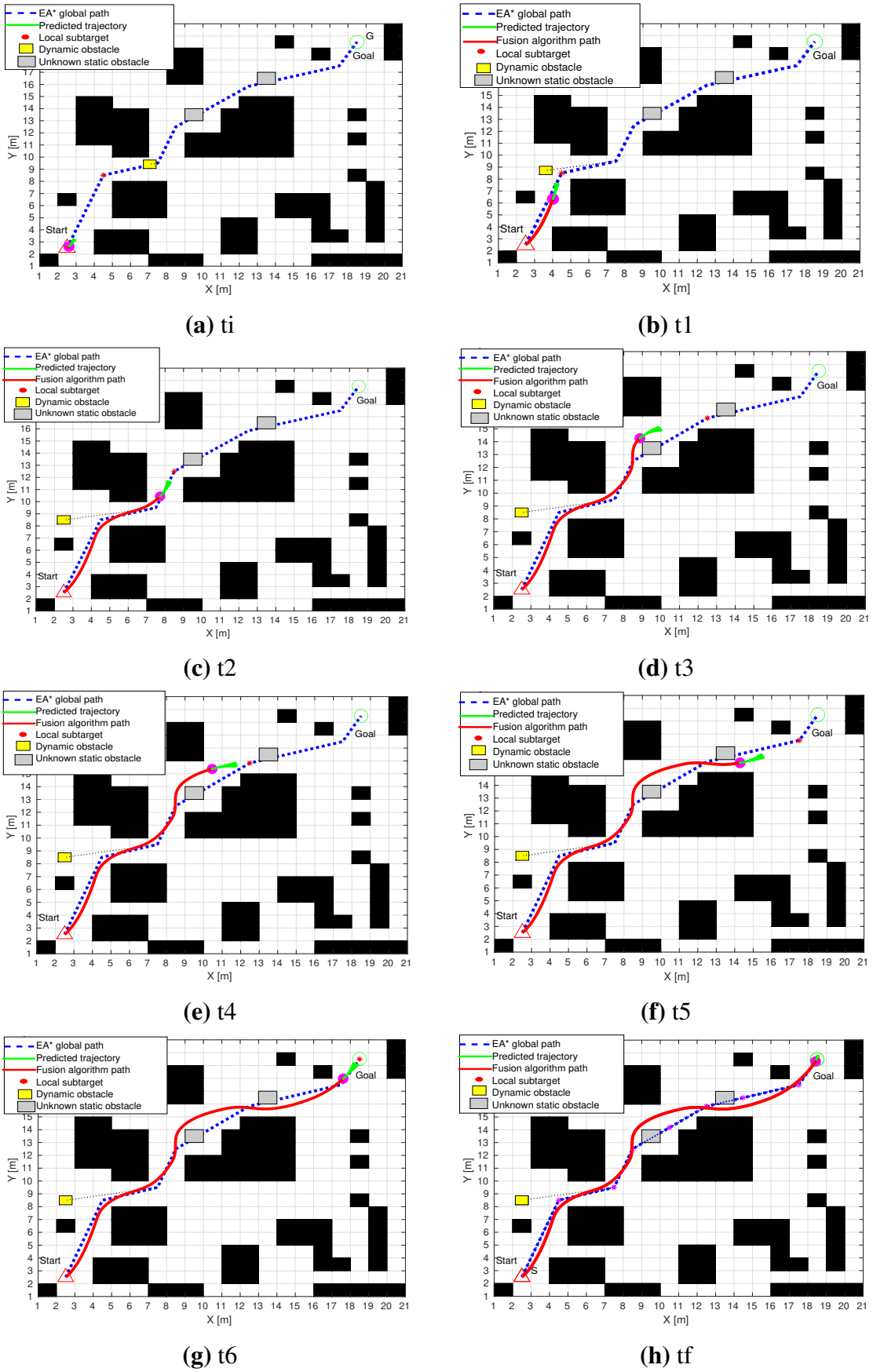


(g) t6



(h) tf

Figure 4.8: EA*-IDWA fusion algorithm trajectory in a dynamic environment in map 1 from time t_i to t_f.



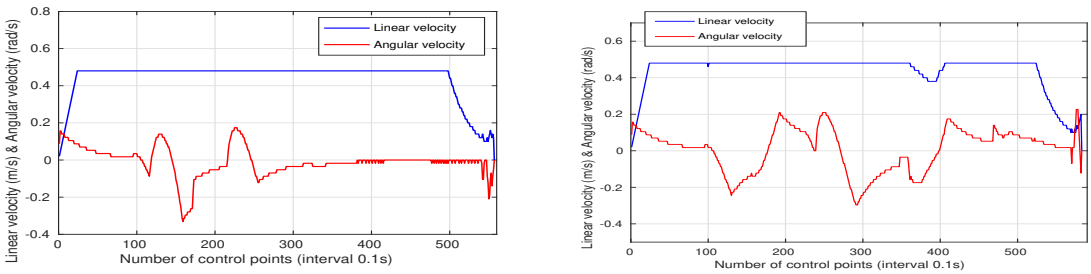
(a) ti **(b)** t1

(c) t2 **(d)** t3

(e) t4 **(f)** t5

(g) t6 **(h)** tf

Figure 4.9: EA*-IDWA Fusion algorithm trajectory in a dynamic environment in map 2 from time t_i to t_f.



(a) Map 1 (20x20) **(b)** Map 2 (20x20)

Figure 4.10: Analysis of IDWA velocity profiles across different maps.

4.6 Conclusion

This chapter introduces a novel dynamic path planning framework for mobile robots by combining an Enhanced A* algorithm with an Improved Dynamic Window Approach (DWA). The Enhanced A* algorithm builds upon the conventional A* by integrating more adaptive heuristic functions that enhance global pathfinding efficiency. Moreover, the dynamic adjustment of the cost function enables the algorithm to better accommodate changes in the environment, resulting in more flexible and effective global route planning compared to the classic A* approach. For local motion control, the Improved DWA is incorporated to facilitate real-time obstacle avoidance, allowing the robot to adjust its trajectory in response to nearby obstacles while following the global path derived from the Enhanced A*. The improvements in the DWA include the adoption of refined evaluation metrics and dynamic tuning of parameters, which significantly improve navigation performance in cluttered and complex surroundings. The integration of these two algorithms yields a well-balanced navigation strategy that merges efficient global path planning with responsive local obstacle avoidance. Experimental outcomes indicate that this approach enables the robot to navigate dynamically changing environments with enhanced efficiency, safety, and adaptability. Future work will aim to expand this method to handle dynamic obstacles and more complex environmental factors, as well as to optimize its real-time computational efficiency for practical deployment scenarios. In the next chapter, we will introduce a hybrid control technique that combines the Dynamic Feedback Linearization approach for trajectory tracking with the Fuzzy Logic concepts for online obstacle avoidance, evaluate and tested in structured and indoor environments.

5

Hybrid Control for Trajectory Tracking and Obstacle Avoidance in Mobile Robots Using Fuzzy Dynamic Feedback Linearization under ROS Middle-ware

This chapter presents a novel hybrid control strategy that merges Fuzzy Logic with Dynamic Feedback Linearization (DFL). This integration aims to ensures both tasks of trajectory tracking and obstacle avoidance capabilities in autonomous mobile robots. While DFL handles the precise trajectory following, the Fuzzy system is designed to manage reactive obstacle avoidance behavior. The effectiveness of this combined approach, including enhanced navigation accuracy, stability, and robust obstacle handling within structured environments, is substantiated through both simulation and experimental validations.

Contents

5.1 Introduction

Mobile robots are increasingly deployed in dynamic environments where executing complex tasks requires not only following a predefined trajectory but also adapting to unforeseen obstacles along the way. To ensure safe and efficient navigation, a key requirement is the robot's ability to temporarily deviate from its desired path to avoid collisions with static or dynamic obstacles. Once the environment is deemed safe, the robot must intelligently rejoin and continue along the original trajectory, maintaining task objectives such as timing, accuracy, and coverage. This capability is essential in applications like autonomous delivery, surveillance, and exploration, where continuous path adherence and obstacle avoidance must be balanced in real time. The aim of this chapter is investigate a feasible solution for this class of problems through a hybrid method that combines bpth concepts of fuzzy logic and dynamic feedback linearizarion. Simulation and experimental tests are presented to assess the performances of the proposed approach.

5.2 Proposed approach

A comprehensive overview of the proposed control architecture is illustrated in Figure 5.1. It integrates a robust feedback controller that ensure the task of trajectory tracking with a fuzzy logic system that allows the robot to avoid collision with obstacles through a fusion module. The proposed tracking controller is based on the concept of feedback linearization. This choice is motivated by the simplicity of design and robustness ensured by this approach [86]. As discussed in Chapter 2, DFL is primarily utilized to ensure accurate trajectory tracking by converting the robot's nonlinear kinematic model into a linear representation, which then permits the application of standard linear control techniques. A second control block is embedded in the system to manage real-time obstacles avoidance. This system processes inputs such as the relative distances and orientations between the robot and the surrounding obstacles to generate the suitable corrective angular velocity commands that enable the robot to avoid obstacles. The third block in our control strategy is the fuzzy logic based fusion process that unsure the real time update of the impact of each block in the overall strategy:

- Accurate tracking when the path is free of obstacles (DFL active),

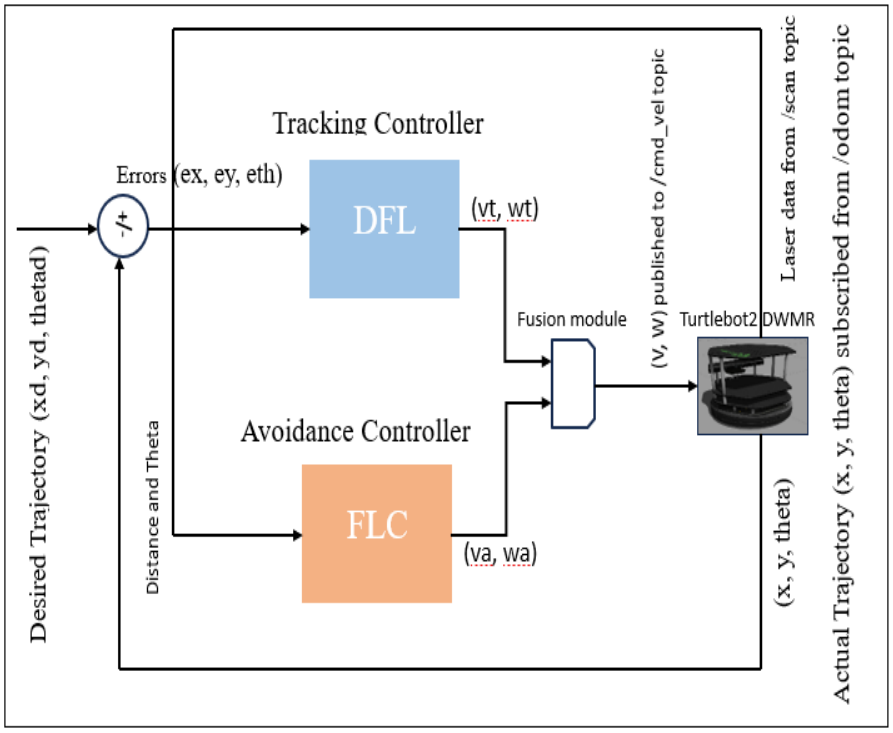


Figure 5.1: Conceptual diagram of the proposed control framework and its constituent modules within the ROS middleware.

- Smooth and reactive obstacle avoidance when hazards are detected (FLC active),
- Continuous operation without control conflict via a clean switch mechanism.

5.2.1 Trajectory tracking based on DFL concept

Considering the unicycle robot model described in Cartesian coordinates defined in (3.14), which represent the system's flat outputs and according to the development seen in section 5.5, the tracking task can be accomplished via the feedback linearizing control law formulated as:

$$\begin{cases} u_1 = \ddot{x}_d + k_{p_1}(x_d - x) + k_{d1}(\dot{x}_d - \dot{x}) \\ u_2 = \ddot{y}_d + k_{p_2}(y_d - y) + k_{d2}(\dot{y}_d - \dot{y}) \end{cases} \tag{5.1}$$

The linear and angular velocities are derived from these virtual controls, with the constraint that the linear velocity v must be nonzero, i.e., $v \neq 0$. This is expressed as follows:

$$\begin{cases} v = \int \zeta \, dt \\ \omega = \dfrac{u_2 * \cos\theta - u_1 * \sin\theta}{v} \end{cases} \tag{5.2}$$

where ζ can be expressed as follows:

$$\zeta = u_1 * \cos\theta + u_2 * \sin\theta \tag{5.3}$$

The second derivative of the robot's position, corresponding to these controls, is expressed as:

$$\ddot{F} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -v\sin(\theta) \\ \sin(\theta) & v\cos(\theta) \end{bmatrix} \begin{bmatrix} \xi \\ \omega \end{bmatrix} \tag{5.4}$$

In these equations, (x_d, y_d) signify the coordinates of the desired trajectory, while $k_{p_1}, k_{p_2}, k_{d_1}, k_{d_2}$ are the controller gain constants. Furthermore, θ denotes the mobile robot's orientation, and v and ω represent its linear and angular velocities, respectively.

The meticulous choice of control strategies is paramount for ensuring the performance and robustness of autonomous mobile robots operating in environments, whether static or dynamic. This work leverages Dynamic Feedback Linearization (DFL) for trajectory tracking, effectively transforming the robot's intrinsic nonlinear dynamics into a linear system. This transformation facilitates precise and efficient path-following capabilities in real time.

For obstacle avoidance, a Fuzzy Logic Controller (FLC) is employed. Its adoption is justified by its inherent flexibility, resilience in the face of uncertainty, and its capacity to manage intricate, dynamic scenarios. The seamless integration of the control laws defined in (5.2) and (5.4) forms a cohesive and highly robust navigation strategy. This combined approach empowers the robot to accurately adhere to its planned trajectory while concurrently performing dynamic obstacle avoidance maneuvers.

5.2.2 Obstacle avoidance

If an obstacle is detected within a frontal field of view of $\frac{pi}{4}$ radians and at a distance less than or equal to d_0, a second controller is activated to define an obstacle avoidance strategy. This strategy is based on the relative position of the obstacle: if the obstacle is on the right side, the

robot applies a leftward angular velocity to steer away; if the obstacle is on the left, it turns right to avoid it. This process can be ensured according to the following simple switching strategy:

$$\omega_a = \begin{cases} +1 \quad if \quad \theta_o \leq 0 \ (Turn\ left) \\ -1 \quad if \quad \theta_o > 0 \ (Turn\ right) \end{cases} \tag{5.5}$$

Remark 1 *It is important to note that the proposed method for obstacles avoidance behavior does not alter the tracking velocity, i.e.* $v_a = 0$.

5.2.3 Fuzzy logic based fusion process

A critical component of the proposed navigation system is its capability for effective manipulation of the developed tracking controller and the obstacles avoidance controller in the overall bahavior of the system. To achieve a seamless and adaptable transition between trajectory tracking and obstacle avoidance, a weighted fusion strategy is implemented. This strategy utilizes the output `F` from a fuzzy controller as a blending factor. The resultant control commands are expressed as:

$$\begin{cases} v = (1-F)v_t + Fv_a \\ \omega = (1-F)\omega_t + F\omega_a \end{cases} \tag{5.6}$$

where, v and ω are the final linear and angular velocity commands, that dictate the robot's final motion. The terms v_t and ω_t are the linear and angular velocities generated through the DFL controller to ensure the tracking task, while v_a and ω_a are obtained from the obstacles avoidance module to ensure safety of the robot.
Equation (5.6) the parameter F must be bounded as $0 \leq F \leq 1$. Figure 5.2 illustrates the design methodology of the fuzzy controller that is adopted to generate the value of F, which is structured around four primary blocks: inputs fuzzification, fuzzy rules formulation, fuzzy inference system, and defuzzification module.
The developed FLC incorporates two input variables and one output variable. The first input, denoted `Dobst`, quantifies the relative distance between the robot and an the nearest obstacle. The second input, denoted `Thetaobs`, indicates the relative orientation between the robot's current heading and the nearest obstacle's position. The output, designated as `F`, serves as a weighting factor within the subsequent fusion process.

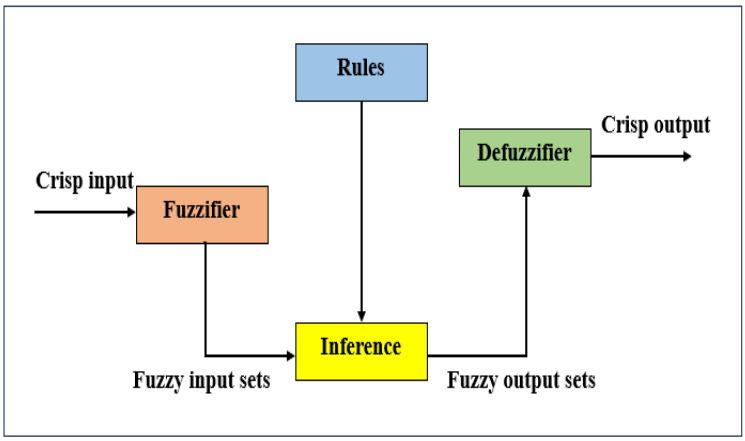
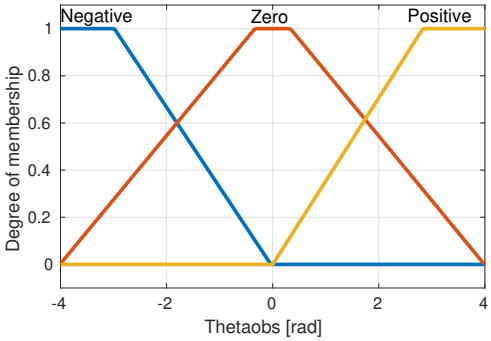
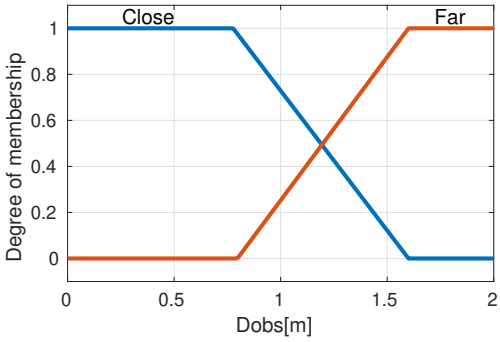


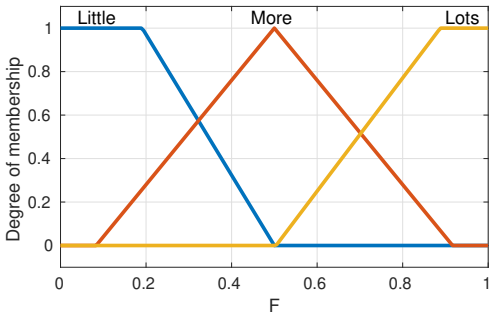
Figure 5.2: Architectural representation of the fuzzy logic controller.



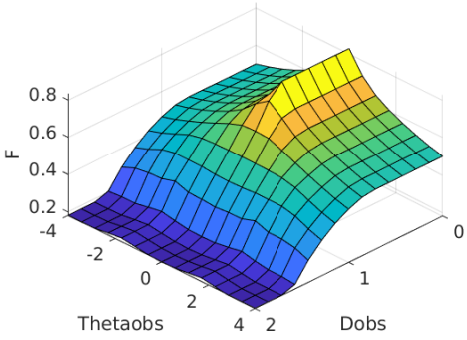
(a) Fuzzy input: `thetaObs`



(b) Fuzzy input: `distObs`



(c) Fuzzy output: `F`



(d) Fuzzy inference surface

Figure 5.3: Input and output membership functions and the fuzzy surface plot.

All fuzzy variables are defined across continuous domains, utilizing triangular and trapezoidal membership functions, as depicted in Figure 5.3. The input variable `Dobst` ranges from [0, 2] and is categorized into fuzzy sets such as "Close" and "Far," which describe the relative proximity of obstacles. The input variable `Thetaobs` spans the interval [-4, 4] and is

partitioned into sets like "Negative," "Zero," and "Positive," reflecting the obstacle's orientation relative to the robot's heading.
The output variable `F` is bounded within the domain [0, 1] and comprises the fuzzy sets "Little," "More," and "Lots," signifying small, moderate, and substantial corrective actions, respectively.

Table 5.1: Fuzzy rules table

Rule	Inputs		Output
	distObs	**thetaObs**	**F**
1	Close	Negative	More
2	Close	Zero	Lots
3	Close	Positive	More
4	Far	Negative	Little
5	Far	Zero	Little
6	Far	Positive	Little

Fuzzy control systems are built upon a foundation of inference rules, which are constructed using the previously defined membership functions. As detailed in Table 5.1, each rule establishes a mapping from fuzzy inputs to a corresponding fuzzy output. The controller dynamically determines the value of the output `F` based on the robot's real-time distance and angular relationship to nearby obstacles. The Mamdani-type fuzzy inference mechanism is employed for rule evaluation, and defuzzification is executed using the centroid method to yield a precise, crisp output value. This final output, `F`, is then critically integrated into the fusion strategy.

5.2.4 ROS (Robot Operating System)

Robot Operating System (ROS) is an open-source, flexible middleware framework designed to support the development of robot software. It provides a structured communications layer above the host operating systems of a heterogeneous set of computers, offering a set of tools, libraries, and conventions that simplify the task of creating complex and robust robot behavior. Rather than being a real-time operating system, ROS acts as a message-passing infrastructure, enabling modular and distributed design through processes called nodes, which communicate via topics, services, and actions. [173]

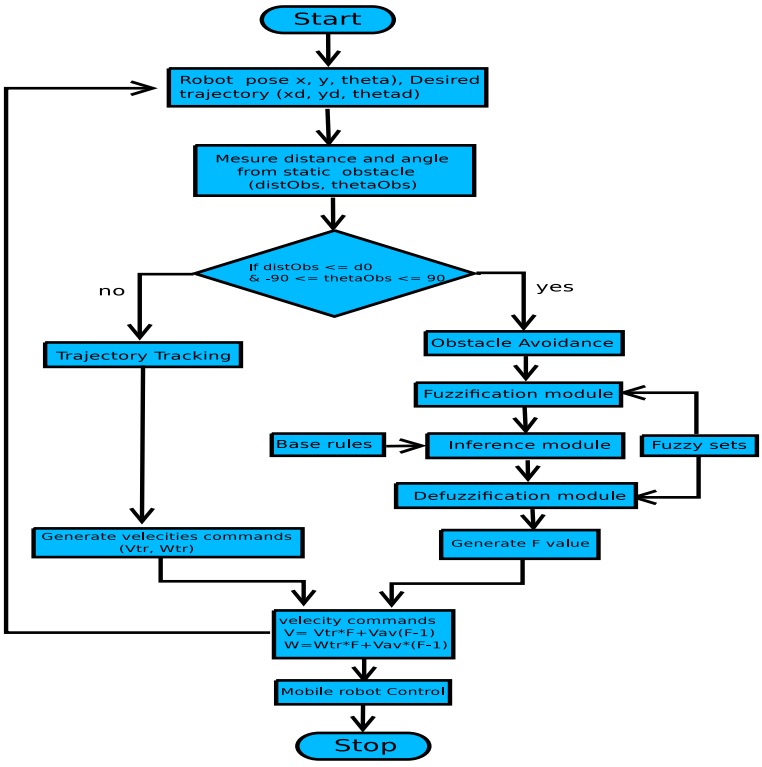


Figure 5.4: Flowchart illustrating the integrated control system, combining DFL for trajectory tracking with FLC for obstacle avoidance.

ROS promotes code reuse, hardware abstraction, and modularity, and it is widely used in both academic research and industrial robotics. Its core functionalities include hardware abstraction, low-level device control, message-passing between processes, and package management, along with simulation and visualization tools such as Gazebo and RViz.

The Robot Operating System (ROS) architecture relies on core components that enable modular, interconnected, and flexible robotic systems. These components include nodes, messages, topics, and services. Nodes are the fundamental units of computation, exchanging information via messages published and subscribed to topics, or through request/reply interactions using services [174].

Below is a detailed description of the key components of the ROS architecture:

1. **Nodes:**
   Nodes are the fundamental units of computation in a ROS system. They represent individual programs or processes, each responsible for a specific task. Nodes can be implemented in various programming languages such as C++ or Python. Common examples include sensor drivers, control algorithms, and user interface modules.

2. **Messages:**
   Messages are structured data packets used for communication between nodes. They define the format and content of the data being exchanged. ROS includes a variety of built-in message types (e.g., `std_msgs/String`, `geometry_msgs/Twist`), and also allows users to define custom message types.

3. **Topics:**
   Topics are named buses or channels through which nodes can publish and subscribe to messages. A node can publish messages on a specific topic, and other nodes can subscribe to that topic to receive the data. Topics allow for decoupled communication, meaning nodes do not need direct awareness of each others existence.

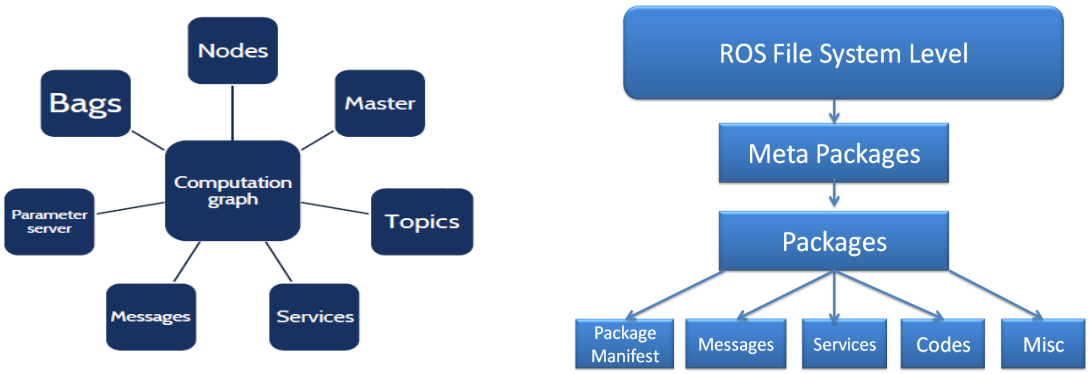
4. **Services:**
   Services provide a request/reply mechanism between nodes. A client node sends a request to a server node, which processes the request and sends back a reply. Services are typically used for tasks that require a response, such as retrieving data or executing a specific command.

5. **Master:**
   The ROS Master is responsible for facilitating communication between nodes. It maintains a registry of all active nodes and their topics and services, enabling nodes to find each other and establish peer-to-peer connections. In ROS 1, the Master is centralized and considered a single point of failure. In contrast, ROS 2 adopts a decentralized architecture to improve robustness.

6. **Parameter Server:**
   The Parameter Server provides a shared, multi-variable storage space for configuration parameters. These parameters can be used by nodes to control their behavior dynamically without modifying the code, enabling runtime configuration and tuning.

7. **Launch Files:**
   Launch files are XML files (in ROS 1) or Python-based scripts (in ROS 2) used to start and configure multiple nodes simultaneously. They allow users to set parameters, remap topics, and automate system startup. Launch files are essential for managing large and complex robotic applications.

8. **Bags:**
   Bags (with extension `.bag`) are files that record ROS message data during runtime. They are useful for data logging, debugging, simulation, and offline analysis. Bags can be replayed to mimic live robot behavior and test algorithms without requiring the robot to be physically present.



(a) ROS computational graph showing nodes, topics, and services.

(b) ROS File System Level.

Figure 5.5: Illustrations of the Robot Operating System (ROS) architecture.

5.3 Motion control of the pioneer 3-dx mobile robot

The Pioneer 3-DX is a differential-drive mobile robot, meaning its movement is controlled by adjusting the angular velocities of its two independently driven wheels. The robot's motion can be modeled using unicycle kinematics, which are commonly used for non-holonomic wheeled robots.

5.3.1 Kinematic model

Let the robot's pose in the global frame be defined as: $\mathbf{x} = [x \;\; y \;\; \theta]^T$ where:

- x, y: Cartesian coordinates of the robot's center,
- θ: Orientation angle of the robot with respect to the global frame.

The control inputs are the linear velocity v and the angular velocity ω. The robots motion is governed by the following kinematic equations:

$$\begin{cases} \dot{x} = v\cos(\theta) \\ \dot{y} = v\sin(\theta) \\ \dot{\theta} = \omega \end{cases} \tag{5.7}$$

These inputs are derived from the wheel velocities as:

$$\begin{cases} v = \dfrac{r}{2}(\omega_R + \omega_L) \\ \omega = \dfrac{r}{L}(\omega_R - \omega_L) \end{cases} \tag{5.8}$$

where:

- r: Radius of the wheels,
- L: Distance between the two wheels (wheelbase),
- ω_R, ω_L: Angular velocities of the right and left wheels, respectively.

5.3.2 Wheel velocity-based control strategy

The motion behavior of a differential drive robot such as the Pioneer 3-DX depends on the velocities of its left (v_L) and right (v_R) wheels. The control strategy can be described as follows:

- If $v_R > 0$ and $v_L > 0$ with $v_R = v_L$ ($v > 0$, $\omega = 0$), the robot moves forward in a straight line.
- If $v_R < 0$ and $v_L < 0$ with $v_R = v_L$ ($v < 0$, $\omega = 0$), the robot moves backward in a straight line.
- If $v_R > 0$ and $v_L > 0$ with $v_R > v_L$ ($v > 0$, $\omega > 0$), the robot performs a left turn along a forward arc.
- If $v_R > 0$ and $v_L > 0$ with $v_R < v_L$ ($v > 0$, $\omega < 0$), the robot performs a right turn along a forward arc.
- If $v_R < 0$ and $v_L < 0$ with $v_R > v_L$ ($v < 0$, $\omega > 0$), the robot performs a left turn along a backward arc.
- If $v_R < 0$ and $v_L < 0$ with $v_R < v_L$ ($v < 0$, $\omega < 0$), the robot performs a right turn along a backward arc.
- If $v_R < 0$ and $v_L > 0$ ($v = 0$, $\omega \neq 0$) with $v_R = -v_L$, the robot rotates in place counterclockwise.

- If $v_R > 0$ and $v_L < 0$ ($v = 0$, $\omega \neq 0$) with $v_R = -v_L$, the robot rotates in place clockwise.
- If $v_R = 0$ and $v_L = 0$ ($v = 0$, $\omega = 0$), the robot remains stationary situation.

5.3.3 Control scheme

The control strategy of the Pioneer 3-DX is typically organized into two layers:

- **High-level control:** This layer handles trajectory generation, path planning (e.g., A*, DWA), and decision making. It outputs a reference trajectory or pose that the robot must follow.
- **Low-level control:** This layer ensures that the robot's actual pose tracks the reference trajectory using feedback control techniques.

A common approach for low-level control is a trajectory tracking controller. Given a desired pose $\mathbf{x}_d = \begin{bmatrix} x_d & y_d & \theta_d \end{bmatrix}^T$, the tracking errors are defined as:

$$\begin{cases} e_x = \cos(\theta)(x_d - x) + \sin(\theta)(y_d - y) \\ e_y = -\sin(\theta)(x_d - x) + \cos(\theta)(y_d - y) \\ e_\theta = \theta_d - \theta \end{cases} \tag{5.9}$$

A simple feedback control law can be expressed as:

$$\begin{cases} v = v_d \cos(e_\theta) + k_1 e_x \\ \omega = \omega_d + k_2 v_d e_y + k_3 \sin(e_\theta) \end{cases} \tag{5.10}$$

where v_d and ω_d are the desired linear and angular velocities, and k_1, k_2, k_3 are positive control gains. This control law stabilizes the robot toward the desired pose and can be extended with fuzzy logic or reinforcement learning to adapt in real-time dynamic environments.

5.4 Simulation Results and Discussion

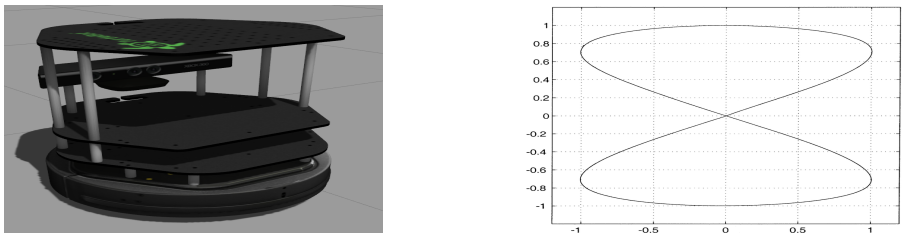
Numerical tests were carried out in this section to assess and evaluate the feasibility and performance of the proposed approach. The simulations were performed within a realistic and controlled environment using the *Gazebo* simulator, which was seamlessly integrated with *MATLAB R2020b*

via the *ROS1 Noetic middleware*. This *3D simulator* facilitated a comprehensive evaluation and strong validation of the robotic control strategies. In the tests of this section, we have used ***TurtleBot2*** mobile robot, visually represented by its *Gazebo* model in Figure 5.6a as a differential drive model to execute the proposed algorithms. This set of tests was run on PC featured an *Intel Core i9 processor*, a *12GB NVIDIA GPU*, *32GB of RAM*, and ran on *Linux Ubuntu 20.04 LTS*. Through Robot Operating System (ROS), control commands were generated and transmitted to the test prototype *TurtleBot2*, while continuous sensor data, including odometry feedback and *Lidar* data were simultaneously monitored. The technical specifications and key operational parameters for the TurtleBot2 are summarized in Table 5.4.

5.4.1 Trajectory tracking

The objective of this test is to drive the *TurtleBot2* precisely along the predefined path illustrated in Figure 5.6b. This path is mathematically defined in (5.11), where $0 \leq t \leq T$ represents the simulation duration and T is the total execution time. The parameter a is a fixed positive constant throughout the experiment. The values for all constant parameters for this test are fixed as listed in Table 5.2.

$$\begin{cases} x_d(t) = a * sin(t/10) \\ y_d(t) = a * sin(t/20) \end{cases} \tag{5.11}$$



(a) Gazebo model of the TurtleBot2 mobile robot.

(b) Eight-shaped reference trajectory tracking for mobile robots

Figure 5.6: Simulation setup and reference trajectory used for trajectory tracking.

The robot operates within an unobstructed environment, as depicted in Figure 5.7a. In this visualization, the green circle denotes the robot's starting position, and the black cylindrical shape represents the mobile robot

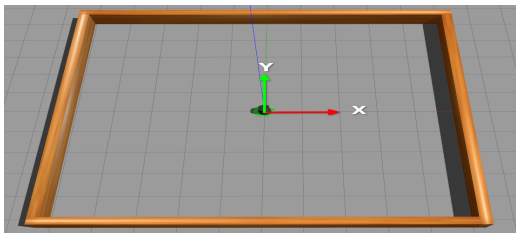
itself.

The simulated results in Figure 5.8 demonstrate the effectiveness of the proposed methodology. Figure 5.8a illustrates the tracking errors between the reference and actual trajectories in the x, y, and θ (orientation) coordinates. As seen, the consistently very small observed errors validate the high efficacy of the proposed approach in achieving precise trajectory tracking.

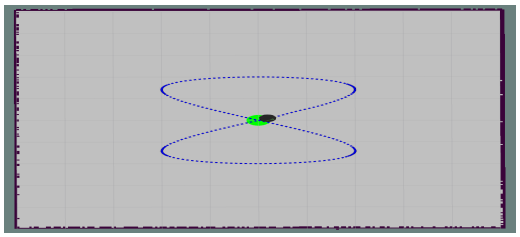
Figure 5.8b displays the driving linear (the blue) curve and angular (the red curve) velocities (ω and v) of the mobile robot during the task. The smooth fluctuations in both velocity components confirm that the robot's motion adheres to its dynamic constraints, thereby ensuring stable and controlled performance.

In Figure 5.9a, the tracking results are visually depicted, with the blue curve representing the reference trajectory and the red curve showing the actual path followed by the robot. The close alignment between the two curves, characterized by minimal variance, signifies smooth and accurate tracking performance.

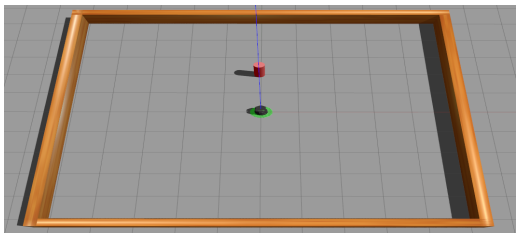
Figure 5.7b further presents the trajectory tracking visualization within the *ROS RViz tool*. Here, the green circle marks the robot's initial starting point, while the blue curve illustrates the actual trajectory traversed by the mobile robot from its origin to its destination.



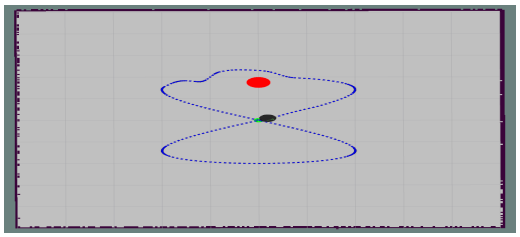
(a) Empty gazebo environment for robot simulation



(b) Robot trajectory in an obstacle-free environment (RViz)



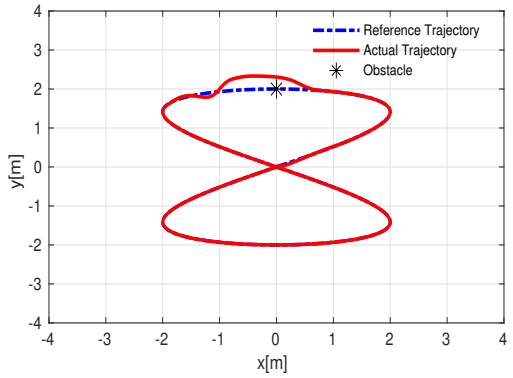
(c) Gazebo simulation environment for static obstacle avoidance



(d) Robot trajectory visualization in an environment with obstacles (RViz)

Figure 5.7: Gazebo simulation and RViz visualization for robotic environments.

Figures 5.8c and 5.8d present the tracking error and velocity variations, respectively, during the combined trajectory tracking and obstacle avoidance behavior. While tracking errors remain minimal during the general path following, they show a notable increase, particularly in orientation, when the robot executes maneuvers to safely circumvent the obstacle. Figure 5.8d illustrates slight fluctuations in the robot's linear and angular velocities during the tracking phase, with more pronounced variabilityespecially in angular velocityduring the active obstacle avoidance maneuvers.



(a) Trajectory tracking process

(b) rajectory tracking with obstacle avoidance

Figure 5.9: Analysis of Robot Motion in Tracking and Avoidance Scenarios.

Figure 5.9b comprehensively presents the combined tracking and avoidance results. The blue curve denotes the reference trajectory, the red curve illustrates the actual trajectory followed by the robot, and the black star indicates the position of the static, known obstacle. The actual trajectory closely aligns with the reference during normal tracking, exhibiting minimal deviation. However, an increased variance is observed as the robot approaches the obstacle within its safety threshold, demonstrating smooth, precise, and secure navigation around the impediment.
Figure 5.7d visually represents the trajectory tracking and obstacle avoidance within the ROS RViz tool. The green circle indicates the starting position, the red circle denotes the static obstacle, and the blue curve traces the actual path of the mobile robot from its start to its goal. The overall simulation results for both the pure tracking and obstacle avoidance tasks are summarized in Table 5.3. This table indicates a slight increase in both the total path length and travel time when obstacle avoidance is engaged, confirming that safe navigation was prioritized.

Table 5.2: Simulation parameters

Parameter	Value
Sampling time (dt)	0.1 (s)
Simulation time (T)	126 (s)
Constant (a)	2
Linear tracking velocity (vt)	0.11 (m/s)
k_{p1},k_{p2}, k_{d1}, k_{d2}, k_{th}	1, 1, 0.7, 0.7, 1

Table 5.3: Summary of simulation experiment data results

Behavior	Path length (m)	Time (s)
Trajectory tracking	18.94	129
Obstacle avoidance	19.25	131

5.5 Experiments and Results Validation

Piratical tests are provided in this section to confirm the simulation results and rigorously validate the proposed algorithms. The mobile robot employed for these tests is shown in Figure 5.11a. This prototype is outfitted with a comprehensive suite of sensors, including an Inertial Measurement Unit (IMU), a SICK LMS200 LiDAR, an MS Kinect v1 depth camera, 16 sonar sensors, and a wheel encoder. The robot uses differential drive locomotion. A master-slave wireless connection was established during the experiment: a low-level laptop PC, mounted on the mobile robot, served as the slave, while a high-performance PC, running ROS, acted as the master for overall system control. The experimental setup of the master laptop comprised a 64-bit Ubuntu 20.04 operating system, powered by an Intel Core i9 CPU, 32 GB of RAM, and a 12 GB NVIDIA GeForce graphics card.
The parameters specified in Table 5.4 comprehensively fulfill the experimental requirements of this study. Wheel encoders, vital instruments for determining a robot's position and orientation, are essential components in the kinematic modeling and serve as crucial inputs for the presented algorithms. The accuracy of encoder data directly impacts the effectiveness of the algorithms; any inaccuracies can adversely affect the precision of trajectory tracking and obstacle avoidance maneuvers. The robot is tasked for same challenge of the considered test in the simulation level. Both tra-

Table 5.4: Parameters of the two mobile robots: TurtleBot2 and Pioneer-3DX

Parameter	Turtlebot2	Pioneer-3DX
Robot weight	6.3 (kg)	9 (kg)
Operating payload	5 (kg)	17 (kg)
Max. forward/backward Speed	0.65 (m/s)	1.2 (m/s)
Rotation speed	3.14 (rad/s)	5.23 (rad/s)
Locomotion drive	Differential drive	Differential drive
Traversable terrain	Indoor	Indoor, Outdoor

jectory tracking and obstacle avoidance tasks initiated at the origin $(0,0)$, with the target point also located at these coordinates. For the obstacle avoidance challenge, a rectangular block, measuring $0.2\,\text{m} \times 0.3\,\text{m}$, was strategically positioned within the test environment.

5.5.1 Trajectory tracking

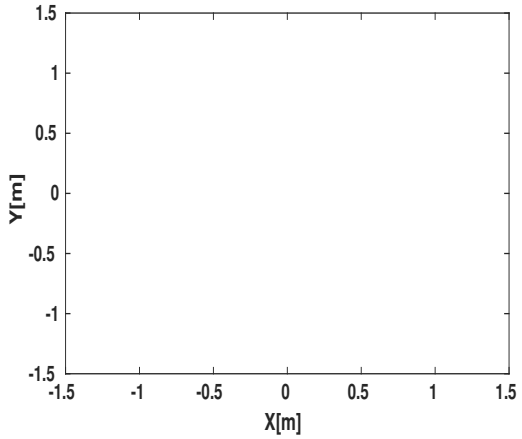
Figure 5.11a presents the Adapt MobileRobots Pioneer 3-DX DWMR, which is equipped with an onboard netbook PC and all the necessary sensors for mobile robot navigation. Figure 5.11b illustrates the experimental environment, defined as a square area of 4×4 meters, with the green square indicating the starting position.

In Figure 5.12a, which visualizes the robot's tracking behavior, the red eight-shaped curve represents the trajectory executed from the start to the goal during the tracking task. The robot's motion appears smooth and precise, as further demonstrated in Figure 5.13a. This figure uses the ROS RViz tool to show the actual trajectory (blue curve) with the robot positioned centrally.

Figure 5.14a shows the tracking errors observed during the trajectory-following task. The error remains minimal, with only minor deviations attributed to the robot's movement and the inherent limitations of odometry readings. Figure 5.14b illustrates the linear velocity (blue curve) and angular velocity (red curve), both of which exhibit smooth profiles within acceptable operational limits.

The comparison of the actual (blue) and reference (red) trajectories in Figure 5.10a indicates a high degree of similarity, confirming the accuracy of the tracking performance. A video demonstration of the initial trajectory

following experiment is available at: https://youtu.be/CybDwajlTUU

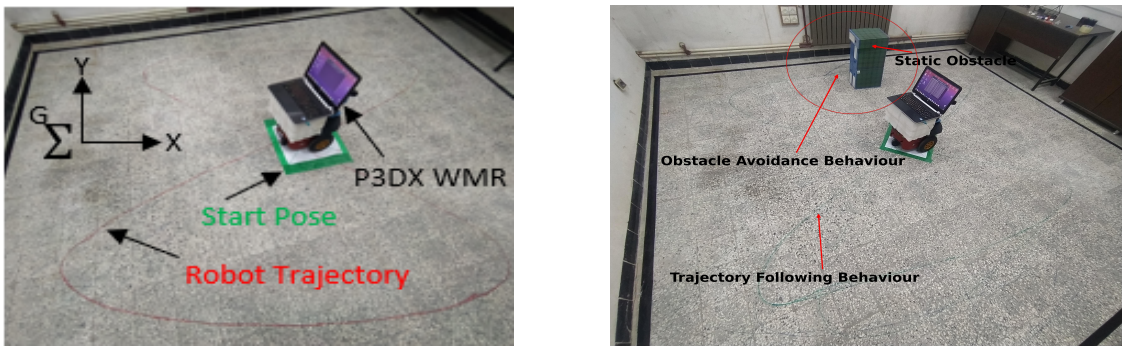


(a) tracking task **(b)** avoidance task

Figure 5.10: Experimental Validation: Analysis of Robot Trajectories

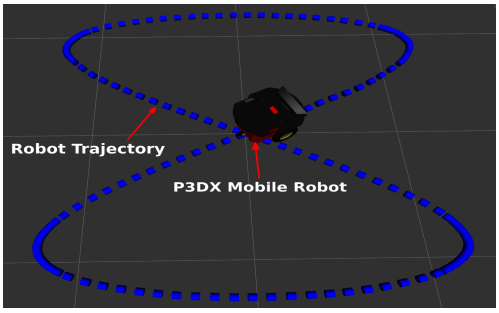
(a) Pioneer 3-DX mobile robot **(b)** Experimental platform

Figure 5.11: Environmental modeling: (a) Adapted MobileRobots Pioneer 3-DX, (b) Real-world environment of the mobile robot.

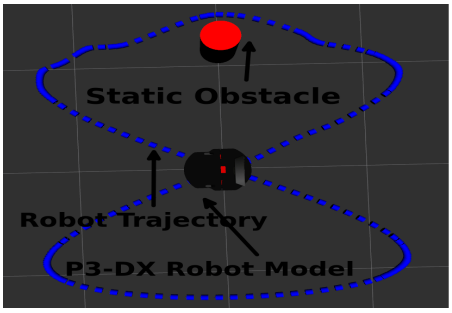


(a) Tracking tracking process **(b)** Obstacle avoidance process

Figure 5.12: Trajectory tracking and obstacle avoidance behaviors.



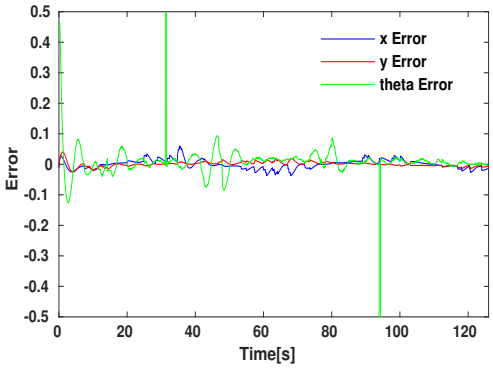
(a) Actual trajectory in tracking task



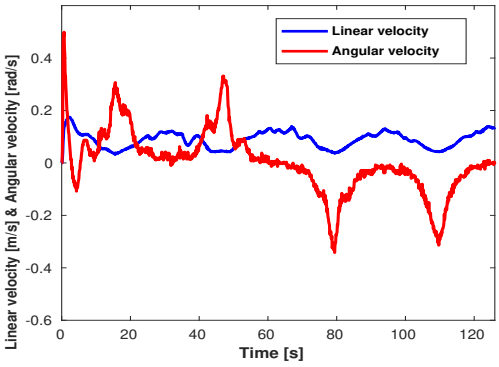
(b) Actual trajectory in tracking with avoidance task

Figure 5.13: Visualization of the executed robot trajectory in RViz tool.

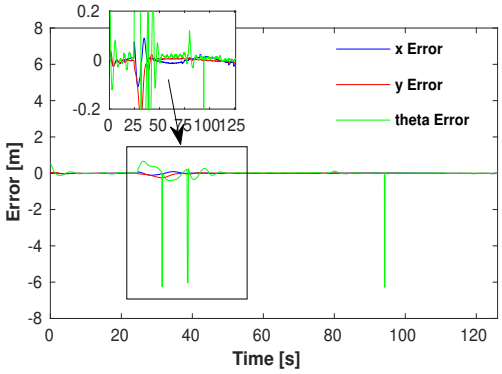
5.5.2 Obstacle avoidance



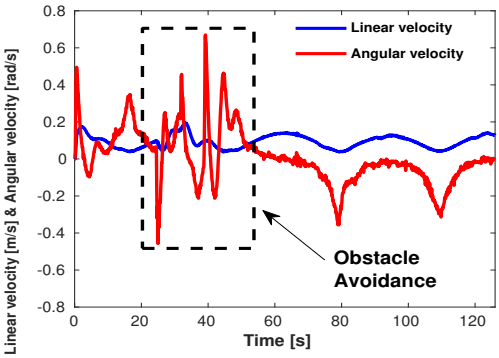
(a) Analysis of pose errors (X, Y, and θ) during Tracking



(b) Linear (v) and angular (ω) velocities of the robot during the trajectory tracking task



(c) Analysis of Pose Errors (X, Y, θ) in an Obstacle Avoidance Scenario



(d) Linear (v) and angular (ω) velocity profiles during trajectory tracking with an obstacle avoidance maneuver

Figure 5.14: Pose errors and velocity profiles during tracking and obstacle avoidance tasks.

Figure 5.12b illustrates the results of applying the combined approach for tracking and obstacle avoidance tasks. The green box represents a static obstacle, which the robot successfully avoids while adhering to the predetermined trajectory, guided by the Dynamic Feedback Linearization (DFL) method.

As shown in Figure 5.13b, the visualization of the actual trajectory within the ROS RViz tool displays the robot's path (blue curve) and the static obstacle (red circle). The robot safely navigates around this obstacle between 26 s and 36 s.

Figure 5.14c shows the tracking errors throughout the task, including the obstacle avoidance maneuver. The errors remain small, primarily due to the robots movement and the limitations of odometry measurements.

Figure 5.14d illustrates the linear velocity (blue curve) and angular velocity (red curve) profiles. Both velocities demonstrate smooth variations and stay within acceptable operating ranges.

In Figure 5.10b, the results of the tracking with obstacle avoidance task are presented, with the executed trajectory (red curve) closely matching the reference trajectory (blue curve). This close alignment demonstrates the accuracy of the control approach.

A video demonstration of the obstacle avoidance experiment is available at: https://youtu.be/Z6Bi9SjnEjY?si=V4Erc6knmebz41R-.

The experimental data for both the trajectory tracking and obstacle avoidance challenges are summarized in Table 5.5. The minor differences in path length and travel time between the two tasks highlight the effectiveness of the method in ensuring safe and efficient navigation.

The strong correlation between simulation and real-world experimental results, obtained using the ROS middleware, underscores the robustness and practical applicability of the proposed methodology across diverse operational scenarios.

Table 5.5: Experimental Results and Analysis

Behavior	Path length (m)	Time (s)
Trajectory tracking	11.77	123
Obstacle avoidance	12.05	126

5.6 Conclusion

In this chapter, we have introduced a novel Dynamic Feedback Linearization (DFL) control strategy, integrated with Fuzzy Logic Control (FLC), to address the complex challenges of robust trajectory tracking and obstacle avoidance for Differentially-Driven Mobile Robots (DWMRs). By exploiting the systems flatness property, this approach effectively decouples and simplifies the problems of motion planning and control design. The controller, formulated in the flat output space, ensures precise trajectory tracking while maintaining overall system stability.

A key advantage of this hybrid control architecturecombining flatness-based feedback with fuzzy logic is the significant enhancement in system performance, motion accuracy, and disturbance rejection compared to traditional linear flatness-based controllers. The proposed method has undergone rigorous validation through theoretical analysis, extensive simulations, and real-world experimental testing. These validations consistently demonstrate the systems robust tracking capabilities and exceptional precision in obstacle avoidance during task execution.

Future research will focus on expanding this methodology to incorporate dynamic obstacle management and integrate advanced sensor technologies, such as laser scanners and vision cameras. These enhancements will significantly improve the robots environmental perception, enabling safe and efficient navigation in both static and dynamic environments. This continuous development will ensure the ongoing effectiveness and adaptability of the proposed approach in increasingly complex real-world applications.

General Conclusion

The work presented in this thesis addresses the fundamental challenges of motion planning and decision-making in mobile robots, with a particular focus on integrating the A-Star algorithm for global path planning and Dynamic Window Approach (DWA) for local path planning and fuzzy dynamic feedback linearization (FDFL) for trajectory tracking and obstacle avoidance into autonomous navigation of wheeled mobile robot, we explored the use of classical and advanced algorithms for both global path planning and local obstacle avoidance and trajectory tracking, emphasizing the need for robustness, adaptability, and real-time responsiveness in static and dynamic environments.

This work begins with a general overview of mobile robot navigation with required tasks and modules for accomplishing such as localization, mapping, path planning, motion control, and decision making. On the one hand, we focus on path planning and motion control tasks with an overview and state of the art about path planning algorithms and approaches existing in literature; on the other hand, we describe the motion control technique to execute the planned path generated by path planning tasks.

Chapter 2, we explore the principles, challenges, and interests of different planning strategies and control techniques for mobile robot navigation. The feedback linearization approach is introduced as a fundamental tool for the control of robotic systems for their usefulness not only in the trajectory tracking phase but also in the trajectory generation phase along a predefined path. An exposition of the main advancements and developments in dynamic feedback linearization techniques and fuzzy control techniques is introduced, with definition and their modules and integration into mobile robot obstacle avoidance.

Chapter 3 presents a hybrid path planning method that integrates an adaptive A* algorithm with an improved Dynamic Window Approach (DWA) to address the limitations of traditional A*, including excessive path nodes, long planning time, unsmooth trajectories, proximity to obstacles, and its restriction to static environments. Additionally, the proposed method aims

to overcome the local minimum problem inherent in the original DWA approach. First, the adaptive A* algorithm incorporates an adaptive weighting mechanism for the heuristic function, enhancing search efficiency. To further optimize the planned path, a redundant-node removal strategy is applied to eliminate unnecessary path nodes, and Eta Spline curves are used for path smoothing. Furthermore, a fuzzy control system is introduced to optimize the weights of the DWA evaluation function dynamically. Finally, the adaptive A* algorithm and the optimized DWA algorithm are integrated to achieve a more efficient and reliable path-planning strategy. Simulation results demonstrate that the adaptive A* algorithm reduces planning time by 30.9%, the number of path nodes by 54.55%, and the number of searched nodes by 17.14% compared to the traditional A* algorithm. Similarly, the optimized DWA reduces path length by 6.74% and planning time by 84.83% compared to the original DWA approach. These improvements significantly enhance search efficiency and path smoothness while effectively addressing the local minimum problem. Overall, this hybrid approach provides a robust and efficient solution for autonomous mobile robot navigation, with promising applications in real-world robotics. In future work, we plan to incorporate unknown and dynamic obstacles into the environment and validate our results through real-world experiments using a physical robot under the Robot Operating System (ROS).

Chapter 4 presents a dynamic path planning fusion method that integrates an enhanced A* algorithm with an improved Dynamic Window Approach (DWA). On one hand, traditional A*-based global path planning suffers from several limitations, including long and suboptimal paths, high planning time, lack of smoothness, proximity to obstacles, and restrictions to static environments. To address these issues, an enhanced A* algorithm is proposed. It incorporates an adaptive heuristic mechanism that dynamically adjusts weights in response to environmental changes. Additionally, a node optimization strategy is introduced to improve search efficiency. A path optimization strategy is introduced to eliminate redundant nodes and turning points, ensuring a smoother trajectory, followed by cubic B-spline curve smoothing for enhanced path continuity. On the other hand, the traditional DWA used for local path planning presents several drawbacks, such as susceptibility to local minima, difficulty navigating narrow passages, and occasional failure to reach the goal. Its main limitation lies in weak obstacle avoidance in complex and dynamic scenarios. To overcome these issues, an improved DWA algorithm is proposed by modifying and optimizing its evaluation function to better adapt to un-

known and dynamic obstacles. The integration of the enhanced A* and improved DWA results in a hybrid path planning strategy that is both efficient and reliable. Simulation results show that the enhanced A* algorithm reduces the path length in Map1 and Map2 by 7.8% and 3.9%, respectively, compared to the traditional A* algorithm. Likewise, the improved DWA achieves reductions in path length by 3.13% and 4.57%, and planning time by 66.25% and 64.63%, respectively, compared to the original DWA. These enhancements significantly improve path smoothness, search efficiency, and obstacle avoidance capabilities while effectively mitigating the local minimum issue. Overall, this hybrid approach offers a robust and efficient solution for autonomous mobile robot navigation, with strong potential for real-world robotic applications. As future work, we plan to incorporate unknown and dynamic obstacles into the environment and validate the proposed method through real-world experiments using a physical robot operating under the Robot Operating System (ROS).

Chapter 5, presents a novel control strategy is introduced that combines Dynamic Feedback Linearization (DFL) with Fuzzy Logic Control (FLC) to address the demanding tasks of robust trajectory tracking and obstacle avoidance for differentially driven mobile robots (DWMRs). The approach begins by exploiting the system's differential flatness, which facilitates the decoupling of motion planning and control tasks. The controller is formulated in the flat output domain, allowing for accurate trajectory tracking while preserving system stability. The integration of flatness-based feedback with fuzzy logic enhances control performance, offering superior trajectory precision, disturbance rejection, and robustness compared to traditional linear feedback methods. The effectiveness of the proposed control architecture has been thoroughly demonstrated through a combination of theoretical analysis, comprehensive simulations, and experimental validation. These assessments consistently underscore the controller's ability to achieve high tracking accuracy and reliable obstacle avoidance in diverse scenarios. Future developments will focus on extending the systems capabilities to handle dynamic obstacles and leveraging advanced sensory inputs, such as LiDAR and vision-based technologies, to further enhance environmental awareness. These advancements aim to improve the robot's ability to operate safely and adaptively in complex, real-world environments, ensuring the long-term relevance and applicability of the proposed solution.

In particular, the introduction of fuzzy reasoning into the path planning framework has enhanced the robots ability to cope with uncertainties, of-

fering a more flexible control system that adapts to real-time environmental changes. The results demonstrated that the proposed fuzzy-based motion planning strategies improved the robots trajectory tracking, obstacle avoidance, and overall robustness compared to traditional methods. The findings of this thesis contribute significantly to the development of more intelligent and autonomous mobile robots capable of navigating both static and dynamic environments. Future work will focus on further enhancing the integration of perception and motion planning, incorporating real-time sensory feedback, and applying these methods to more complex environments and real-world robotic platforms.

Bibliography

[1] S. Pütz, “Navigation control and path planning for autonomous mobile robots", *KI - Künstl. Intell.*, vol. 37, no. 24, pp. 183186, Dec. 2023. 10.1007/s13218-024-00836-x

[2] J. Sun, J. Zhao, X. Hu, H. Gao, and J. Yu, “Autonomous navigation system of indoor mobile robots using 2D Lidar", *Mathematics*, vol. 11, no. 6, p. 1455, Mar. 2023. https://doi.org/10.3390/math11061455

[3] S. J. Al-Kamil and R. Szabolcsi, “Enhancing mobile robot navigation: Optimization of trajectories through machine learning techniques for improved path planning efficiency", *Mathematics*, vol. 12, no. 12, p. 1787, Jun. 2024. https://doi.org/10.3390/math12121787

[4] R.-H. Sun, X. Zhao, C.-D. Wu, L. Zhang, and B. Zhao, “Research on mobile robot navigation method based on semantic information", *Sensors (Basel)*, vol. 24, no. 13, p. 4341, Jul. 2024. https://doi.org/10.3390/s24134341

[5] M.-F. R. Lee and S. H. Yusuf, “Mobile robot navigation using deep reinforcement learning", *Processes (Basel)*, vol. 10, no. 12, p. 2748, Dec. 2022. https://doi.org/10.3390/pr10122748

[6] J. Galarza-Falfan et al., “Path planning for Autonomous Mobile Robot using intelligent algorithms", *Technologies (Basel)*, vol. 12, no. 6, p. 82, Jun. 2024. https://doi.org/10.3390/technologies12060082

[7] R. Roy, Y.-P. Tu, L.-J. Sheu, W.-H. Chieng, L.-C. Tang, and H. Ismail, Path planning and motion control of indoor mobile robot under exploration-based SLAM (e-SLAM), *Sensors (Basel)*, vol. 23, no. 7, Mar. 2023. https://doi.org/10.3390/s23073606

[8] J. Zhong, D. Kong, Y. Wei, X. Hu, and Y. Yang, “Efficiency-optimized path planning algorithm for car-like mobile robots in bilateral constraint corridor environments", *Rob. Auton. Syst.*, vol. 186, no. 104923, p. 104923, Apr. 2025. https://doi.org/10.1016/j.robot.2025.104923

[9] P. Li, D. Chen, Y. Wang, L. Zhang, and S. Zhao, “Path planning of mobile robot based on improved TD3 algorithm in dynamic environment", *Heliyon*, vol. 10, no. 11, p. e32167, Jun. 2024. https://doi.org/10.1016/j.heliyon.2024.e32167

[10] F. Huo, S. Zhu, H. Dong, and W. Ren, “A new approach to smooth path planning of Ackerman mobile robot based on improved ACO algorithm and B-spline curve", *Rob. Auton. Syst.*, vol. 175, no. 104655, p. 104655, May 2024. https://doi.org/10.1016/j.robot.2024.104655

[11] A. Marashian and A. Razminia, “Mobile robots path-planning and path-tracking in static and dynamic environments: Dynamic programming approach", *Rob. Auton. Syst.*, vol. 172, no. 104592, p. 104592, Feb. 2024. https://doi.org/10.1016/j.robot.2023.104592

[12] P. K. Mohanty, “Path planning of mobile robots under uncertain navigation environments using FCM clustering ANFIS", *Wirel. Pers. Commun.*, vol. 137, no. 2, pp. 12511276, Jul. 2024. https://doi.org/10.1007/s11277-024-11463-y

[13] Q. Xing, S. Xu, H. Wang, J. Wang, W. Zhao, and H. Xu, “Path planning of a mobile robot using an improved mixed-method of potential field and wall following", *Int. J. Adv. Robot. Syst.*, vol. 20, no. 3, p. 172988062311691, May 2023. https://doi.org/10.1177/17298806231169186

[14] M. A. Shoeib, J. Lewandowski, and A. M. Omara, “A novel methodology for vision-based path planning and obstacle avoidance in mobile robot applications", *Adv. Robot.*, vol. 38, no. 12, pp. 802817, Jun. 2024. https://doi.org/10.1080/01691864.2024.2315591

[15] S. Abdallaoui, E.-H. Aglzim, A. Chaibet, and A. Kribèche, “Thorough review analysis of safe control of autonomous vehicles: Path planning and navigation techniques", *Energies*, vol. 15, no. 4, p. 1358, Feb. 2022. https://doi.org/10.3390/en15041358

[16] H. Qin, S. Shao, T. Wang, X. Yu, Y. Jiang, and Z. Cao, “Review of autonomous path planning algorithms for mobile robots", *Drones*, vol. 7, no. 3, p. 211, Mar. 2023. https://doi.org/10.3390/drones7030211

[17] S. Louda, N. Karkar, F. Seghir, and O. Boutalbi, “Mobile robot path planning algorithm based on A* algorithm and dynamic window approach for autonomous navigation", *in 2024 International Conference on Advances in Electrical and Communication Technologies (ICAECOT)*, Setif, Algeria, 2024, pp. 16. https://doi.org/10.1109/ICAECOT62402.2024.10828660

[18] X. Xu, J. Zeng, Y. Zhao, and X. Lü, “Research on global path planning algorithm for mobile robots based on improved A", *Expert Syst. Appl.*, vol. 243, no. 122922, p. 122922, Jun. 2024. https://doi.org/10.1016/j.eswa.2023.122922

[19] C. Li, X. Huang, J. Ding, K. Song, and S. Lu, “Global path planning based on a bidirectional alternating search A* algorithm for mobile robots", *Comput. Ind. Eng.*, vol. 168, no. 108123, p. 108123, Jun. 2022. https://doi.org/10.1016/j.cie.2022.108123

[20] P. Zhou, Z. Xie, W. Zhou, and Z. Tan, “A heuristic integrated scheduling algorithm based on improved Dijkstra algorithm", *Electronics (Basel)*, vol. 12, no. 20, p. 4189, Oct. 2023. https://doi.org/10.3390/electronics12204189

[21] Z. Lin, K. Wu, R. Shen, X. Yu, and S. Huang, “An efficient and accurate A-star algorithm for autonomous vehicle path planning", *IEEE Trans. Veh. Technol.*, vol. 73, no. 6, pp. 90039008, Jun. 2024. https://doi.org/10.1109/TVT.2023.3348140

[22] B. Wu, W. Zhang, X. Chi, D. Jiang, Y. Yi, and Y. Lu, “A novel AGV path planning approach for narrow channels based on the Bi-RRT algorithm with a failure rate threshold", *Sensors (Basel)*, vol. 23, no. 17, Aug. 2023. https://doi.org/10.3390/s23177547

[23] L. Qiao, X. Luo, and Q. Luo, “An optimized probabilistic roadmap algorithm for path planning of mobile robots in complex environments with narrow channels", *Sensors (Basel)*, vol. 22, no. 22, p. 8983, Nov. 2022. https://doi.org/10.3390/s22228983

[24] S. Tian, Y. Li, Y. Kang, and J. Xia, “Multi-robot path planning in wireless sensor networks based on jump mechanism PSO and safety gap obstacle avoidance", *Future Gener. Comput. Syst.*, vol. 118, pp. 3747, May 2021. https://doi.org/10.1016/j.future.2020.12.012

[25] C. Miao, G. Chen, C. Yan, and Y. Wu, “Path planning optimization of indoor mobile robot based on adaptive ant colony algorithm", *Comput. Ind. Eng.*, vol. 156, no. 107230, p. 107230, Jun. 2021. https://doi.org/10.1016/j.cie.2021.107230

[26] S. Kumar and A. Sikander, “Optimum mobile robot path planning using improved artificial bee colony algorithm and evolutionary programming", *Arab. J. Sci. Eng.*, Jan. 2022. https://doi.org/10.1007/s13369-021-06326-8

[27] D. Foead, A. Ghifari, M. B. Kusuma, N. Hanafiah, and E. Gunawan, “A systematic literature review of A* pathfinding", *Procedia Comput. Sci.*, vol. 179, pp. 507514, 2021. https://doi.org/10.1016/j.procs.2021.01.034

[28] C. Yin, C. Tan, C. Wang, and F. Shen, “An improved A-star path planning algorithm based on mobile robots in medical testing laboratories", *Sensors (Basel)*, vol. 24, no. 6, p. 1784, Mar. 2024. https://doi.org/10.3390/s24061784

[29] Y. Sun, J. Yang, D. Zhao, Y. Shu, Z. Zhang, and S. Wang, “A global trajectory planning framework based on minimizing the risk index", *Actuators*, vol. 12, no. 7, p. 270, Jun. 2023. https://doi.org/10.3390/act12070270

[30] Y. Sun, Q. Yuan, Q. Gao, and L. Xu, A multiple environment available path planning based on an improved A* algorithm, *Int. J. Comput. Intell. Syst.*, vol. 17, no. 1, Jul. 2024. https://doi.org/10.1007/s44196-024-00571-z

[31] Z. Yang, J. Li, L. Yang, and H. Chen, “A smooth jump point search algorithm for mobile robots path planning based on a two-dimensional grid model", *J. Robot.*, vol. 2022, pp. 115, Aug. 2022. https://doi.org/10.1155/2022/7682201

[32] Y. Cao and N. Mohamad Nor, “An improved dynamic window approach algorithm for dynamic obstacle avoidance in mobile robot formation", *Decision Analytics Journal*, vol. 11, no. 100471, p. 100471, Jun. 2024, https://doi.org/10.1016/j.dajour.2024.100471

[33] X. Ji, S. Feng, Q. Han, H. Yin, and S. Yu, “Improvement and fusion of A* algorithm and dynamic window approach considering complex environmental information", *Arab. J. Sci. Eng.*, vol. 46, no. 8, pp. 74457459, Aug. 2021. https://doi.org/10.1007/s13369-021-05445-6

[34] W. Zhang, G. Xu, Y. Song, and Y. Wang, “An obstacle avoidance strategy for complex obstacles based on artificial potential field method", *J. Field Robot.*, vol. 40, no. 5, pp. 12311244, Aug. 2023, https://doi.org/10.1002/rob.22183

[35] J. Wu, X. Ma, T. Peng, and H. Wang, “An improved Timed Elastic Band (TEB) algorithm of autonomous ground vehicle (AGV) in complex environment", *Sensors (Basel)*, vol. 21, no. 24, p. 8312, Dec. 2021. https://doi.org/10.3390/s21248312

[36] M. Al-Mallah, M. Ali, and M. Al-Khawaldeh, “Obstacles avoidance for mobile robot using Type-2 fuzzy logic controller", *Robotics*, vol. 11, no. 6, p. 130, Nov. 2022, https://doi.org/10.3390/robotics11060130

[37] X. Cheng, S. Zhang, S. Cheng, Q. Xia, and J. Zhang, “Path-following and obstacle avoidance control of nonholonomic wheeled mobile robot based on deep reinforcement learning", *Appl. Sci. (Basel)*, vol. 12, no. 14, p. 6874, Jul. 2022. https://doi.org/10.3390/app12146874

[38] S. E. Seghiri, N. Mansouri, and A. Chemori, “Socializing A* algorithm for crowd- and socially aware navigation", *Arab. J. Sci. Eng.*, vol. 50, no. 10, pp. 71937205, May 2025. https://doi.org/10.1007/s13369-024-09334-6

[39] K. K. A. Farag, H. H. Shehata, and H. M. El-Batsh, “Mobile robot obstacle avoidance based on neural network with a standardization technique", *J. Robot.*, vol. 2021, pp. 114, Nov. 2021. https://doi.org/10.1155/2021/1129872

[40] X. Gong et al., “A local path planning algorithm for robots based on improved DWA", *Electronics (Basel)*, vol. 13, no. 15, p. 2965, Jul. 2024. https://doi.org/10.3390/electronics13152965

[41] Y. Sun et al., “Local path planning for mobile robots based on fuzzy dynamic window algorithm", *Sensors (Basel)*, vol. 23, no. 19, p. 8260, Oct. 2023. https://doi.org/10.3390/s23198260

[42] M. Kobayashi, H. Zushi, T. Nakamura, and N. Motoi, “Local path planning: Dynamic window approach with Q-learning considering congestion environments for mobile robot", *IEEE Access*, vol. 11, pp. 9673396742, 2023. https://doi.org/10.1109/ACCESS.2023.3311023

[43] O. A. Abubakr, M. A. Jaradat, and M. F. Abdel-Hafez, “Intelligent optimization of adaptive dynamic window approach for mobile robot motion control using fuzzy logic", *IEEE Access*, vol. 10, pp. 119368119378, 2022. https://doi.org/10.1109/ACCESS.2022.3220703

[44] T. Wang, A. Li, D. Guo, G. Du, and W. He, “Global dynamic path planning of AGV based on fusion of improved A* algorithm and dynamic window method", *Sensors (Basel)*, vol. 24, no. 6, p. 2011, Mar. 2024. https://doi.org/10.3390/s24062011

[45] Z. Wang and G. Li, “Research on path planning algorithm of driverless ferry vehicles combining improved A* and DWA", *Sensors (Basel)*, vol. 24, no. 13, p. 4041, Jun. 2024. https://doi.org/10.3390/s24134041

[46] J. Yao and M. Xin, “Suboptimal control design for differential wheeled mobile robots with $\theta-$D technique", *in 2021 60th IEEE Conference on Decision and Control (CDC)*, Austin, TX, USA, 2021, https://doi.org/10.1109/CDC45484.2021.9683480

[47] E. A. Padilla-García, R. D. Cruz-Morales, J. González-Sierra, D. Tinoco-Varela, and M. R. Lorenzo-Gerónimo, “Design, assembly and control of a differential/omnidirectional mobile robot through additive manufacturing", *Machines*, vol. 12, no. 3, p. 163, Feb. 2024, https://doi.org/10.3390/machines12030163

[48] H. T. Najm, N. S. Ahmad, and A. S. Al-Araji, “Enhanced path planning algorithm via hybrid WOA-PSO for differential wheeled mobile robots", *Syst. Sci. Control Eng.*, vol. 12, no. 1, Dec. 2024, https://doi.org/10.1080/21642583.2024.2334301

[49] J. Kim and B. K. Kim, “Cornering trajectory planning avoiding slip for differential-wheeled mobile robots", *IEEE Trans. Ind. Electron.*, vol. 67, no. 8, pp. 66986708, Aug. 2020, https://doi.org/10.1109/TIE.2019.2941156

[50] D. Mújica-Vargas, V. Vela-Rincón, A. Luna-Álvarez, A. Rendón-Castro, M. Matuz-Cruz, and J. Rubio, “Navigation of a differential wheeled robot based on a type-2 fuzzy inference tree", *Machines*, vol. 10, no. 8, p. 660, Aug. 2022, https://doi.org/10.3390/machines10080660

[51] M. Yao, H. Deng, X. Feng, P. Li, Y. Li, and H. Liu, “Global path planning for differential drive mobile robots based on improved BSGA* algorithm", *Appl. Sci. (Basel)*, vol. 13, no. 20, p. 11290, Oct. 2023, https://doi.org/10.3390/app132011290

[52] A. Selek, M. Seder, and I. Petrovi, “Smooth autonomous patrolling for a differential-drive mobile robot in dynamic environments", *Sensors (Basel)*, vol. 23, no. 17, Aug. 2023, https://doi.org/10.3390/s23177421

[53] R. Mathew and S. S. Hiremath, “Development of waypoint tracking controller for differential drive mobile robot", *in 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, Paris, France, 2019, https://doi.org/10.1109/CoDIT.2019.8820389

[54] C.-L. Shih and L.-C. Lin, “Trajectory planning and tracking control of a differential-drive mobile robot in a picture drawing application", *Robotics*, vol. 6, no. 3, p. 17, Aug. 2017. https://doi.org/10.3390/robotics6030017

[55] T. N. T. Cao, B. T. Pham, N. T. Nguyen, D.-L. Vu, and N.-V. Truong, “Second-order terminal sliding mode control for trajectory tracking of a differential drive robot", *Mathematics*, vol. 12, no. 17, p. 2657, Aug. 2024, https://doi.org/10.3390/math12172657

[56] C. Wang, P. Shi, and I. Rudas, “Tracking control for two-wheeled mobile robots via event-triggered mechanism", *ISA Trans.*, Nov. 2024, https://doi.org/10.1016/j.isatra.2024.11.032

[57] H. Huang and J. Gao, “Backstepping and novel sliding mode trajectory tracking controller for wheeled mobile robots", *Mathematics*, vol. 12, no. 10, p. 1458, May 2024, https://doi.org/10.3390/math12101458

[58] S. Moorthy and Y. H. Joo, “Formation control and tracking of mobile robots using distributed estimators and A biologically inspired approach", *J. Electr. Eng. Technol.*, vol. 18, no. 3, pp. 22312244, 2023, https://doi.org/10.1007/s42835-022-01213-0

[59] H. Li and A. V. Savkin, “An algorithm for safe navigation of mobile robots by a sensor network in dynamic cluttered industrial environments", *Robot. Comput. Integr. Manuf.*, vol. 54, pp. 6582, Dec. 2018, https://doi.org/10.1016/j.rcim.2018.05.008

[60] C. Wang, J. Wang, C. Li, D. Ho, J. Cheng, T. Yan, L. Meng, M.Q.-H. Meng, “Safe and robust mobile robot navigation in uneven indoor environments", *Sensors (Basel)*, vol. 19, no. 13, p. 2993, Jul. 2019, https://doi.org/10.3390/s19132993

[61] Y. Raoui and N. Elmennaoui, “Accurate robot navigation using visual invariant features and dynamic neural fields", *International Journal of Robotics and Control Systems*, vol. 4, no. 4, pp. 15841601, Sep. 2024, https://doi.org/10.31763/ijrcs.v4i4.1545

[62] A. Abadi, A. Ayeb, M. Labbadi, D. Fofi, T. Bakir, and H. Mekki, “Robust Tracking Control of Wheeled Mobile Robot Based on Differential Flatness and Sliding Active Disturbance Rejection Control: Simulations and Experiments", *Sensors*, vol. 24, no. 9, pp. 2849, 2024, https://doi.org/10.3390/s24092849

[63] S. Louda, N. Karkar, F. Seghir, and O. Boutalbi, “Fuzzy Dynamic Feedback Linearization for Efficient Mobile Robot Trajectory Tracking and Obstacle Avoidance in Autonomous Navigation", *International Journal of Robotics and Control Systems*, vol. 5, no. 2, p. 881-901, Mar. 2025, https://doi.org/10.31763/ijrcs.v5i2.1780

[64] S. Khesrani, A. Hassam, O. Boutalbi, and M. Boubezoula, “Motion planning and control of nonholonomic mobile robot using flatness and fuzzy logic concepts", *International Journal of Dynamics and Control*, vol. 9, no. 4, pp. 16601671, Dec. 2021, https://doi.org/10.1007/s40435-020-00754-4

[65] H. Xie, J. Zheng, R. Chai, and H. T. Nguyen, “Robust tracking control of a differential drive wheeled mobile robot using fast nonsingular terminal sliding mode", *Computers & Electrical Engineering*, vol. 96, p. 107488, 2021, https://doi.org/10.1016/j.compeleceng.2021.107488

[66] Z. Sun, H. Xie, J. Zheng, Z. Man, and D. He, “Path-following control of Mecanum-wheels omnidirectional mobile robots using nonsingular terminal sliding mode", *Mech. Syst. Signal Process.*, vol. 147, no. 107128, p. 107128, Jan. 2021, https://doi.org/10.1016/j.ymssp.2020.107128

[67] Y. Xie et al., “Coupled fractional-order sliding mode control and obstacle avoidance of a four-wheeled steerable mobile robot", *ISA Trans.*, vol. 108, pp. 282294, Feb. 2021, https://doi.org/10.1016/j.isatra.2020.08.025

[68] A. Haqshenas M., M. M. Fateh, and S. M. Ahmadi, “Adaptive control of electricallydriven nonholonomic wheeled mobile robots: Taylor seriesbased approach with guaranteed asymptotic stability", *Int. J. Adapt. Control Signal Process.*, vol. 34, no. 5, pp. 638661, May 2020, https://doi.org/10.1002/acs.3104

[69] R. Miranda-Colorado and N. R. Cazarez-Castro, “Observer-based fuzzy trajectory-tracking controller for wheeled mobile robots with kinematic disturbances", *Eng. Appl. Artif. Intell.*, vol. 133, no. 108279, p. 108279, Jul. 2024, https://doi.org/10.1016/j.engappai.2024.108279

[70] R. Kubo, Y. Fujii, and H. Nakamura, “Control lyapunov function design for trajectory tracking problems of wheeled mobile robot", *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 61776182, 2020, https://doi.org/10.1016/j.ifacol.2020.12.1704

[71] Z. Tao, Y.-H. Wang, G.-Q. Li, and G. Hou, “Lyapunov based global trajectory tracking control of wheeled mobile robot", *J. Phys. Conf. Ser.*, vol. 2478, no. 10, p. 102018, Jun. 2023, https://doi.org/10.1088/1742-6596/2478/10/102018

[72] G. Oriolo, A. De Luca, and M. Vendittelli, “WMR control via dynamic feedback linearization: design, implementation, and experimental validation", *IEEE Trans. Control Syst. Technol.*, vol. 10, no. 6, pp. 835852, Nov. 2002, https://doi.org/10.1109/TCST.2002.804116

[73] L. Martins, C. Cardeira, and P. Oliveira, “Feedback linearization with zero dynamics stabilization for quadrotor control", *J. Intell. Robot. Syst.*, vol. 101, no. 1, Jan. 2021, https://doi.org/10.1007/s10846-020-01265-2

[74] H. Xie, J. Zheng, Z. Sun, H. Wang, and R. Chai, “Finite-time tracking control for nonholonomic wheeled mobile robot using adaptive fast nonsingular terminal sliding mode", *Nonlinear Dyn.*, vol. 110, no. 2, pp. 14371453, Oct. 2022, https://doi.org/10.1007/s11071-022-07682-2

[75] A. Keymasi Khalaji and M. Jalalnezhad, “Robust forward-backward control of wheeled mobile robots", *ISA Trans.*, vol. 115, pp. 3245, Sep. 2021, https://doi.org/10.1016/j.isatra.2021.01.016

[76] G. Oriolo, “Wheeled Robots", in *Encyclopedia of Systems and Control, London: Springer London*, pp. 18, 2020, https://doi.org/10.1007/978-1-4471-5102-9_178-2

[77] L. Chen and Y. Jia, “Output feedback tracking control of flat systems via exact feedforward linearization and LPV techniques", *Int. J. Control Autom. Syst.*, vol. 17, no. 3, pp. 606616, Mar. 2019, https://doi.org/10.1007/s12555-018-0459-1

[78] A. M. Kopp, L. Fuchs, and C. Ament, “Flatness-based identification of nonlinear dynamics", in *2024 32nd Mediterranean Conference on Control and Automation (MED)*, Chania - Crete, Greece, 2024, https://doi.org/10.1109/MED61351.2024.10566206

[79] A. Abadi, A. E. Amraoui, H. Mekki, and N. Ramdani, “Flatness-based active disturbance rejection control for a wheeled mobile robot subject to slips and external environmental disturbances", *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 95719576, 2020, https://doi.org/10.1016/j.ifacol.2020.12.2443

[80] N. An, Q. Wang, X. Zhao, and Q. Wang, “Differential flatness-based distributed control of underactuated robot swarms", *Appl. Math. Mech.*, vol. 44, no. 10, pp. 17771790, Oct. 2023, https://doi.org/10.1007/s10483-023-3040-8

[81] K. Kaaniche, O. El-Hamrawy, N. Rashid, M. Albekairi, and H. Mekki, “Mobile robot control based on 3D visual servoing: A new approach combining pose estimation by neural network and differential flatness", *Appl. Sci. (Basel)*, vol. 12, no. 12, p. 6167, Jun. 2022, https://doi.org/10.3390/app12126167

[82] J.-C. Ryu and S. K. Agrawal, “Differential flatness-based robust control of mobile robots in the presence of slip", *Int. J. Rob. Res.*, vol. 30, no. 4, pp. 463475, Apr. 2011, https://doi.org/10.1177/0278364910385586

[83] S. R. Sahoo and S. S. Chiddarwar, “Mobile robot control using bond graph and flatness based approach", *Procedia Comput. Sci.*, vol. 133, pp. 213221, 2018, https://doi.org/10.1016/j.procs.2018.07.026

[84] K. S. Yakovlev, A. A. Andreychuk, J. S. Belinskaya, and D. A. Makarov, “Safe interval path planning and flatness-based control for navigation of a mobile robot among static and dynamic obstacles", *Autom. Remote Control*, vol. 83, no. 6, pp. 903918, Jun. 2022, https://doi.org/10.1134/S000511792206008X

[85] G. Rigatos, J. Pomares, P. Siano, M. AL-Numay, M. Abbaszadeh, and G. Cuccurullo, “Flatnessbased control in successive loops for mechatronic motion transmission systems", *Asian J. Control*, vol. 26, no. 6, pp. 28072842, Nov. 2024, https://doi.org/10.1002/asjc.3378

[86] O. Boutalbi, K. Benmahammed, K. Henni, and B. Boukezata, “A high-performance control algorithm based on a curvature-dependent decoupled planning approach and flatness concepts for non-holonomic mobile robots", *Intell. Serv. Robot.*, vol. 12, no. 2, pp. 181196, Apr. 2019. https://doi.org/10.1007/s11370-018-00270-7

[87] K. Katona, H. A. Neamah, and P. Korondi, “Obstacle avoidance and path planning methods for autonomous navigation of mobile robot", *Sensors (Basel)*, vol. 24, no. 11, p. 3573, Jun. 2024, https://doi.org/10.3390/s24113573

[88] X. Cheng, S. Zhang, S. Cheng, Q. Xia, and J. Zhang, “Path-following and obstacle avoidance control of nonholonomic wheeled mobile robot based on deep reinforcement learning", *Appl. Sci. (Basel)*, vol. 12, no. 14, p. 6874, Jul. 2022, https://doi.org/10.3390/app12146874

[89] K. K. A. Farag, H. H. Shehata, and H. M. El-Batsh, “Mobile robot obstacle avoidance based on neural network with a standardization technique", *J. Robot.*, vol. 2021, pp. 114, Nov. 2021, https://doi.org/10.1155/2021/1129872

[90] M. M. J. Samodro, R. D. Puriyanto, and W. Caesarendra, “Artificial Potential Field path planning algorithm in differential drive mobile robot platform for dynamic environment", *International Journal of Robotics and Control Systems*, vol. 3, no. 2, pp. 161170, Mar. 2023, https://doi.org/10.31763/ijrcs.v3i2.944

[91] S. S. Bolbhat, A. S. Bhosale, G. Sakthivel, D. Saravanakumar, R. Sivakumar, and J. Lakshmipathi, “Intelligent obstacle avoiding AGV using vector field histogram and supervisory control", *J. Phys. Conf. Ser.*, vol. 1716, no. 1, p. 012030, Dec. 2020, https://doi.org/10.1088/1742-6596/1716/1/012030

[92] L. A. Zadeh, “Fuzzy sets", *Inf. Contr.*, vol. 8, no. 3, pp. 338353, Jun. 1965. https://doi.org/10.1016/S0019-9958(65)90241-X

[93] T.-W. Wang, H.-P. Huang, and Y.-L. Zhao, “Vision-guided autonomous robot navigation in realistic 3D dynamic scenarios", *Appl. Sci. (Basel)*, vol. 15, no. 5, p. 2323, Feb. 2025. https://doi.org/10.3390/app15052323

[94] Y. Ou, Y. Cai, Y. Sun, and T. Qin, “Autonomous navigation by mobile robot with sensor fusion based on deep reinforcement learning", *Sensors (Basel)*, vol. 24, no. 12, p. 3895, Jun. 2024, https://doi.org/10.3390/s24123895

[95] Y. Tang, M. A. Zakaria, and M. Younas, “Path planning trends for autonomous mobile robot navigation: A review", *Sensors (Basel)*, vol. 25, no. 4, p. 1206, Feb. 2025, https://doi.org/10.3390/s25041206

[96] W. Ding, J.-X. Zhang, and P. Shi, “Adaptive fuzzy control of wheeled mobile robots with prescribed trajectory tracking performance", *IEEE Trans. Fuzzy Syst.*, vol. 32, no. 8, pp. 45104521, Aug. 2024, https://doi.org/10.1109/TFUZZ.2024.3401691

[97] D. Luo, X. Huang, Y. Huang, M. Miao, and X. Gao, “Optimal trajectory planning for wheeled robots (OTPWR): A globally and dynamically optimal trajectory planning method for wheeled mobile robots", *Machines*, vol. 12, no. 10, p. 668, Sep. 2024, https://doi.org/10.3390/machines12100668

[98] J. T. Licardo, M. Domjan, and T. Orehovaki, “Intelligent roboticsA systematic review of emerging technologies and trends", *Electronics (Basel)*, vol. 13, no. 3, p. 542, Jan. 2024, https://doi.org/10.3390/electronics13030542

[99] V. Tinoco, M. F. Silva, F. N. Santos, R. Morais, S. A. Magalhães, and P. M. Oliveira, “A review of advanced controller methodologies for robotic manipulators", *Int. J. Dyn. Contr.*, vol. 13, no. 1, Jan. 2025., https://doi.org/10.1007/s40435-024-01533-1

[100] T. Lackner, J. Hermann, C. Kuhn, and D. Palm, “Review of autonomous mobile robots in intralogistics: state-of-the-art, limitations and research gaps", *Procedia CIRP*, vol. 130, pp. 930935, 2024, https://doi.org/10.1016/j.procir.2024.10.187

[101] F. J. Naranjo-Campos, A. De Matías-Martínez, J. G. Victores, J. A. Gutiérrez Dueñas, A. Alcaide, and C. Balaguer, “Assistance in picking up and delivering objects for individuals with reduced mobility using the TIAGo robot", *Appl. Sci. (Basel)*, vol. 14, no. 17, p. 7536, Aug. 2024, https://doi.org/10.3390/app14177536

[102] S. Amertet, G. Gebresenbet, and H. M. Alwan, “Optimizing the performance of a wheeled mobile robots for use in agriculture using a linear-quadratic regulator", *Rob. Auton. Syst.*, vol. 174, no. 104642, p. 104642, Apr. 2024, https://doi.org/10.1016/j.robot.2024.104642

[103] S. Amertet and G. Gebresenbet, “Collision avoidance for wheeled mobile robots in smart agricultural systems using control barrier function quadratic programming", *Appl. Sci. (Basel)*, vol. 15, no. 5, p. 2450, Feb. 2025, https://doi.org/10.3390/app15052450

[104] J. Moritz, M. Musa, and U. Wejinya, “Design and modeling of a two-wheeled differential drive robot", *Int. J. Control Autom. Syst.*, vol. 22, no. 7, pp. 22732282, Jul. 2024., https://doi.org/10.1007/s12555-023-0288-8

[105] P. Szelg, S. Dudzik, and A. Podsiedlik, “Investigation on the mobile wheeled robot in terms of energy consumption, travelling time and path matching accuracy", *Energies*, vol. 16, no. 3, p. 1210, Jan. 2023, https://doi.org/10.3390/en16031210

[106] C. Zhu, B. Li, C. Zhao, and Y. Wang, “Trajectory tracking control of car-like mobile robots based on extended state observer and backstepping control", *Electronics (Basel)*, vol. 13, no. 8, p. 1563, Apr. 2024, https://doi.org/10.3390/electronics13081563

[107] L. Tagliavini, G. Colucci, A. Botta, P. Cavallone, L. Baglieri, and G. Quaglia, “Wheeled mobile robots: State of the art overview and kinematic comparison among three omnidirectional locomotion strategies", *J. Intell. Robot. Syst.*, vol. 106, no. 3, p. 57, Oct. 2022, https://doi.org/10.1007/s10846-022-01745-7

[108] J. Luo et al., “Robotics perception and control: Key technologies and applications", *Micromachines (Basel)*, vol. 15, no. 4, p. 531, Apr. 2024. https://doi.org/10.3390/mi15040531

[109] C. Medina Sánchez, M. Zella, J. Capitán, and P. J. Marrón, “From perception to navigation in environments with persons: An indoor evaluation of the state of the art", *Sensors (Basel)*, vol. 22, no. 3, p. 1191, Feb. 2022. https://doi.org/10.3390/s22031191

[110] Y. Liu, S. Wang, Y. Xie, T. Xiong, and M. Wu, “A review of sensing technologies for indoor autonomous mobile robots", *Sensors (Basel)*, vol. 24, no. 4, p. 1222, Feb. 2024. https://doi.org/10.3390/s24041222

[111] K. He, H. Ding, N. Xu, and K. Guo, “Wheel odometry with deep learning-based error prediction model for vehicle localization", *Appl. Sci. (Basel)*, vol. 13, no. 9, p. 5588, Apr. 2023. https://doi.org/10.3390/app13095588

[112] B. M. Al-Hadithi and C. Pastor, “Cost-effective localization of mobile robots using ultrasound beacons and Differential Time-of-Flight measurement", *Appl. Sci. (Basel)*, vol. 14, no. 17, p. 7597, Aug. 2024. https://doi.org/10.3390/app14177597

[113] J.-W. Jung, J.-S. Park, T.-W. Kang, J.-G. Kang, and H.-W. Kang, “Mobile robot path planning using a laser range finder for environments with transparent obstacles", *Appl. Sci. (Basel)*, vol. 10, no. 8, p. 2799, Apr. 2020. https://doi.org/10.3390/app10082799

[114] K. R. Park, S. Lee, and E. Kim, Eds., Visual and camera sensors. *Mdpi AG*, 2021. https://doi.org/10.3390/books978-3-0365-1583-0

[115] J. Rao, H. Bian, X. Xu, and J. Chen, “Autonomous visual navigation system based on a single camera for floor-sweeping robot", *Appl. Sci. (Basel)*, vol. 13, no. 3, p. 1562, Jan. 2023. https://doi.org/10.3390/app13031562

[116] A. Marroquín, G. Garcia, E. Fabregas, E. Aranda-Escolástico, and G. Farias, “Mobile robot navigation based on embedded computer vision", *Mathematics*, vol. 11, no. 11, p. 2561, Jun. 2023. https://doi.org/10.3390/math11112561

[117] G. Farias, E. Fabregas, E. Torres, G. Bricas, S. Dormido-Canto, and S. Dormido, “A distributed vision-based navigation system for khepera IV mobile robots", *Sensors (Basel)*, vol. 20, no. 18, p. 5409, Sep. 2020. https://doi.org/10.3390/s20185409

[118] F. Wang et al., “Object-based reliable visual navigation for mobile robot", *Sensors (Basel)*, vol. 22, no. 6, p. 2387, Mar. 2022. https://doi.org/10.3390/s22062387

[119] B. Al-Tawil, A. Candemir, M. Jung, and A. Al-Hamadi, “Mobile robot navigation with enhanced 2D mapping and multi-sensor fusion", *Sensors (Basel)*, vol. 25, no. 8, p. 2408, Apr. 2025. https://doi.org/10.3390/s25082408

[120] C. Bräuer-Burchardt, R. Ramm, P. Kühmstedt, and G. Notni, “The duality of ray-based and pinhole-camera modeling and 3D measurement improvements using the ray-based model", *Sensors (Basel)*, vol. 22, no. 19, p. 7540, Oct. 2022. https://doi.org/10.3390/s22197540

[121] C. Veiga Almagro et al., “Monocular robust depth estimation vision system for robotic tasks interventions in metallic targets", *Sensors (Basel)*, vol. 19, no. 14, p. 3220, Jul. 2019. https://doi.org/10.3390/s19143220

[122] A. Zaremba and S. Nitkiewicz, “Distance estimation with a stereo camera and accuracy determination", *Appl. Sci. (Basel)*, vol. 14, no. 23, p. 11444, Dec. 2024. https://doi.org/10.3390/app142311444

[123] V. Uinskis, M. Nowicki, A. Dzedzickis, and V. Buinskas, “Sensor-fusion based navigation for autonomous mobile robot", *Sensors (Basel)*, vol. 25, no. 4, Feb. 2025. https://doi.org/10.3390/s25041248

[124] F. F. R. Merveille, B. Jia, Z. Xu, and B. Fred, “Advancements in sensor fusion for underwater SLAM: A review on enhanced navigation and environmental perception", *Sensors (Basel)*, vol. 24, no. 23, p. 7490, Nov. 2024. https://doi.org/10.3390/s24237490

[125] N. Y. Ko and T.-Y. Kuc, “Fusing range measurements from ultrasonic beacons and a laser range finder for localization of a mobile robot", *Sensors (Basel)*, vol. 15, no. 5, pp. 1105011075, May 2015. https://doi.org/10.3390/s150511050

[126] Y.-H. Huang and C.-T. Lin, “Indoor localization method for a mobile robot using LiDAR and a dual AprilTag", *Electronics (Basel)*, vol. 12, no. 4, p. 1023, Feb. 2023. https://doi.org/10.3390/electronics12041023

[127] G. Grunwald, A. Cieko, K. Krasuski, and D. Tanajewski, “Optimal global positioning system/European Geostationary Navigation Overlay Service positioning model using smartphone", *Appl. Sci. (Basel)*, vol. 14, no. 5, p. 1840, Feb. 2024. https://doi.org/10.3390/app14051840

[128] D. Wu, Z. Ma, W. Xu, H. He, and Z. Li, “Visual odometry based on improved Oriented features from accelerated segment test and Rotated binary robust independent elementary features", *World Electric Veh. J.*, vol. 15, no. 3, p. 123, Mar. 2024. https://doi.org/10.3390/wevj15030123

[129] B. Sadeghi Bigham, O. Abbaszadeh, M. Zahedi-Seresht, S. Khosravi, and E. Zarezadeh, “Optimal robot pose estimation using scan matching by turning function", *Mathematics*, vol. 11, no. 6, p. 1449, Mar. 2023. https://doi.org/10.3390/math11061449

[130] M. Alatise and G. Hancke, “Pose estimation of a mobile robot based on fusion of IMU data and vision data using an extended Kalman filter", *Sensors (Basel)*, vol. 17, no. 10, p. 2164, Sep. 2017. https://doi.org/10.3390/s17102164

[131] C. Zuo, D. Xie, L. Wu, X. Tang, and H. Zhang, “An improved adaptive Monte Carlo localization algorithm integrated with a virtual motion model", *Sensors (Basel)*, vol. 25, no. 8, p. 2471, Apr. 2025. https://doi.org/10.3390/s25082471

[132] H. Mansur, M. Gadhwal, J. E. Abon, and D. Flippo, “Mapping for autonomous navigation of agricultural robots through crop rows using UAV", *Agriculture*, vol. 15, no. 8, p. 882, Apr. 2025. https://doi.org/10.3390/agriculture15080882

[133] J. Dai, D. Li, Y. Li, J. Zhao, W. Li, and G. Liu, “Mobile robot localization and mapping algorithm based on the fusion of image and laser point cloud", *Sensors (Basel)*, vol. 22, no. 11, p. 4114, May 2022. https://doi.org/10.3390/s22114114

[134] Y. Lu, H. Wang, J. Sun, and J. A. Zhang, “Enhanced simultaneous localization and mapping algorithm based on deep learning for highly dynamic environment", *Sensors (Basel)*, vol. 25, no. 8, p. 2539, Apr. 2025. https://doi.org/10.3390/s25082539

[135] S. Huang and H.-Z. Huang, “A frame-to-frame Scan Matching algorithm for 2D lidar based on attention", *Appl. Sci. (Basel)*, vol. 12, no. 9, p. 4341, Apr. 2022. https://doi.org/10.3390/app12094341

[136] H. Wu, Y. Liu, C. Wang, and Y. Wei, “An effective 3D instance map reconstruction method based on RGBD images for indoor scene", *Remote Sens. (Basel)*, vol. 17, no. 1, p. 139, Jan. 2025. https://doi.org/10.3390/rs17010139

[137] B. Z. Türkkol, N. Altunta, and S. Çekirdek Yavuz, “A smooth global path planning method for Unmanned Surface Vehicles using a novel combination of rapidly exploring random tree and Bézier Curves", *Sensors (Basel)*, vol. 24, no. 24, Dec. 2024. https://doi.org/10.3390/s24248145

[138] B. Lu et al., “Hybrid path planning combining potential field with sigmoid curve for autonomous driving", *Sensors (Basel)*, vol. 20, no. 24, p. 7197, Dec. 2020. https://doi.org/10.3390/s20247197

[139] P. S. Trakas, S. I. Anogiatis, and C. P. Bechlioulis, “Trajectory tracking with obstacle avoidance for nonholonomic mobile robots with Diamond-shaped velocity constraints and output performance specifications", *Sensors (Basel)*, vol. 24, no. 14, p. 4636, Jul. 2024. https://doi.org/10.3390/s24144636

[140] W. Chen, F. Liu, and H. Zhao, “Trajectory-tracking control of unmanned vehicles based on adaptive variable parameter MPC", *Appl. Sci. (Basel)*, vol. 14, no. 16, p. 7285, Aug. 2024. https://doi.org/10.3390/app14167285

[141] K. Worthmann, M. W. Mehrez, M. Zanon, G. K. I. Mann, R. G. Gosine, and M. Diehl, “Model predictive control of nonholonomic mobile robots without stabilizing constraints and costs", *IEEE Trans. Control Syst. Technol.*, vol. 24, no. 4, pp. 13941406, Jul. 2016. https://doi.org/10.1109/TCST.2015.2488589

[142] B. Ahmadi, M. W. Mehrez, W. W. Melek, and A. Khajepour, “Model Predictive Control for Reliable Path Following with Application to the Autonomous Vehicle and Considering Different Vehicle Models", *in 2021 5th International Conference on Vision, Image and Signal Processing (ICVISP)*, 2021, pp. 2732. https://doi.org/10.1109/ICVISP54630.2021.00014

[143] T. Gameiro et al., “Evaluation of PID-based algorithms for UGVs", *Algorithms*, vol. 18, no. 2, p. 63, Jan. 2025. https://doi.org/10.3390/a18020063

[144] L. Martins, C. Cardeira, and P. Oliveira, “Linear quadratic regulator for trajectory tracking of a quadrotor", *IFAC-PapersOnLine*, vol. 52, no. 12, pp. 176181, 2019. https://doi.org/10.1016/j.ifacol.2019.11.195

[145] X. Fan, J. Wang, H. Wang, L. Yang, and C. Xia, “LQR trajectory tracking control of unmanned wheeled tractor based on improved quantum genetic algorithm", *Machines*, vol. 11, no. 1, p. 62, Jan. 2023. https://doi.org/10.3390/machines11010062

[146] B. Jiang, J. Li, and S. Yang, “An improved sliding mode approach for trajectory following control of nonholonomic mobile AGV", *Sci. Rep.*, vol. 12, no. 1, p. 17763, Oct. 2022. https://doi.org/10.1038/s41598-022-22697-w

[147] A. A. Kabanov, S. Stoyanov, and E. N. Kabanova, Trajectory-tracking control of mobile robot via feedback linearization, *in Advances in Intelligent Systems and Computing, Cham: Springer International Publishing*, 2018, pp. 3241. https://doi.org/10.1007/978-3-319-68324-9_4

[148] T. Pavlic et al., “Cognitive model of the closed environment of a mobile robot based on measurements", *Appl. Sci. (Basel)*, vol. 11, no. 6, p. 2786, Mar. 2021. https://doi.org/10.3390/app11062786

[149] J. Gao, N. Liu, H. Li, Z. Li, C. Xie, and Y. Gou, “Reinforcement learning decision-making for autonomous vehicles based on Semantic Segmentation", *Appl. Sci. (Basel)*, vol. 15, no. 3, p. 1323, Jan. 2025. https://doi.org/10.3390/app15031323

[150] A. tefek, V. T. Pham, V. Krivanek, and K. L. Pham, “Optimization of fuzzy logic controller used for a differential drive wheeled mobile robot", *Appl. Sci. (Basel)*, vol. 11, no. 13, p. 6023, Jun. 2021. https://doi.org/10.3390/app11136023

[151] R. Hoseinnezhad, “A comprehensive review of Deep learning techniques in mobile robot path planning: Categorization and analysis", *Appl. Sci. (Basel)*, vol. 15, no. 4, p. 2179, Feb. 2025. https://doi.org/10.3390/app15042179

[152] M. Siwek, J. Panasiuk, L. Baranowski, W. Kaczmarek, P. Prusaczyk, and S. Borys, “Identification of differential drive robot dynamic model parameters", *Materials (Basel)*, vol. 16, no. 2, p. 683, Jan. 2023. https://doi.org/10.3390/ma16020683

[153] B. Charlet, J. Lévine, and R. Marino, “On dynamic feedback linearization", *Syst. Control Lett.*, vol. 13, no. 2, pp. 143151, Aug. 1989. https://doi.org/10.1016/0167-6911(89)90031-5

[154] J. Baillieul and T. Samad, Eds., Encyclopedia of systems and control. *London: Springer London*, 2013. https://doi.org/10.1007/978-1-4471-5102-9

[155] A. Isidori, “Control of nonlinear systems via dynamic state-feedback", *in Algebraic and Geometric Methods in Nonlinear Control Theory, Dordrecht: Springer Netherlands*, 1986, pp. 121145. https://doi.org/10.1007/978-94-009-4706-1_8

[156] S. Singh, “Decoupling of invertible nonlinear systems with state feedback and precompensation", *IEEE Trans. Automat. Contr.*, vol. 25, no. 6, pp. 12371239, Dec. 1980. https://doi.org/10.1109/TAC.1980.1102546

[157] G. Klanar, A. Zdear, S. Blai, and I. krjanc, Chapter 3 - Control of Wheeled Mobile Systems, *in Wheeled Mobile Robotics*, G. Klanar, A. Zdear, S. Blai, and I. krjanc, Eds. Butterworth-Heinemann, 2017, pp. 61159. https://doi.org/10.1016/B978-0-12-804204-5.00003-2

[158] R. Saatchi, “Fuzzy logic concepts, developments and implementation", *Information (Basel)*, vol. 15, no. 10, p. 656, Oct. 2024. https://doi.org/10.3390/info15100656

[159] C. Dumitrescu, P. Ciotirnae, and C. Vizitiu, “Fuzzy logic for intelligent control system using soft computing applications", *Sensors (Basel)*, vol. 21, no. 8, p. 2617, Apr. 2021. https://doi.org/10.3390/s21082617

[160] A. tefek, V. T. Pham, V. Krivanek, and K. L. Pham, “Optimization of fuzzy logic controller used for a differential drive wheeled mobile robot", *Appl. Sci. (Basel)*, vol. 11, no. 13, p. 6023, Jun. 2021. https://doi.org/10.3390/app11136023

[161] Y. Lambat, N. Ayres, L. Maglaras, and M. A. Ferrag, “A Mamdani type Fuzzy Inference System to calculate employee susceptibility to phishing attacks", *Appl. Sci. (Basel)*, vol. 11, no. 19, p. 9083, Sep. 2021. https://doi.org/10.3390/app11199083

[162] L. C. Sousa et al., “Obstacle avoidance technique for mobile robots at autonomous human-robot collaborative warehouse environments", *Sensors (Basel)*, vol. 25, no. 8, p. 2387, Apr. 2025. https://doi.org/10.3390/s25082387

[163] M. Candeloro, A. M. Lekkas, and A. J. Sørensen, “A Voronoi-diagram-based dynamic path-planning system for underactuated marine vessels", *Control Eng. Pract.*, vol. 61, pp. 4154, Apr. 2017. http://dx.doi.org/10.1016/j.conengprac.2017.01.007

[164] S. Erke, D. Bin, N. Yiming, Z. Qi, X. Liang, and Z. Dawei, “An improved A-Star based path planning algorithm for autonomous land vehicles", *Int. J. Adv. Robot. Syst.*, vol. 17, no. 5, p. 172988142096226, Sep. 2020. http://dx.doi.org/10.1177/1729881420962263

[165] H. Zhang, Y. Tao, and W. Zhu, “Global path planning of unmanned surface vehicle based on improved A-star algorithm", *Sensors (Basel)*, vol. 23, no. 14, Jul. 2023. http://dx.doi.org/10.3390/s23146647

[166] H. Park and J.-H. Lee, “-spline curve fitting based on adaptive curve refinement using dominant points", *Comput. Aided Des.*, vol. 39, no. 6, pp. 439451, Jun. 2007. http://dx.doi.org/10.1016/j.cad.2006.12.006

[167] D. Xiang, H. Lin, J. Ouyang, and D. Huang, “Combined improved A* and greedy algorithm for path planning of multi-objective mobile robot", *Sci. Rep.*, vol. 12, no. 1, p. 13273, Aug. 2022. http://dx.doi.org/10.1038/s41598-022-17684-0

[168] C. Zhang, X. Yang, R. Zhou, and Z. Guo, “A path planning method based on improved A* and fuzzy control DWA of underground mine vehicles", *Appl. Sci. (Basel)*, vol. 14, no. 7, p. 3103, Apr. 2024. https://doi.org/10.3390/app14073103

[169] L. Liu, B. Wang, and H. Xu, “Research on path-planning algorithm integrating optimization A-star algorithm and artificial potential field method", *Electronics (Basel)*, vol. 11, no. 22, p. 3660, Nov. 2022. https://doi.org/10.3390/electronics11223660

[170] A. Ravankar, A. A. Ravankar, Y. Kobayashi, Y. Hoshino, and C.-C. Peng, “Path smoothing techniques in robot navigation: State-of-the-art, current and future challenges", *Sensors (Basel)*, vol. 18, no. 9, p. 3170, Sep. 2018. https://doi.org/10.3390/s18093170

[171] Z. Durakl and V. Nabiyev, “A new approach based on Bezier curves to solve path planning problems for mobile robots", *J. Comput. Sci.*, vol. 58, no. 101540, p. 101540, Feb. 2022. https://doi.org/10.1016/j.jocs.2021.101540

[172] Y. Zhu and T. Lu, “A fuzzy-based improved dynamic window approach for path planning of mobile robot", *in Intelligent Robotics and Applications, Singapore: Springer Nature Singapore*, 2023, pp. 586597. https://doi.org/10.1007/978-981-99-6492-5_50

[173] P. Guo, H. Shi, S. Wang, L. Tang, and Z. Wang, “An ROS architecture for autonomous mobile robots with UCAR platforms in smart restaurants", *Machines*, vol. 10, no. 10, p. 844, Sep. 2022. https://doi.org/10.3390/machines10100844

[174] A. Bonci, F. Gaudeni, M. C. Giannini, and S. Longhi, “Robot Operating System 2 (ROS2)-based frameworks for increasing robot autonomy: A survey", *Appl. Sci. (Basel)*, vol. 13, no. 23, p. 12796, Nov. 2023. https://doi.org/10.3390/app132312796

List of Publications

Research Articles:

1. **S. Louda**, N. Karkar, F. Seghir, and O. Boutalbi, “Fuzzy Dynamic Feedback Linearization for Efficient Mobile Robot Trajectory Tracking and Obstacle Avoidance in Autonomous Navigation", *International Journal of Robotics and Control Systems*, vol. 5,no. 2, pp. 881-901, Mar. 2025. https://doi.org/10.31763/ijrcs.v5i2.1780

Conference Papers:

- **International Conferences:**

1. **S. Louda**, N. Karkar, F. Seghir, and O. Boutalbi, “Mobile robot path planning algorithm based on A* algorithm and dynamic window approach for autonomous navigation", *in 2024 International Conference on Advances in Electrical and Communication Technologies (ICAECOT)*, Setif, Algeria, 2024, pp. 1-6. https://doi.org/10.1109/ICAECOT62402.2024.10828660
2. **S. Louda**, N. Karkar, F. Seghir, and S. Refoufi, “Mobile robot path planning based on A-Star algorithm and artificial potential field method for autonomous navigation", *in 2024 12th International Conference on Systems and Control (ICSC)*, Batna, Algeria, 2024, pp. 441-446. https://doi.org/10.1109/ICSC63929.2024.10928867
3. **S. Louda**, N. Karkar, and F. Seghir, “Autonomous Navigation of a Differential Mobile Robot using Depth Camera Sensor-based ROS", *in 2023 International Conference on Electronics Engineering and Telecommunications (ICEET)*, Bordj Bou Arreridj, Algeria, Nov. 2023, pp. 17.
4. N. Karkar, **S. Louda**, “Visual object tracking using the bat algorithm", *in 2022 International Conference on Energy and Material Sciences (ICEMS)*, Skikda, Algeria, Nov. 2022

- **National Conferences:**

1. **S. Louda**, N. Karkar, F. Seghir, and O. Boutalbi, “A Comparative Performance Analysis of PID, MPC, and DWA Controllers for Mobile Robot Path Planning and Navigation", *The First National Conference on Innovation in Electronics and Telecommunication Technologies (NCIETT’2025)*, Tebessa, Algeria, Sep, 2025.
2. **S. Louda**, N. Karkar, F. Seghir, and O. Boutalbi, “Local Path Planning for Mobile Robot Navigation Based on a Fuzzy-Dynamic Window Approach", *The First National Conference on AI Innovations Across Disciplines (AYAD’2025)*, Mila, Algeria, Sept-Oct, 2025.
3. **S. Louda**, N. Karkar, and F. Seghir, “Wall-Following Control in Mobile Robots Using Sonar, FSM, and PID", *The National Conference on Mechatronic Engineering (NCME’2025)*, Boumerdes, Algeria, Oct, 2025.
4. **S. Louda**, N. Karkar, and F. Seghir and O. Boutalbi, “Fuzzy Logic Control Approach for Mobile Robot Navigation in Unknown Environments", *The First National Conference on Information Technology and Intelligent Systems (NCITIS’2025)*, Batna, Algeria, Oct, 2025.

A

Prove 1: Kinematic Model of Differential Wheeled Mobile Robot

From (3.13), the linear velocity v can be written as the following:

$$v = \omega R \tag{A.1}$$

where R denotes the radius of both robot wheels. Then one can get this:

$$\begin{cases} v_R = \omega \left(R + \dfrac{L}{2} \right) \\ v_L = \omega \left(R - \dfrac{L}{2} \right) \end{cases} \tag{A.2}$$

$$R = \frac{L}{2} \left(\frac{v_R + v_L}{v_R - v_L} \right) \tag{A.3}$$

Finally, from the above demonstrations, the linear velocity can be written as

$$v = \frac{v_R + v_L}{2} \tag{A.4}$$

The angular velocity depends on the difference between the linear velocity of right and left velocities, respectively.

$$v_R - v_L = \omega \left(R + \frac{L}{2} \right) - \omega \left(R - \frac{L}{2}) \right) \tag{A.5}$$

$$v_R - v_L = \omega L \tag{A.6}$$

After that, the angular velocity ω can be written as:

$$\omega = \frac{v_R - v_L}{L} \tag{A.7}$$

B

ROS Network Communication Between Master and Slave Nodes

In a ROS (Robot Operating System) network, communication between the ROS Master and slave nodes is essential for enabling distributed robotic systems, as illustrated in Figure B.1a. In this setup, `roscore` runs on the ROS Master, while the control and path planning code is executed in MATLAB on a slave machine. When the MATLAB node connects, it registers with the Master, which handles node discovery and connection setup. Actual data exchange occurs directly between nodes. To ensure proper communication, `ROS_MASTER_URI` in MATLAB must point to the Master's IP address, and `ROS_IP` must be set to the slave's IP address. Proper network configuration and time synchronization are essential to ensure stable communication and prevent connection failures.

ROS Network Setup

1. Remote (Master) PC Add to `.bashrc` file:

```
~$ export ROS_HOSTNAME=192.168.0.121   # Example IP of master PC
~$ export ROS_MASTER_URI=http://192.168.0.121:11311
```

2. Netbook (Slave) PC Add to `.bashrc` file:

```
~$ export ROS_HOSTNAME=192.168.0.200   # Example IP of slave PC
~$ export ROS_MASTER_URI=http://192.168.0.121:11311
```

3. Commands to Run

- On the remote (master) PC:

```
~$ roscore
```

This command starts the ROS master and parameter server.

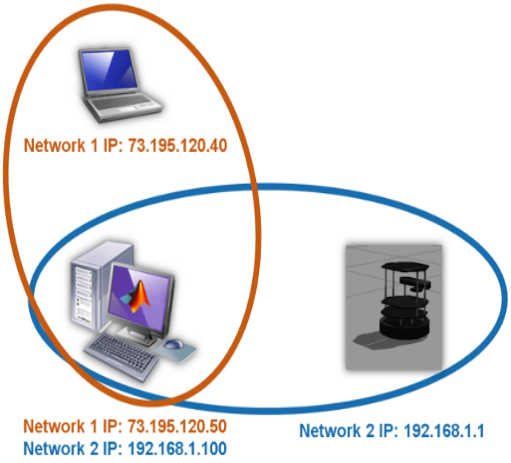
- On the netbook (slave) PC:

```
~$ rosrun rosaria rosaria
```

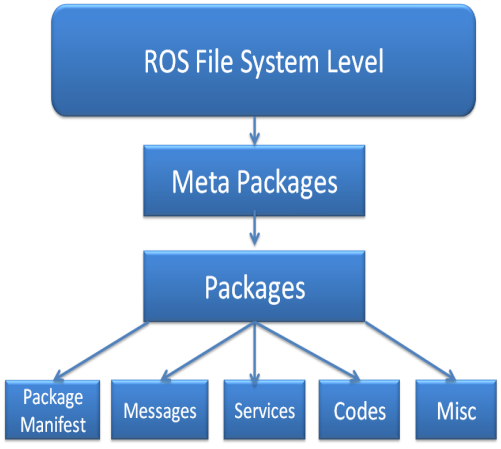
This command runs the `rosaria` node that interfaces between the robot and computer. It publishes the velocity command topic `/RosAria/cmd_vel` and the odometry topic `/RosAria/pose`.

Further Resources

- `https://www.ros.org/` Official Site for ROS Middleware.
- `https://wiki.ros.org/Documentation` Official documentation and package descriptions.
- `https://www.turtlebot.com/turtlebot2/` Official homepage for the TurtleBot2 mobile robot.
- `https://wiki.ros.org/ROSARIA/Tutorials/How%20to%20use%20ROSARIA` Tutorial for using the ROSARIA node in ROS.



(a) ROS network comminication.



(b) ROS computational graph showing nodes, topics, and services.

Figure B.1: Network communication and Robot Operating System (ROS) architecture.

C

Eta3-Splines for the Smooth Path Generation

The parameters α_i and β_i of the Eta3-Spline used for path smoothing are given as follows:

$\alpha_0 = X_A$; $\alpha_1 = \eta_1 cos(\theta_A)$; $\alpha_2 = \frac{1}{2}\eta_2 cos(\theta_A) - \frac{1}{2}\eta_1^2 k_A sin(\theta_A)$; $\alpha_3 = \frac{1}{6}\eta_5 cos(\theta_A) - \frac{1}{6}(\eta_1^3 k'_A + 3\eta_1\eta_3 k_A) sin(\theta_A)$; $\alpha_4 = 35(X_B - X_A) - (20\eta_1 + 5\eta_3 + \frac{2}{3}\eta_5) cos(\theta_A) + (5\eta_1^2 k_A + \frac{2}{3}\eta_1^3 k'_A + 2\eta_1\eta_3 k_A) sin(\theta_A) - (15\eta_2 - \frac{5}{2}\eta_4 + \frac{1}{6}\eta_6) cos(\theta_B) - (\frac{5}{2}\eta_2^2 k_B - \frac{1}{6}\eta_2^3 k'_B - \frac{1}{2}\eta_2\eta_4 k_B) sin(\theta_B)$; $\alpha_5 = -84(X_B - X_A) + (45\eta_1 + 10\eta_3 + \eta_5) cos(\theta_A) - (10\eta_1^2 k_A + \eta_1^3 k'_A + 3\eta_1\eta_3 k_A) sin(\theta_A) + (39\eta_2 - 7\eta_4 + \frac{1}{2}\eta_6) cos(\theta_B) + (7\eta_2^2 k_B - \frac{1}{2}\eta_2^3 k'_B - \frac{3}{2}\eta_2\eta_4 k_B) sin(\theta_B)$; $\alpha_6 = 70(X_B - X_A) - (36\eta_1 + \frac{15}{2}\eta_3 + \frac{2}{3}\eta_5) cos(\theta_A) + (\frac{15}{2}\eta_1^2 k_A + \frac{2}{3}\eta_1^3 k'_A + 2\eta_1\eta_3 k_A) sin(\theta_A) - (34\eta_2 - \frac{13}{2}\eta_4 + \frac{1}{2}\eta_6) cos(\theta_B) - (\frac{13}{2}\eta_2^2 k_B - \frac{1}{2}\eta_2^3 k'_B - \frac{3}{2}\eta_2\eta_4 k_B) sin(\theta_B)$; $\alpha_7 = -20(X_B - X_A) + (10\eta_1 + 2\eta_3 + \frac{1}{6}\eta_5) cos(\theta_A) - (2\eta_1^2 k_A + \frac{1}{6}\eta_1^3 k'_A + \frac{1}{2}\eta_1\eta_3 k_A) sin(\theta_A) + (10\eta_2 - 2\eta_4 + \frac{1}{6}\eta 6) cos(\theta_B) + 2\eta_2^2 k_B - \frac{1}{6}\eta_2^3 k'_B - \frac{1}{2}\eta_2\eta_4 k_B) sin(\theta_B)$; $\beta_0 = Y_A$; $\beta_1 = \eta_1 sin(\theta_A)$; $\beta_2 = \frac{1}{2}\eta_2 sin(\theta_A) + \frac{1}{2}\eta_1^2 k_A cos(\theta_A)$; $\beta_3 = \frac{1}{6}\eta_5 sin(\theta_A) + \frac{1}{6}(\eta_1^3 k'_A + 3\eta_1\eta_3 k_A) cos(\theta_A)$; $\beta_4 = 35(X_B - X_A) - (20\eta_1 + 5\eta_3 + \frac{2}{3}\eta_5) sin(\theta_A) - (5\eta_1^2 k_A + \frac{2}{3}\eta_1^3 k'_A + 2\eta_1\eta_3 k_A) cos(\theta_A) - (15\eta_2 - \frac{5}{2}\eta_4 + \frac{1}{6}\eta_6) sin(\theta_B) + (\frac{5}{2}\eta_2^2 k_B - \frac{1}{6}\eta_2^3 k'_B - \frac{1}{2}\eta_2\eta_4 k_B) cos(\theta_B)$; $\beta_5 = -84(X_B - X_A) + (45\eta_1 + 10\eta_3 + \eta_5) sin(\theta_A) + (10\eta_1^2 k_A + \eta_1^3 k'_A + 3\eta_1\eta_3 k_A) cos(\theta_A) + (39\eta_2 - 7\eta_4 + \frac{1}{2}\eta_6) sin(\theta_B) + (7\eta_2^2 k_B - \frac{1}{2}\eta_2^3 k'_B - \frac{3}{2}\eta_2\eta_4 k_B) cos(\theta_B)$; $\beta_6 = 70(X_B - X_A) + (36\eta_1 + \frac{15}{2}\eta_3 + \frac{2}{3}\eta_5) sin(\theta_A) + (\frac{15}{2}\eta_1^2 k_A + \frac{2}{3}\eta_1^3 k'_A + 2\eta_1\eta_3 k_A) cos(\theta_A) + (34\eta_2 - \frac{13}{2}\eta_4 + \frac{1}{2}\eta_6) sin(\theta_B) - (\frac{13}{2}\eta_2^2 k_B - \frac{1}{2}\eta_2^3 k'_B - \frac{3}{2}\eta_2\eta_4 k_B) cos(\theta_B)$; $\beta_7 = -20(X_B - X_A) + (10\eta_1 + 2\eta_3 + \frac{1}{6}\eta_5) sin(\theta_A) + (2\eta_1^2 k_A + \frac{1}{6}\eta_1^3 k'_A + 0.5\eta_1\eta_3 k_A) cos(\theta_A) + (10\eta_2 - 2\eta_4 + \frac{1}{6}\eta_6) sin(\theta_B) - (2\eta_2^2 k_B - \frac{1}{6}\eta_2^3 k'_B - \frac{1}{2}\eta_2\eta_4 k_B) cos(\theta_B)$;

ملخص

تهدف هذه الأطروحة إلى تطوير الملاحة الذاتية للروبوتات المتنقلة ذات العجلات من خلال معالجة التحديات الأساسية في تخطيط الحركة والتحكم في البيئات الساكنة والديناميكية على حد سواء. تم تطوير إطار تخطيط مسار هجين يجمع بين خوارزمية A* مُحسَّن لتوليد المسار العالمي ونهج النافذة الديناميكية Dynamic Window Approach (DWA) معدل لتجنب العوائق المحلية في الزمن الحقيقي. تعمل خوارزمية A* التكيفية على زيادة كفاءة البحث من خلال دالة إرشادية ديناميكية، وتبسيط المسار باستخدام إستراتجية إزالة النقاط الزائدة، وتنعيم بواسطة (Eta-Spline) و (Cubic B-Spline) في التخطيط المحلي، تم توظيف نظام المنطق الضبابي لتحسين دالة التقييم في خوارزمية DWA بشكل ديناميكي، مع اقتراح صيغة معدّلة للبيئات الديناميكية. يؤدي هذا التحسين إلى تقليل زمن التخطيط، وتحسين نعومة المسار، والتغلب على مشكلة الوقوع في (Local Minimum) . أما في مجال التحكم، فتقترح الأطروحة استراتيجية تتبع مسار قوية تعتمد على دمج خطيّة التغذية الراجعة الديناميكية Dynamic Feedback Linearization (DFL) مع التحكم بالمنطق الضبابي Fuzzy Logic Control (FLC) ومن خلال استغلال خاصية الانبساط التفاضلي للنظام، تم اشتقاق قانون التحكم في فضاء المخرجات المسطحة لضمان دقة تتبع المسار وتحقيق سلوكيات فعّالة لتجنب العوائق. يعزز الدمج بين نموذجية DFL ونهج المنطق الضبابي القائم على البيانات دقة التتبع والمتانة ومقاومة الاضطرابات الخارجية. تم التحقق من فعالية الطرق المقترحة من خلال محاكاة مكثفة وتجارب واقعية، حيث أظهرت النتائج تحسينات ملموسة في كفاءة الملاحة ودقة التحكم وأداء تجنب العوائق مقارنة بالأساليب التقليدية.

الكلمات المفتاحية: الملاحة الذاتية, تخطيط المسار, خوارزمية(A-Star) ,نهج نافذة ديناميكية (DWA) ,التحكم بالمنطق الضبابي(FLC),الخطية بالتغذية الراجعة الديناميكية (DFL).

Abstract

This thesis aims to advance autonomous navigation for wheeled mobile robots by addressing fundamental challenges in motion planning and control within both static and dynamic environments. A hybrid path-planning framework is developed, combining an enhanced adaptive A* algorithm for global path generation with a modified Dynamic Window Approach (DWA) for real-time local obstacle avoidance. The adaptive A* algorithm improves search efficiency through a dynamic heuristic function, path simplification using redundant jump point removal, and trajectory smoothing via (Eta-Spline and Cubic B-Spline) interpolation. For local planning, a fuzzy logic system is employed to dynamically optimize the DWA evaluation function, with a modified formulation proposed for dynamic environments. This enhancement reduces planning time, improves trajectory smoothness, and effectively mitigates the local minima problem. In terms of control, the thesis proposes a robust trajectory tracking strategy based on the integration of Dynamic Feedback Linearization (DFL) and Fuzzy Logic Control (FLC). By exploiting the system's differential flatness property, the control law is derived in the flat output space to ensure accurate path tracking and effective obstacle avoidance behaviors. The synergy between model-based DFL and data-driven fuzzy control enhances tracking accuracy, robustness, and disturbance rejection. The proposed methods are validated through extensive simulations and real-world experiments, demonstrating significant improvements in navigation efficiency, control precision, and obstacle avoidance performance compared to conventional approaches.

Keywords: Autonomous Navigation, Path Planning, A-Star Algorithm, Dynamic Window Approach (DWA), Fuzzy Logic Control (FLC), Dynamic Feedback Linearization (DFL).

Résumé

Cette thèse vise à faire progresser la navigation autonome des robots mobiles à roues en abordant les défis fondamentaux de la planification du mouvement et du contrôle dans des environnements à la fois statiques et dynamiques. Un cadre de planification de trajectoire hybride est développé, combinant un algorithme A* adaptatif amélioré pour la génération globale de trajectoire avec une version modifiée de la méthode Dynamic Window Approach (DWA) pour l'évitement d'obstacles local en temps réel. L'algorithme A* adaptatif améliore l'efficacité de la recherche grâce à une fonction heuristique dynamique, à la simplification du trajet via la suppression redondante des points de saut, et au lissage de la trajectoire par interpolation (Eta-Spline et Cubic B-Spline). Pour la planification locale, un système de logique floue est employé pour optimiser dynamiquement la fonction d'évaluation de la DWA, avec une formulation modifiée proposée pour les environnements dynamiques. Cette amélioration réduit le temps de planification, améliore la fluidité de la trajectoire et atténue efficacement le problème des minima locaux. En contrôle, la thèse propose une stratégie robuste de suivi de trajectoire basée sur l'intégration de la linéarisation par rétroaction dynamique (Dynamic Feedback Linearization - DFL) et du contrôle logique flou (Fuzzy Logic Control - FLC). En exploitant la propriété de platitude différentielle du système, la loi de contrôle est dérivée dans l'espace des sorties plates afin d'assurer un suivi précis de la trajectoire et un comportement efficace d'évitement d'obstacles. La synergie entre la méthode DFL basée sur le modèle et le contrôle flou fondé sur les données améliore la précision du suivi, la robustesse et la réjection des perturbations. Les méthodes proposées sont validées par de nombreuses simulations et expérimentations réelles, démontrant des améliorations significatives en termes d'efficacité de navigation, de précision de contrôle et de performance d'évitement d'obstacles comparées aux approches classiques.

Mots-clés : Navigation Autonome, Planification de Trajectoire, Algorithme A-Star, Approche par Fenêtre Dynamique (DWA), Contrôle Logique Flou (FLC), Linéarisation par Rétroaction Dynamique (DFL).