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

# ADVANCED CONTROL AND LOCAL PLANNER FOR MOBILE ROBOTS

**Ahmed Adnan Hamzah AL-NASERI**

MASTER OF SCIENCE THESIS
Computer Engineering Department
Computer Engineering Program

Supervisor
Asst. Prof. Dr. Erkan USLU

January, 2024

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

# ADVANCED CONTROL AND LOCAL PLANNER FOR MOBILE ROBOTS

A thesis submitted by Ahmed Adnan Hamzah AL-NASERI in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE** is approved by the committee on 24.01.2024 in Computer Engineering Department , Computer Engineering Program .

Asst. Prof. Dr. Erkan USLU
Yildiz Technical University
Supervisor

**Approved By the Examining Committee**

Asst. Prof. Dr. Erkan USLU, Supervisor
Yildiz Technical University _____

Assoc. Prof. Dr. Hamza Osman İLHAN, Member
Yildiz Technical University _____

Asst. Prof. Dr. Nihal ALTUNTAŞ, Member
Istanbul Gelisim University _____

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

Ahmed Adnan Hamzah
AL-NASERI

Signature

*Dedicated to my family*
*and my best friend*

# ACKNOWLEDGEMENTS

All gratitude to Allah and His favour for completing this task. I thank God for all of the opportunities, hardships, and strength you have given upon me in order for me to accomplish my thesis.

I sincerely meant "thank you" to everyone who had a beneficial contribution in getting my thesis accomplished, more especially, I'd like to express my heartfelt gratitude to Asst. Prof. Dr. Erkan Uslu, was there for me every step of the way as I worked on my thesis and provided invaluable scientific and technical advice.

Lastly, I'd like to express sincere thanks to my family, who were there for me every step of the way, offering words of encouragement and giving the resources I needed to finish my master's thesis and all of my other academic endeavours. It is really appreciated.

Ahmed Adnan Hamzah AL-NASERI

# TABLE OF CONTENTS

# LIST OF SYMBOLS

| | |
|---|---|
| $\omega$,$w$ | Angular Velocity |
| $\varphi$ | Angle Between wheel and X-Axis |
| $\alpha$ | Angle Between The Vehicle's Orientation and The Line Connecting The Vehicle to The Lookahead Point |
| $\theta$ | Angle Difference Between Robot Direction and X-Axis |
| $\phi$ | Angle Error |
| $c_i$ | Cost Compute |
| $\gamma$ | Curvature of the Vehicle |
| $t_i$ | Feasible Trajectories in the Time Interval |
| $v$ | Linear Velocity |
| $K$ | Multiplication Factor |
| $\delta$ | Steering Angle |

# LIST OF ABBREVIATIONS

| | |
|---|---|
| ADWA | Combination of A* with Dynamic Window Approach |
| AGVs | Automated Guided Vehicles |
| AMCL | Adaptive Monte Carlo Localization |
| AOE degrees | Average Orientation Error Degree |
| APE M | Average Position Error in Meters |
| APP | Combination of A* pure pursuit without pid controller |
| APPD | Combination of A* with pure pursuit and pid controller |
| ATAG s | Average Time to Reach Around the Goal in Seconds |
| ATEB | Combination of A* with Time Elastic Band |
| DPPD | Combination of Dijkstra with pure pursuit and pid controller |
| DWA | Dynamic Window Approach |
| GPS | Global Positioning System |
| OSG | Overshooting Around the Goal |
| PID | Proportional Integral Derivative |
| ROS | Robot Operating System |
| SLAM | Simultaneous Localization and Mapping Protocol |
| TEB | Timed Elastic Band |
| WMRs | Wheeled Mobile Robots |

# LIST OF FIGURES

# LIST OF TABLES

# ABSTRACT

## Advanced Control and Local Planner for Mobile Robots

Ahmed Adnan Hamzah AL-NASERI

Supervisor: Asst. Prof. Dr. Erkan USLU

Wheeled Mobile Robots and Autonomous Ground Vehicles are now essential assets in several industries such as manufacturing, logistics, healthcare, and exploration. The development of WMRs can be tracked from its early concept to the advanced robotic systems that currently exist. This text delves into the comprehensive historical context of WMRs, examining significant achievements, important technologies, and major events that have shaped their development and evolution. The objective of this study is to enhance the efficiency of the navigation system by including a low-resolution mapping technique, in order to address the particular issue of overshooting that arises when combining A* with pure pursuit. This entails the random distribution of dots on the generated map, ensuring that each point is precisely separated by a distance of 0.2 metres. The grid will be implemented in the vacant spaces, establishing a restricted navigation area within manufacturing facilities. Furthermore, this strategy seeks to improve the precision of steering towards the waypoints in our low-resolution grid system. Moreover, there is an expectation to incorporate a PID control technique in conjunction with pure pursuit. This would include employing a PID Controller to govern both the linear and rotational velocity, thereby aiding the trajectory controller. The purpose of this strategic merger is to tackle the present problem of surpassing the set goal and reducing the frequency of encountering barriers while navigating. It accomplishes this by assessing several setups for PID parameters and predicting forthcoming conditions to ascertain the optimal parameter configuration. The main goal of this study is to enhance the manoeuvrability of AGVs in intricate and confined environments.The efficacy of our developed framework was evaluated using both simulation and a physical platform, whereby we compared the performance of local planners Pure Pursuit, DWA, and TEB models. We combined these local planners

with an A* and Dijkstra algorithm operating on a low-resolution grid-based system. The outcomes of three metrics The simulation tests emphasised the importance of adjusting look ahead, linear, and angular velocities to minimize the time and distance needed to achieve the goal, as well as to reduce orientation errors towards the target. Conversely, the actual platform studies yielded the most effective steering parameters for navigating complex landscapes.

**Keywords:** Mapping, Localization, Global planner, Local planner, Pure Pursuit with proportional-integral-derivative control.

# ÖZET

## Mobil Robotlar için İleri Kontrol ve Yerel Planlayıcı

Ahmed Adnan Hamzah AL-NASERI

Danışman: Asst. Prof. Dr. Erkan USLU

Tekerlekli Mobil Robotlar ve Otonom Kara Araçları, imalat, lojistik, sağlık hizmetleri ve keşif gibi çeşitli endüstrilerde artık vazgeçilmez varlıklardır. WMR'lerin gelişimi, erken konseptinden mevcut olarak var olan gelişmiş robot sistemlerine kadar izlenebilir. Bu metin, WMR'lerin kapsamlı tarihsel bağlamına dalmakta olup, gelişimlerini ve evrimlerini şekillendiren önemli başarıları, önemli teknolojileri ve büyük olayları incelemektedir. Bu çalışmanın amacı, A* ile saf takip kombinasyonunda ortaya çıkan aşırı atlamaların özel bir sorununu ele almak için düşük çözünürlüklü bir haritalama tekniğini navigasyon sisteminin verimliliğini artırmak için içermektir. Bu, oluşturulan haritadaki noktaların rastgele dağıtımını içerir ve her noktanın kesin olarak 0.2 metre uzaklıkta olduğundan emin olur. Izgara, boş alanlarda uygulanacak ve imalat tesisleri içinde sınırlı bir navigasyon alanı oluşturacaktır. Ayrıca, bu strateji, düşük çözünürlüklü ızgara sistemimizdeki yol noktalarına yönelik yönlendirme hassasiyetini artırmayı amaçlamaktadır. Ayrıca, saf takiple birlikte bir PID kontrol tekniğini dahil etme beklentisi bulunmaktadır. Bu, PID Denetleyicisi'nin hem lineer hem de dönüşsel hızı yönetmek için kullanılmasını içerir ve bu da yol kontrolcüsüne yardımcı olur. Bu stratejik birleşmenin amacı, belirlenen hedefi aşma sorununu ele almak ve navigasyon sırasında engellerle karşılaşma sıklığını azaltmaktır. Bu, PID parametreleri için birkaç kurulumu değerlendirerek ve gelecekteki koşulları tahmin ederek en uygun parametre yapılandırmasını belirlemek suretiyle başarılır. Bu çalışmanın ana hedefi, AGV'lerin karmaşık ve sınırlı ortamlarda manevra kabiliyetini artırmaktır. Geliştirdiğimiz çerçevenin etkinliği, hem simülasyon hem de fiziksel bir platform kullanılarak değerlendirildi ve yerel planlayıcılar Pure Pursuit, DWA ve TEB modellerinin performansı karşılaştırıldı. Bu yerel planlayıcıları düşük çözünürlüklü bir ızgara tabanlı sistemde çalışan bir A* ve Dijkstra algoritması ile birleştirdik. Üç metriğin sonuçları, simülasyon

testlerinin hedefe ulaşmak için gereken zaman ve mesafeyi minimize etmek, ayrıca hedefe doğru yönelim hatalarını azaltmak için bakış açısını, lineer ve açısal hızları ayarlamak için önemini vurgulamıştır. Bununla birlikte, gerçek platform çalışmaları, karmaşık manzaraları gezinmek için en etkili yönlendirme parametrelerini ortaya koymuştur.

**Anahtar Kelimeler:** Haritalama, Konum belirleme, Global planlayıcı, Yerel planlayıcı, Oransal-integral-türevsel denetleyici temelli saf takip.

# 1
# INTRODUCTION

## 1.1 Motivation

In contemporary smart warehouses and manufacturing plants, the pivotal role played by Automated Guided Vehicles (AGVs) in optimizing logistic operations cannot be overstated. However, amidst their efficiency, a persisting issue in these environments pertains to the navigation process of AGVs, necessitating targeted solutions and enhancements. The critical challenge that demands immediate attention is the optimization of AGVs navigation to ensure efficient and secure movement within these dynamic spaces [1, 2].

One of the primary hurdles encountered in AGV's navigation within smart warehouses and manufacturing facilities is the lack of precision in traversing predefined routes, often leading to undesired outcomes such as overshooting designated areas and unintended collisions [3]. This problem, if left unaddressed, not only compromises the safety of the operational environment but also impedes smooth workflow and logistical efficiency, consequently impacting productivity[4–7].

To tackle these challenges head-on, our research endeavors to introduce specific solutions aimed at revolutionizing the AGVs navigation process. Our proposed approach revolves around the implementation of a marker-based system to create restricted navigation areas for AGVs. By strategically deploying markers within the operational space, we aim to define precise pathways, thereby confining AGV movement to designated routes while minimizing the occurrences of deviation and overshooting.

Moreover, in our pursuit of refining AGV navigation, we envisage the integration of a Proportional-Integral-Derivative (PID) controller mechanism. The incorporation of this controller, encompassing both Linear and Angular Velocity, aims to revolutionize AGV movement by offering precise control over speed and direction.

Through the PID controller's dynamic adjustments, our objective is to mitigate the instances of overshooting and crashing, enhancing the AGVs' navigation accuracy while ensuring a safer operational environment[8].

This research venture represents a significant step towards addressing the pressing issues encountered in AGV navigation within smart warehouses and manufacturing facilities. By harnessing innovative technologies and adopting a strategic approach, our goal is to not only reduce operational risks but also augment the overall efficiency, safety, and productivity of AGV-assisted logistical operations in modern industrial settings[9].

## 1.2 Literature review

Wheeled mobile robots from a historical perspective, wheeled mobile robots, commonly known as ground robots, represent a pivotal milestone in the realm of autonomous and semi-autonomous machinery. The history of robotic development as shown in Figure 1.1.

Fitted with wheels, these robots exhibit a remarkable ability to traverse terrestrial landscapes. The progression of WMRs mirrors a ceaseless drive towards automation, heightened efficiency, and groundbreaking innovation across diverse sectors. Delving into their historical trajectory is paramount not only for comprehending their existing prowess but also for envisioning the remarkable strides they are poised to make in the future. Early Origins (1950s-1960s)The inception of WMRs dates back to the mid-20th century, when visionaries in the field started conceiving the idea of machines endowed with autonomous mobility.

An exemplary milestone during this period was the creation of George Devol's "Unimate" in 1954, which heralded the dawn of industrial robotics[10]. The history of automated guided vehicle systems started in the 1950s as shown in Figure 1.2

The American automotive industry was the initiator of the invention of automated guided vehicles and was the first where these systems for organizing modern intralogistics were used. Soon afterward almost all industries started to use AGVs to optimize material flows. Automated guided vehicle systems have become a key component of today's intralogistics. The history of AGVs can be divided into four eras according to the state of the technology available. The first era of AGVs began in 1955 and lasted nearly 20 years. At the beginning, there had been the idea of replacing the driver of a vehicle with automated logistics systems and porting goods using automation. The second era started around 1970 and was

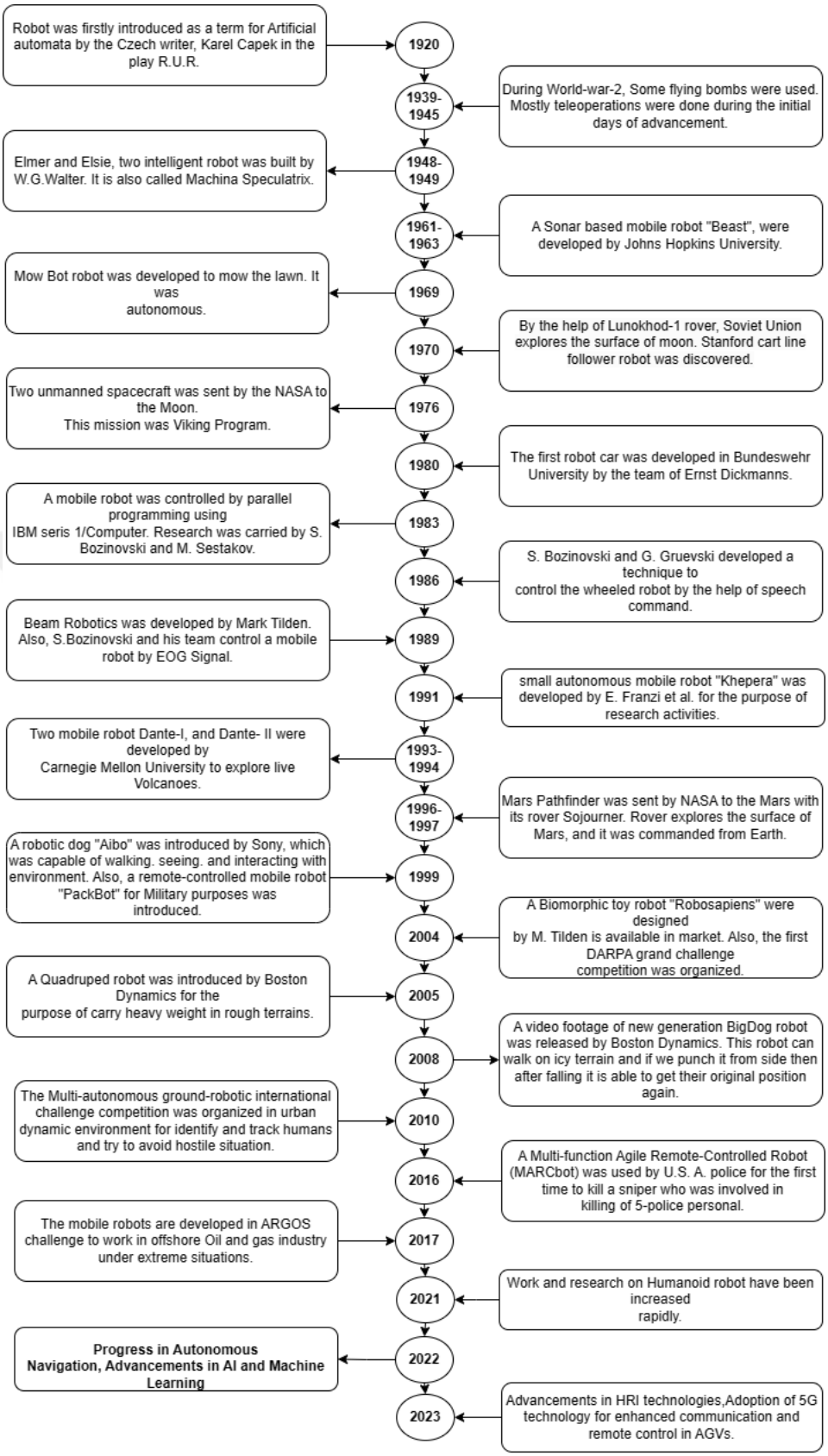



**Figure 1.1** Robotic developments in the last 100 years

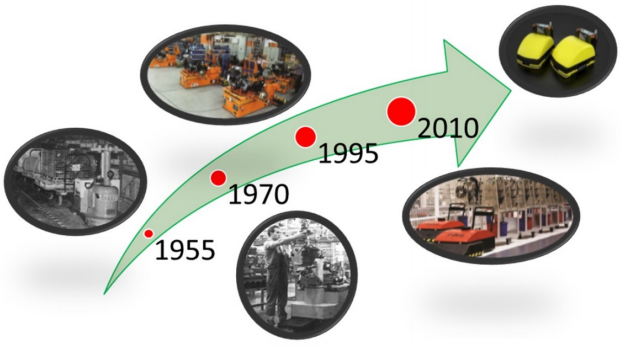



**Figure 1.2** WMRs

characterized by simple experiments with the implementation of the first on-board computers. The third era began in the mid-1990s. The devices were equipped with electronic guidance systems and proximity sensors. The first attempts using Wi-Fi communication to transfer data were realized. The fourth era is a fluent continuation of the development of the third era. The effort is to go the way of autonomy of these systems up to the level which allows the available technology when the control systems are able to decide autonomously on the motion in space, they are able to detect an obstacle and select an alternate path [11].While these initial robots were predominantly focused on manipulation tasks, the concept of autonomous mobility was already beginning to take shape as shown in Figure 1.3 .

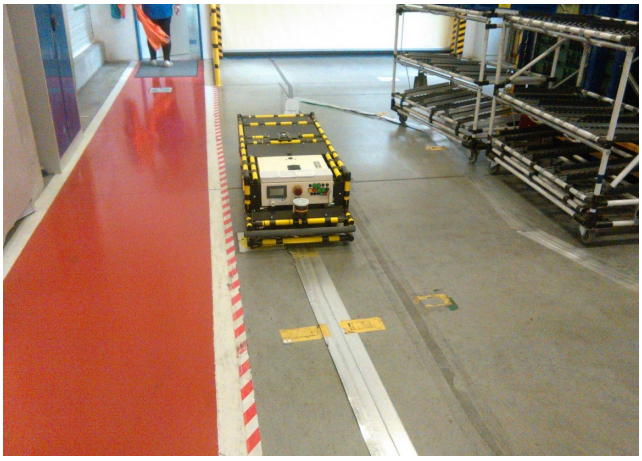


**Figure 1.3** AGVs

The Shakey Robot (1960s-1970s)During the 1960s and 1970s, the Stanford Research Institute (SRI) pioneered the development of the "Shakey" robot, often hailed as the world's inaugural mobile robot see Figure1.4 .

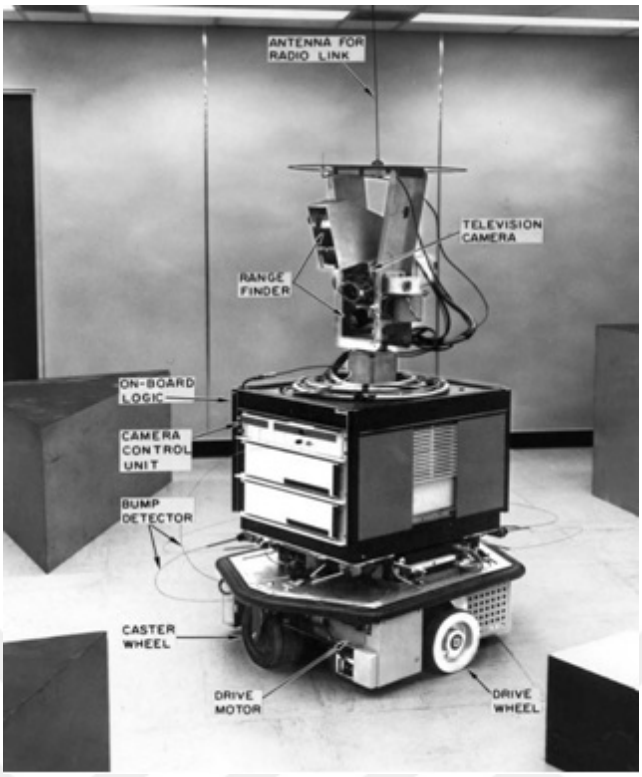



**Figure 1.4** "Shakey" robot

Shakey utilized a blend of wheels and tracks for locomotion, complemented by a suite of sensors for navigation. While its functionalities may seem rudimentary in comparison to contemporary standards, shakey played a pivotal role in establishing the groundwork for autonomous mobile robotics [12]. Advancements in navigation (1980s-1990s)The 1980s stand out as a transformative period in robotics, characterized by significant advancements in sensor technology and navigation algorithms. During this epoch, researchers embarked on the integration of cutting-edge laser rangefinders and ultrasonic sensors, endowing robots with a newfound level of perceptual acumen. Concurrently,creative solutions such as the "Bug Algorithm" and the "Potential Field Method" emerged as foundational navigation algorithms, fundamentally reshaping the landscape of obstacle avoidance and path planning strategies[13]. The Pioneer Era (1990s-2000s)The 1990s heralded the pioneering age of wheeled mobile robots (WMRs), witnessing the ascent of commercial marvels like the renowned "Pioneer" series developed by active media robotics. These robots swiftly found their niche across diverse domains including research, education, and industry, sparking an even greater fervor for the advancement of autonomous ground vehicles [14] . Robocup and the rise of autonomous soccer (1997-Present) the inception of Robocup, a global robotics competition, stands as a watershed moment in the progression of wheeled mobile robot (WMR) research. The debut of soccer-playing

robots in 1997 marked the dawn of an epoch characterized by autonomous navigation, collaborative interactions, and instantaneous decision-making. To this day, Robocup remains an instrumental catalyst for soccer playing or pioneering innovations within the realm of robotics [15].

Furthermore military and defense applications (2000s-present)in the 2000s, wheeled robotic platforms experienced a surge in adoption within military and defense contexts. Pioneering machines like the PackBot and Talon assumed pivotal roles in critical operations, including bomb disposal and reconnaissance. Their deployment significantly mitigated risks to human personnel, underlining the transformative impact of wheeled robots in safeguarding lives and enhancing mission efficiency [16].Commercialization and consumer robotics (2010s-Present) in the 2010s, a notable trend emerged with the widespread commercialization of wheeled mobile robots (WMRs) targeting consumer markets. Industry leaders such as irobot achieved remarkable global success, particularly with their robot vacuum series. Concurrently, an array of startups ventured into diverse sectors including agriculture, healthcare, and logistics, harnessing the potential of WMRs. This era witnessed a significant wide spread of WMR technology, making it accessible to a broader and more diverse audience[17] .Autonomous vehicles and the road ahead The contemporary horizon of Wheeled Mobile Robots (WMRs) finds its most transformative application in the realm of autonomous vehicles (AVs). Pioneering companies such as waymo and tesla are at the vanguard of this movement, pushing the boundaries of self-driving cars rooted in WMR principles. Their endeavors hold the promise of fundamentally revolutionizing the landscape of transportation as we know it. The history of wheeled mobile robots is a testament to human ingenuity and the relentless pursuit of automation. From humble beginnings to autonomous vehicles, WMRs have come a long way[18]. Understanding this history is essential for envisioning their promising future in various industries.A journey through time and the history of autonomous ground vehicles (AGVs)The historical journey of autonomous ground vehicles (AGVs) is a testament to human ingenuity, perseverance, and technological advancement. This comprehensive literature review delves into the annals of AGVs, tracing pivotal moments, influential technologies, and key milestones that have steered their development from early inception to the sophisticated autonomous systems of today. By immersing ourselves in this rich chronicle, we glean profound insights into the progression of AGVs and their monumental capacity to redefine transportation and beyond[19] .Early visionaries (1950s-1960s)the inception of autonomous ground vehicles (AGVs) can be traced back to the mid-20th century, an era marked by visionaries like arthur samuel and john mccarthy. Samuel's pioneering checkers-playing

program and mccarthy's groundbreaking contributions to artificial intelligence laid the bedrock for subsequent research in autonomous vehicle technology [20]. Their early insights and innovations continue to resonate profoundly in the domain of self-driven vehicles today.In stanford cart (1960s-1970s) during the 1960s, stanford university's "Stanford Cart" emerged as a trailblazer in the realm of autonomous navigation. Financed by the US department of defense, this wheeled robotic platform represented a groundbreaking endeavor in the ongoing evolution of autonomous ground vehicles (AGVs). Its contributions laid a crucial foundation for subsequent advancements in this field The rise of DARPA challenges (2000s-Present)the defense advanced research projects agency (DARPA) assumed a pivotal role in propelling AGV technology forward, notably through its highly influential autonomous vehicle challenges. Landmark events such as the DARPA grand challenge and urban challenge served as catalysts for innovation, particularly in the domains of perception, planning, and control. Autonomous robots are sent to investigate dangerous areas of the environment in order to gather vital data, create a representation of the surroundings, and identify potential victims' whereabouts[21] . These initiatives have significantly accelerated the development of cutting-edge autonomous ground vehicles .Commercial AGVs (1980s-Present) The 1980s witnessed a pivotal moment in the industrial landscape with the integration of commercial autonomous ground vehicles (AGVs) into sectors like manufacturing and logistics.These pioneers were the first to demonstrate the vast capabilities of AGVs in automating crucial operations involving material handling and transportation in limited operating areas. This period marked a transformative juncture in the industrial landscape, setting the stage for further advancements in AGV technology.AGVs in agriculture (2010s-Present)The agricultural sector experienced a notable surge in the adoption of Autonomous Ground Vehicles (AGVs). This paradigm shift was particularly evident in the deployment of autonomous tractors and robotic harvesters, heralding a new era in farming practices. These innovations not only substantially enhanced operational efficiency but also contributed significantly to cost reduction by minimizing labor expenses . Their groundbreaking efforts in self-driving vehicle technologies catapulted autonomous ground vehicles (AGVs) into the mainstream consciousness, offering the enticing prospect of a future characterized by safer and highly efficient road transportation [22] .

The AGVs continue to evolve, fueled by advancements in artificial intelligence, sensor technologies, and connectivity. The future holds promises of fully autonomous vehicles, improved urban mobility, and novel applications in domains such as healthcare and last-mile delivery.The evolution of autonomous ground

vehicles stands as a testament to human ingenuity and an unwavering dedication to automation. Progressing from modest origins to the brink of a transformative era, AGVs have demonstrated extraordinary advancements. Grasping this historical journey is of paramount importance in appreciating the immense potential these vehicles hold in revolutionizing transportation and elevating our everyday experiences.

Upon reviewing the majority of the researchers' work, it is evident that they primarily address the issue of insufficient accuracy in enhancing the overall navigation precision through the implementation of a local planner, such as pure pursuit with a PID controller. However, there is a notable absence of discussion regarding several parameter options that must be configured, including but not limited to look ahead, PID linear velocity, PID angular velocity, goal tolerance, and obstacles tolerance [23–27].

## 1.3 The objective of the Thesis

The observed phenomena in a pure pursuit variants have been proposed with a schema for variable linear velocities, either assuming a constant velocity or failing to address the point at all [28] .

The pure-pursuit algorithm can suffer from overshooting, where the vehicle passes the desired trajectory point before being able to correct its path. This can occur when the look-ahead distance is too large, causing the vehicle to have difficulty in accurately tracking the reference trajectory.

Overshooting can also be a result of improper selection of the look-ahead distance. If the look-ahead distance is too small, the vehicle may not have enough time to react and adjust its path, leading to overshooting.

To address this issue, recent developments have focused on dynamically changing the look-ahead distance. By selecting multiple goals, modifying the look-ahead distance according to the curvatures, and adjusting lateral deviation, the algorithm can better handle overshooting and improve tracking performance[29].

We have introduced a proposed model designed to integrate the Pure Pursuit technique alongside a PID controller to navigate along a path delineated by a low-resolution grid map in a more accurate manner.

The objective of the mentioned combination is to refine and optimize the navigation system for wheeled mobile robots. Specifically, it aims to address the issue

of overshooting around the goal, which has been observed when combining A* algorithm with pure pursuit. By introducing a low-resolution grid with a 0.2-meter spacing, along with the incorporation of a PID controller, the goal is to achieve precise path following and minimize overshooting. This integrated approach seeks to enhance the robot's ability to navigate efficiently and accurately in diverse environments, making it a significant advancement in mobile robot navigation.

## 1.4 Hypothesis

The integration of A* algorithm with Pure Pursuit, complemented by a low-resolution grid , demonstrates a sophisticated approach to path planning. This combination leverages the strengths of each component, ensuring both accuracy and efficiency in navigation. Additionally, the incorporation of a PID controller further refines the robot's ability to precisely adhere to the designated path. This synergistic approach not only enhances goal-reaching capabilities but also minimizes overshooting, showcasing a comprehensive solution for advanced mobile robot navigation precisely.

## 1.5 Contribution of the thesis

This research contributes to establish a versatile model capable of adept navigation within diverse environments. This is achieved through the automated generation of a low-resolution grid, characterized by markers, following the mapping phase. Primarily, this method seeks to delineate restricted navigation zones within smart warehouses or manufacturing setups. Second of that , the study aims to underscore the impact of employing a PID controller, addressing both linear and angular velocities. The purpose of this controller is to meticulously trace the path delineated by the global planner through the prescribed markers with precision,in general this method augments the over all navigation precision.

# 2
# METHODICAL ANALYSIS

This chapter will comprehensively cover and demonstrate the crucial methods and tools utilized in our thesis model, encompassing all essential components. A meticulously designed grid with distinct marks has been incorporated into the map to facilitate navigation through tiny passages in the warehouse. The path planner deliberately chooses these landmarks to serve as navigational waypoints, directing the robot's orientation. With an inter-marker distance of 0.2 meters, this strategy, referred to as "point-based control," stands as a widely employed technique renowned for its accuracy in robotics navigation. Both the PID-based Pure Pursuit system and the locally designed planners employed in this investigation harness the potential of these map-embedded markers. Leveraging the PID-based Pure Pursuit controller significantly elevates navigation, affirming the premise that mitigating overshooting and trajectory oscillations is feasible through the adept utilization of PID and Pure Pursuit. These solutions aspire to augment the stability, precision, and efficacy of trajectory tracking mechanisms.

## 2.1 Navigation History

To construct a navigation system for a 2D wheeled mobile robot, it is imperative to grasp the fundamental tenets, particularly those concerning Kinematics. Our emphasis revolves around comprehending the intricacies of differential drive systems. Within this domain, conducting comprehensive theoretical pre-calculations assumes paramount importance as they serve as the foundational bedrock preceding the actual construction and deployment of our real platform.

**Introduction to Kinematics and Differential Drive**

Wheeled mobile robots are a mainstay of contemporary robotics, used in everything from space exploration to industrial automation. Designing precise navigation strategies requires an understanding of its kinematics, which is the study of motion without taking forces into account. This in-depth investigation digs into the

kinematics of wheeled mobile robots, clarifying the mathematical underpinnings, different motion types, and implications for control and path planning.

The Kinematics' mathematical foundation is Kinematics quantifies the connections between the locations, velocities, and accelerations of robotic parts. Kinematics is the study of how the body and wheels of mobile robots on wheels move. A robot's kinematic model entails converting wheel velocities to linear and angular velocities for the robot.

In forward kinematics connecting pose to control inputs based on the joint angles, forward kinematics explains the location and orientation of the robot. This entails applying trigonometric relationships to relate wheel velocities to the robot's linear and angular velocities for mobile robots with wheels. Depending on the robot's configuration, such as when using either a differential or omnidirectional drive, the forward kinematic equations undergo modification[30].

Solving for control inputs in inverse kinematics by finding the required wheel velocities for a specified position is the focus of inverse kinematics. In terms of trajectory planning and path tracking, this is especially important. Depending on the geometry of the robot, the inverse kinematic equations might be either linear or nonlinear[31].

In differential drive kinematics of robotics, two separately controlled wheels control the robot's motion. The wheelbase of the robot is included in the kinematic equations that connect wheel velocities to linear and angular velocities. The mathematical formulas describe how the robot's body responds to wheel motions[23].

The omnidirectional drive kinematics, is a holonomic skills , omnidirectional robots allow them to move instantly in any direction. They include lateral velocities in their kinematic equations in addition to linear and angular velocities. These equations make use of wheel speeds, orientations, and geometrical variables[32].

The difference between holonomic non-holonomic motion in some restrections , non-holonomic robots have restrictions because of their wheel arrangement, but holonomic robots can move in any direction without restrictions. Plans for motion planning and control may be affected by this duality,due to its adaptability, holonomic robots frequently involve kinematic equations that are more complicated.

**Non-holonomic Robots**

Non-holonomic robots cannot perform instantaneous translations in any direction without adjusting their orientation first. They typically have limitations on their movement, making them less maneuverable in some directions. These robots have fewer control inputs compared to their degrees of freedom, restricting their ability to move in all directions freely in Figure 2.1.Most wheeled vehicles, such as cars, trucks, and traditional differential-drive robots, fall under this category[33].

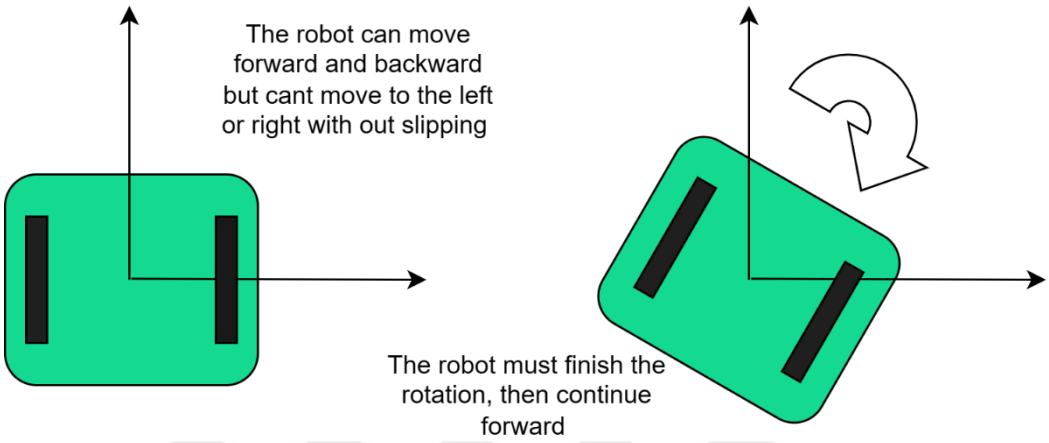



**Figure 2.1** Non-holonomic robots orientation

The robot can move in some directions (forwards and backwards), but not others (side to side ) .

**Holonomic and Non-Holonomic Constraints**

With no restrictions, the kinematic equations of holonomic robots allow for instantaneous movement in any direction. As a result of their wheel layout, non-holonomic robots like differential drive robots are constrained in their movements. due to their flexibility, holonomic robots have more intricate equations[34].

Kinematics serves as the basis for the algorithms used in control and trajectory planning. Robots can follow desired pathways and strike specific poses by adjusting wheel velocities. These methods compute the required wheel commands for precise motions using the kinematic equations.

The kinematic applications and implications study of wheeled mobile robots has implications for a wide range of applications, including autonomous vehicles, planetary exploration, and warehouse automation. Engineers may create control systems that guarantee precise and secure robot motions by having a solid understanding of kinematics.

The challenges in the use of robotic kinematics is changing as a result of developments in sensor technology and machine learning. Simultaneously, tackling the difficulties posed by atypical wheel behaviours, and ambiguities in sensor readings, kinematics provides a systematic framework for comprehending and managing the motion of wheeled mobile robots. The language used to communicate the desired behaviors by robot designers is mathematics. A thorough understanding of kinematics will always be essential for creating robots that can navigate and communicate effectively in a constantly changing environment, as robotics continues to advance[35].

**Forward Kinematics for Differential Drive Robots**

Robot pose represented by $\dot{x}$, $\dot{y}$, $\theta$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \theta \end{bmatrix} = R_z(\phi) \begin{bmatrix} V \\ 0 \\ \omega \end{bmatrix} = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V \\ 0 \\ \omega \end{bmatrix}$$

$$\begin{bmatrix} V \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{R_r}{2} & \frac{R_L}{2} \\ \frac{R_r}{2b} & -\frac{R_L}{2b} \end{bmatrix} \begin{bmatrix} \phi_r \\ \phi_L \end{bmatrix}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \theta \end{bmatrix} = \begin{bmatrix} \frac{R_r}{2}\cos(\phi) & \frac{R_L}{2} - \sin(\phi) \\ \frac{R_r}{2}\sin(\phi) & -\frac{R_L}{2b}\cos(\phi) \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \phi_r \\ \phi_L \end{bmatrix} \tag{2.1}$$

Robots with differential drives have two wheels that can be operated separately. The following equations can be used to link the robot's center's linear velocity V and angular velocity to the wheel velocities Vr (for the right wheel) and V (for the left wheel):

The equation

$$V = R2 \cdot (V_r + V_l) \tag{2.2}$$

$$V = 2R \cdot (V_r + V_l) \tag{2.3}$$

$$\omega = RL \cdot (V_r - V_l) \tag{2.4}$$

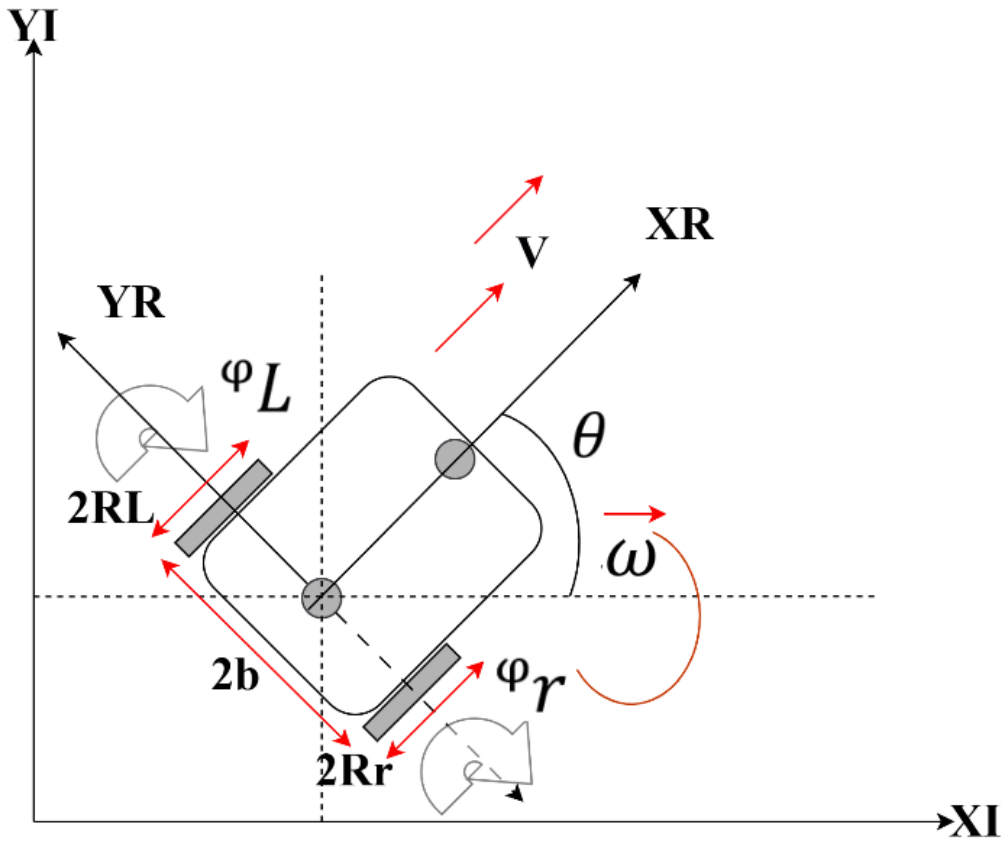



**Figure 2.2** Non-Holonomic robots

$$\omega = LR \cdot (V_r - V_l) \tag{2.5}$$

Where R is the radius of the wheels, and L is the distance between the wheels.

**Inverse Kinematics for Differential Drive Robots**

Using the following formulas to calculate the desired V and the wheel velocities Vr and Vl:

$$V_r = V_R + L2 \cdot \omega \tag{2.6}$$

$$V_r = RV + 2L \cdot \omega \tag{2.7}$$

$$V_l = V_R - L2 \cdot \omega \tag{2.8}$$

$$V_l = RV - 2L \cdot \omega \tag{2.9}$$

## 2.2 ROS

The Robot Operating System (ROS) has become an essential part of the process for creating and deploying robots for a variety of purposes. Because of the unique benefits it offers, it has gained significant acceptance in the robotics field and has become an essential tool for researchers, engineers, and developers working on robot systems. This thesis examines the need for ROS in robot methodology and explains its major benefits in relation to contemporary robotics.

ROS Architecture is a modular and networked architecture that enables programmers to build and implement complicated robotic systems by dissembling them into smaller, more manageable parts known as nodes [36]. Each node carries out a particular function while interacting with other nodes via a publish-subscribe method using topics. This flexibility makes it easier to create, test, and maintain robot systems by allowing for the seamless integration of various algorithms, sensors, actuators, and parts as shown in the Figure 2.3.

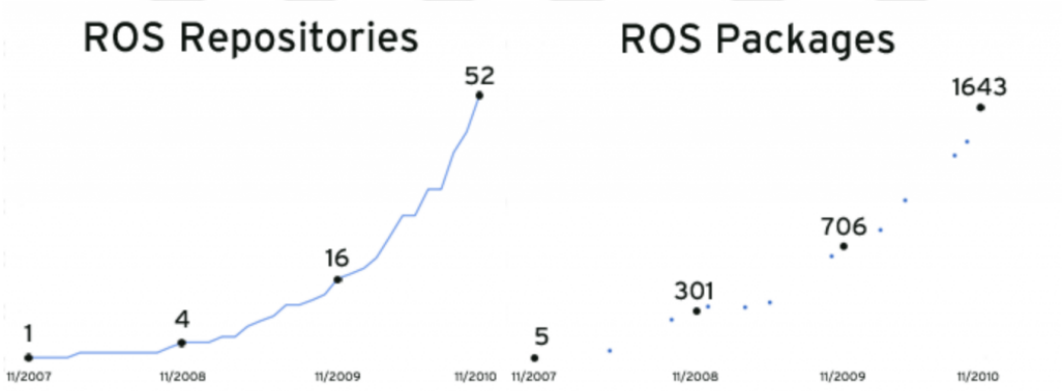



**Figure 2.3** ROS history

one of the major issues in robotics is dealing with various hardware configurations and platforms, according to hardware abstraction. By offering a hardware abstraction layer, ROS overcomes this problem by enabling programmers to build code that is not dependent on any one piece of hardware. This makes it simpler to switch hardware or implement the same algorithm on several robot platforms because the same ROS code can be used across various robots with different sensors and actuators.

ROS offers a reliable communication and data sharing communication architecture

that makes it possible for various robot system components to share and coordinate data effectively. Developers can use distributed and cooperative algorithms like mapping and localization without worrying about the nitty-gritty of communication by using this communication paradigm. Data sharing between nodes facilitates collaboration among several developers working on various facets of the robot system and streamlines the development process.

Rich collection of libraries and tools ROS includes many libraries and tools that support a variety of capabilities, including perception, planning, control, and visualization. Because they may use pre-existing code rather than beginning from scratch, these pre-existing packages and tools help developers save time and effort. Open-source tools and libraries are also a supporter of knowledge exchange and community-driven development.

ROS has powerful simulation capabilities, allowing developers to test robot algorithms and behaviours in a virtual environment before deploying them on physical robots. This capability is crucial for proving algorithm correctness and robustness, lowering the danger of harming expensive hardware during development, and speeding up the iterative development process.

ROS has a huge and active community of researchers, engineers, and hobbyists who actively contribute to its development and maintenance. ROS's community-driven approach supports information exchange, discussion of best practices, and collaboration on research and development projects. This collaborative environment hastens the speed of innovation and hastens the evolution of robot technology.

The versatility and extensibility of ROS make it appropriate for a wide range of robotics, from small research platforms to industrial robots and large-scale autonomous systems. Because it is open-source, developers can alter and customize the code to meet their individual needs, making it applicable to a wide range of applications and research disciplines.

Scalability and application in the real world of ROS is meant to be scalable, allowing it to handle complicated robot systems with several nodes and vast volumes of data. Because of its networked architecture, it can handle real-time data processing and control, making it suited for real-world deployment in time-critical applications like autonomous vehicles and industrial automation.

A robot operating system (ROS) is required in the robot approach because to its distributed and modular design, hardware abstraction, effective communication, and large tool library. ROS promotes community-driven research, cooperation, and the

creation of sophisticated robotic systems while quickening the speed of innovation in the robotics industry. ROS has developed into a vital tool for academics and developers, allowing them to build sophisticated and effective robot systems for a variety of applications. This is due to its flexibility, extensibility, and scalability.

### 2.2.1 Nodes

Nodes are basic computational units that enable the distributed and concurrent execution of tasks inside a robotic system, particularly in the context of robotics and the robot operating system (ROS). These autonomous software components are essential for enabling smooth communication and teamwork between various robot sections and for the development of sophisticated and adaptable robotic applications. This thesis explores the idea of ROS nodes, their essential traits, and their importance in contemporary robotics.

### 2.2.2 Definition and Characteristics of Nodes

A node in the ROS operating system is defined as a self-contained software module that carries out a particular task or feature inside the robot system. Each node functions as a separate process that runs alongside other nodes, enabling the execution of activities in parallel. Because developers may independently establish and operate nodes in this distributed architecture, it is easier to build and maintain massive robotic systems.

Depending on the developer's inclination and the precise specifications of the work at hand, nodes are generally built in programming languages like C++, Python, or even MATLAB. The language-neutral nature of ROS fosters interoperability by allowing nodes built in various languages to interact with one another without any problems.

One of the key characteristics of ROS nodes is their capacity to interact with one another via the publish-subscribe messaging paradigm. In this communication paradigm, nodes communicate by sending and receiving messages, which are data packets that include information pertinent to the job at hand. Nodes have the ability to publish messages to particular topics, while other nodes can subscribe to receive the messages.

The publish-subscribe architecture enables nodes to interact in a decoupled way, which means that nodes do not need to know the identities of the nodes with whom they are speaking. To send or receive data, nodes just need to know the subject name

and the message type. This loose connectivity between nodes improves modularity and flexibility, making it easier to add and delete nodes without impacting other components.

### 2.2.3 Creating and Running ROS Nodes

Developers can use the ROS command-line interface (CLI) or launch files to construct and operate ROS nodes. The CLI enables developers to establish and maintain nodes on their own, providing them with fine-grained control over the system's behavior. Launch files, on the other hand, give a more straightforward approach to start many nodes and define their settings with a single command, making complicated robot systems with many nodes easier to operate.Launching a node entails registering it with the ROS master, which serves as the system's central coordinator. The ROS master keeps track of all accessible nodes and topics, allowing nodes to find one another and create communication channels. This registration method guarantees that nodes can discover one another and collaborate to achieve the general goals of the robot system.ROS nodes are designed to run independently, allowing them to accomplish their functions effectively while without interfering with the functioning of other nodes. Each node operates in its own process space, guaranteeing that a failure or breakdown in one node does not have an impact on the others. This separation improves the robot system's resilience and fault tolerance. Nodes exchange information by publishing data to subjects or subscribing to topics to receive data. Topics serve as communication channels, enabling nodes to exchange data in real-time. When a node has useful information to contribute, it publishes it to a specific topic, and any other nodes interested in that data can subscribe to the topic to receive it. The event-driven nature of the communication paradigm allows nodes to react to real-time changes and efficiently coordinate their actions. A perception node, for example, may post sensor data to a topic, and a control node may subscribe to that topic in order to receive the data and make choices based on it. ROS nodes and multi-robot systems ROS's node-based design is well-suited for multi-robot systems, in which numerous robots collaborate to achieve a shared goal. Each robot in such systems may be represented by its own collection of nodes, allowing for efficient communication and coordination amongst robots. ROS is a suitable platform for constructing sophisticated and collaborative multi-robot systems due to its ability to disperse processing over several nodes and robots.

### 2.2.4 ROS Nodelets

ROS provides the idea of nodelets to further optimize speed and decrease memory overhead. Nodelets are lightweight versions of nodes that may share memory and communication channels and execute within the same process. This resource sharing improves the overall efficiency of the robot system and is especially effective for computationally heavy jobs.

### 2.2.5 Node Communication

ROS supports a variety of node communication patterns, including the previously stated publish-subscribe architecture. ROS also includes alternative communication patterns such as request-response (service calls) and goal-result (actionlib). Service calls allows one node to request data or services from another, whereas actionlib permits asynchronous, goal-driven communication, which is especially suitable for complicated operations that need long-duration execution.

### 2.2.6 ROS Node Debugging and Visualisation

Developers may utilize numerous ROS tools to debug and visualize node behavior. Developers may visualize the node network and watch the messages flowing between nodes using tools like rqt_graph . These visualization tools assist in understanding the behavior of the system, discovering communication difficulties, and fixing mistakes.

### 2.2.7 Integration with Robot Perception and Control

ROS nodes are required for the integration of perception and control modules in robot systems. Perception nodes analyse sensor data, such as camera pictures or lidar scans, to extract relevant information about the robot's environment. Control nodes are responsible for navigating the robot and interacting with it as shown in Figure2.4

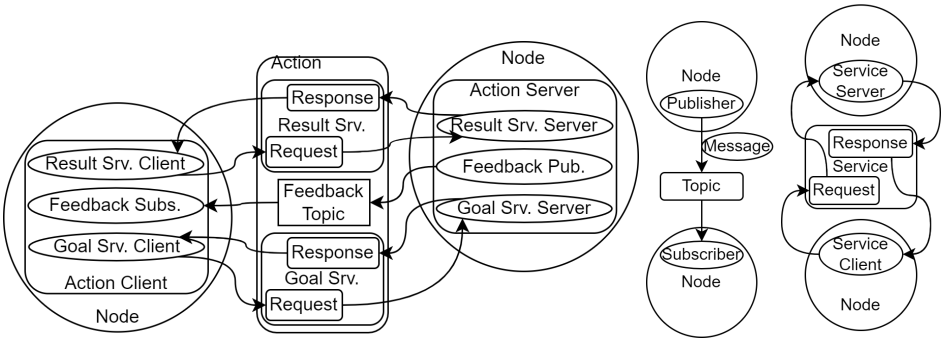



**Figure 2.4** ROS nodes and topics

There are 2 types of actions publishing and subscribing, therefore data can be shared node publish-subscribe or service server-service client. To facilitate message exchange, a topic name must be provided. Here are some instances of such topics given as below :

**/odom**

Odom topic is published by "wheel odometry" node. It uses left encoder ticks and right encoder ticks to calculate position of the robot (x, y , thetha). It is published as a nav_msgs/Odometry type message.

**/map**

This topic is used to publish the map for navigation. It is created by the map_server node by reading a PNG file, and is sent to movebase. Global costmap and markers are created using this topic. It is of the type nav_msgs/OccupancyGrid

**/Scan**

This topic is used to publish the laser data created by the "urg_node" (urg_node is the lidar sensor driver). It is sent to the navigation stack and amcl nodes to be used for localization and collision avoidance. It uses a sensor_msgs/LaserScan type message.

## 2.3 SLAM and G-Mapping

G-Mapping, an abbreviation for grid-based mapping, stands as a renowned methodology in the realm of robotics, addressing the intricate challenge of simultaneous localization and mapping (SLAM). Specifically tailored to craft precise representations of uncharted environments, all the while forecasting the precise location of the robot within said space, G-Mapping has garnered significant

acclaim. Its effectiveness in crafting intricate maps and empowering self-directed movement across a myriad of applications has propelled it to the forefront of robotic mapping techniques.

### 2.3.1 G-Mapping Algorithm Overview

G-Mapping works by using a grid-based representation of the environment in which the workspace is discretized into tiny cells. Each cell represents a section of the environment and is assigned a probability value expressing the possibility of occupancy. The method uses sensor data, mainly from laser range finders, to update the occupancy probability of these cells. G-Mapping builds a map that properly portrays its surroundings by repeatedly combining sensor readings and robot motion data.

#### 2.3.1.1 Map Update

The fundamental idea of the technique is to repeatedly update the map using sensor measurements and motion information. G-mapping determines the likelihood of a new sensor scan given the map whenever one is acquired. It then updates the occupancy probability of the appropriate grid cells using these likelihood values. The accuracy of the map is improved by modifying the robot's attitude based on the particle filter.

#### 2.3.1.2 Real-Time Operation

G-Mapping's efficiency in updating maps and estimating poses in real-time makes it suitable for applications that require continuous navigation and mapping, such as autonomous vehicles and drones. G-Mapping can deal with noisy sensor data, motion inaccuracies, and dynamic situations thanks to its probabilistic nature.

#### 2.3.1.3 Grid-Based Mapping

G-Mapping divides the surroundings into a grid of cells. These cells hold information regarding the likelihood of occupancy. To represent the uncertainty about the environment, all cells are initially assigned low probability. G-Mapping adjusts the probability depending on sensor measurements and the robot's attitude as the robot collects sensor data.

#### 2.3.1.4 Particle Filter Localization

G-Mapping estimates the robot's location using a particle filter-based technique. This strategy entails keeping a collection of particles, each of which represents a potential robot stance. Particles are propagated and resampled as the robot travels and collects sensor data, causing them to converge on the real robot position. G-Mapping can tolerate uncertainty and deliver reliable localization estimates because of this method.

#### 2.3.1.5 Exploration and Mapping

In exploratory missions, such as search and rescue operations or environmental monitoring, where robots must map and traverse unfamiliar surroundings, G-Mapping is essential.

#### 2.3.1.6 Loop Closure

When the robot revisits previously investigated locations, G-Mapping struggles to recognize and appropriately close loops and that map may then become inconsistent as a result.

**Algorithm 1** G-Mapping Algorithm

**Require:** Laser scan data $L$, odometry data $O$
  Initialize empty grid map $m$ and robot pose $x$
  **for** each received laser scan $l$ and odometry data $o$ **do**
    Apply motion model to predict robot pose $x$
    Update grid map $m$ using $l$ and $x$
  **end for**

#### 2.3.1.7 Map Quality

Accurate sensor readings are essential to the map's quality. Noisy data can cause errors and have an impact on how well an algorithm performs. The map can be represented by three categories, Black refers to occupied represented by the number 100, white color refers to the not occupied which means empty or accessible place represented by 0, the third category is grey color refers to the not explored area which is represented by 1 is given in Figure 2.5.

**Figure 2.5** Map categories

#### 2.3.1.8 Computational Load

The particle filter-based technique can be computationally demanding, which might impair real-time performance in systems with limited resources.

G-Mapping has made great strides in the field of SLAM by tackling the difficult problem of simultaneous mapping and localization. Robots can independently traverse and map areas thanks to its grid-based mapping and particle filter-based localization. Despite its difficulties, current research aims to improve the algorithm and make it more durable, paving the way for wider implementation across other robotics applications. The ability of robots to comprehend and interact with their environment are still being shaped via G-Mapping[37–39].

### 2.3.2 Global cost map

The global cost map is created from the occupancy map, global cost map is created only once at the beginning, it contains costs from 0 – 255. 0 is free, and 255 is occupied. In-between score shows how close it is to obstacles.

**Pink –** Innermost layer. It is created from obstacles. (Lethal - 254) .

**Cyan –** Border around obstacles. Robots does not plan in this area to make sure it stays away from obstacles. (Inscribed - 253) .

**Red –** inflation layer. Robot will try to stay away from this area, but it can plan if the space is too narrow (Possible circumscribed - 128).

**Blue** – Free space, robot can plan and move through it without issues. (Free – 0) as shown in Figure2.6.

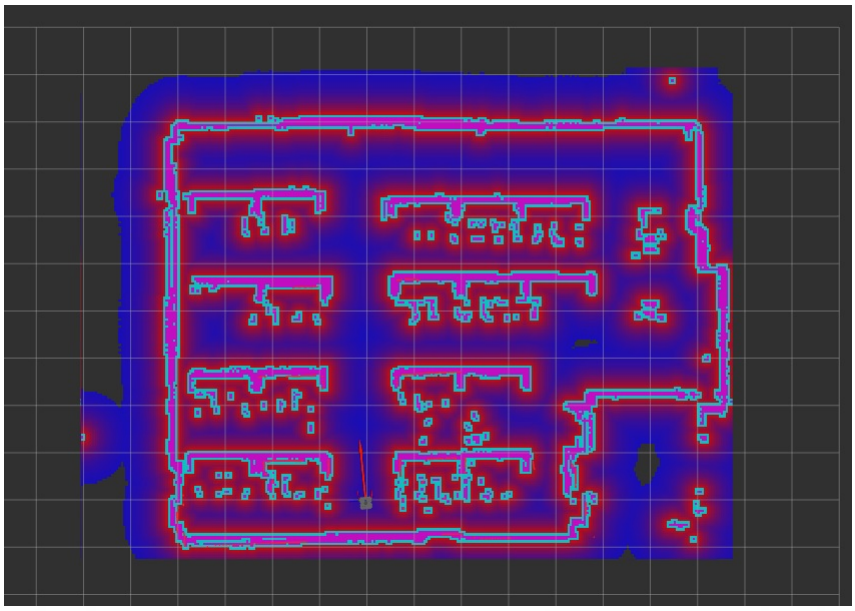


**Figure 2.6** Global costmap

### 2.3.3 Local cost map

Local costmap is created using both occupancy grid and laser data.Local costmap is updated repeatedly using laser data. It contains cost from 0 – 255. 0 is free, 255 is occupied. In-between score shows how close it is to obstacles

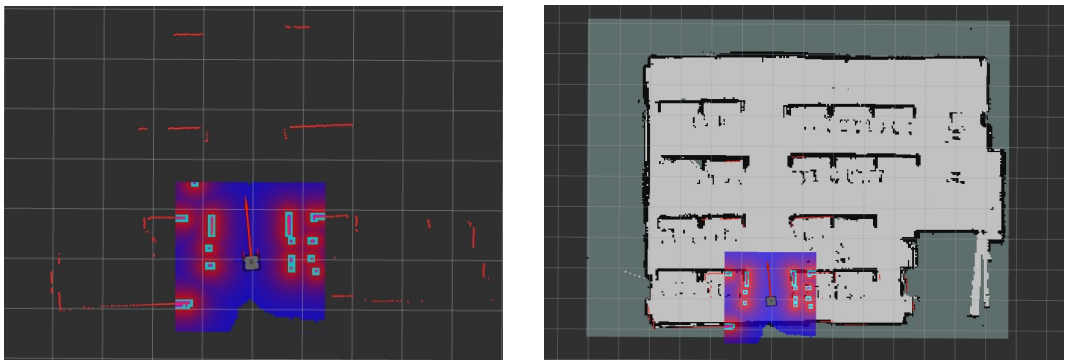
**Pink –** Innermost layer. It is created from obstacles. (Lethal - 254).

**Cyan –** Border around obstacles. Robot does not plan in this area to make sure it stays away from obstacles. (Inscribed - 253).

**Red –** inflation layer. Robot will try to stay away from this area, but it can plan if the space is too narrow (Possible circumscribed - 128).

**Blue –** Free space, robot can plan and move through it without issues. (Free – 0).

Local costmap is used by the local planner. The path with the lowest score is chosen and the robot goes in that direction. If there is no path (everywhere is lethal cost), then the robot stops and starts recovery behavior as shown in Figure 2.7.



a b

**Figure 2.7** Local costmap

## 2.4 Localization

Robotic systems depend on localization to establish their precise location inside a given area. It is essential to a variety of activities, including human collaboration and autonomous navigation. This thesis explores the foundations of localization, including major ideas, mathematical models, and discussing the inherent drawbacks of the process.

In the field of robotics, the phrase "localization" describes the method by which a robot establishes its location and orientation inside a defined area, often in relation to a coordinate system. This entails calculating the robot's posture (position and orientation) using the sensory data that is available and an environment map that has been previously created..

### 2.4.1 Frames of Reference and Coordinate Systems

Robots represent spatial information using a variety of coordinate systems, including polar and cartesian. Accurate localization requires a fundamental understanding of these systems.

### 2.4.2 Particle Filter

An application of the non-parametric Bayesian filter known as the particle filter is localization. It entails creating a collection of particles to symbolize the posterior belief. Inaccurate localization estimations may result from noisy or unclear sensor data. Occlusions and sensor drift are two additional factors that might make the problem worse. Changes in the environment, such as dynamic

obstacles or alterations in landmarks, pose challenges for maintaining accurate localization. Some localization techniques, particularly those based on particle filters, can be computationally intensive and may not be suitable for real-time applications. It might be challenging to pinpoint a robot's location in environments with comparable characteristics or a lack of recognizable landmarks. Robots can function well in a variety of situations because of localization, which is a key component of robotic autonomy. Researchers and practitioners may design reliable localization systems by grasping the underlying principles, using suitable mathematical models, and admitting their limits. This thorough review covers localization in robotics concepts, mathematical representation, and restrictions. It offers a strong basis for comprehending and using localization strategies.

### 2.4.3 Adaptive Monte Carlo Localization (AMCL) Algorithm

AMCL node a popular probabilistic approach for SLAM (Simultaneous Localization and Mapping) problems in mobile robots is adaptive monte carlo localization (AMCL). AMCL, which Dieter Fox created in the early 2000s, has established itself as a leader in the field of autonomous navigation. Using a particle filter architecture, this approach enables a robot to determine its pose (position and orientation within an environment). The unique property of AMCL is that it is adaptable, allowing it to change the number of particles according to the complexity of the environment and the caliber of the sensor data. This article offers a thorough analysis of the AMCL algorithm, encompassing its essential elements, underlying ideas, mathematical formulation, useful considerations, and notable applications. The particle filter, a Bayesian estimation method, lies at the heart of AMCL. It works by resembling the robot's position as a collection of weighted particles. Each particle contains a hypothesis about the robot's condition, such as its position and orientation.AMCL relies on sensor models to evaluate the likelihood of a given particle being the true robot pose. This involves comparing sensor measurements to expected readings based on the particle's hypothesis[40–42].

#### 2.4.3.1 Motion Model

The motion model predicts the robot's future state based on control inputs (e.g., velocity commands cmd vel). It accounts for the uncertainties associated with the robot's motion. In fundamental ideas to make an estimate of the robot's posture, AMCL makes use of the Bayesian filtering principles. It keeps a distribution across the potential postures, which is referred to as the posterior distribution, and it recursively updates this distribution whenever it receives new sensor data

or control inputs. The algorithm will first make a forecast, and then it will correct that prediction. The prediction stage involves the propagation of particles under the motion model. During the correction stage, the original weights of the particles are adjusted according to the probabilities of the observed sensor data.

**Prediction Step**

Sample a set of particles based on the motion model.

Apply control inputs to predict the next state of each particle.

**Correction Step**

Calculate the likelihood of sensor measurements given the particles' poses then update particle weights based on the calculated likelihood.

#### 2.4.3.2 Resampling

Select particles with higher weights with higher probability for the next iteration. Practical considerations first the number of particles one crucial aspect of using AMCL is determining the appropriate number of particles. Too few particles can result in poor localization accuracy, while too many can lead to computational inefficiencies. Second of that the sensor calibration, ensuring accurate calibration of sensors is essential for the reliable operation of AMCL. Inaccurate sensor readings can lead to incorrect pose estimates.

AMCL finds applications in various domains, including autonomous vehicles AMCL is employed in self-driving cars for accurate localization in dynamic environments. In robotics research, the researchers use AMCL to evaluate new localization techniques and algorithms.AMCL enables robots in warehouse automation to navigate and operate efficiently in industrial settings. The adaptive monte carlo localization (AMCL) approach has emerged as a significant technique in the field of mobile robots, providing strong and flexible solutions for localization issues. AMCL (Adaptive Monte Carlo Localization) is a robust method for accurately estimating the position and orientation (pose) of a robot in various surroundings. This is achieved by using bayesian filtering principles and merging sensor readings with motion models. By doing so, AMCL ensures that the pose estimate is both precise and dependable. Understanding its major components, mathematical formulation, practical concerns, and real-world applications equips academics and practitioners in the area of robotics to utilize the full potential of this powerful algorithm. This thorough research offers a complete examination of

the adaptive monte carlo localization (AMCL) technique, encompassing its core components, underlying concepts, mathematical formulation, practical concerns, and noteworthy applications. The green particles or arrows observed during robot navigation are commonly associated with the localization procedure in the AMCL (Adaptive Monte Carlo Localization) method. Within the realm of robotic localization, these elements serve as depictions of the particle filter employed by AMCL. Particle filters utilize a multitude of particles, typically represented as arrows or clusters, to estimate the robot's pose or position in its surroundings. These particles represent possible positions of the robot, which are determined based on sensor readings and movement predictions. They are used to create a probability distribution that helps determine the most likely position of the robot. While the robot is in motion and collecting sensor data, the particles continually update and come together to form more precise estimations. This process takes into account the uncertainty in localization and assists the robot in improving its position estimation as time progresses as shown in Figure 2.8 .

**Figure 2.8** AMCL localization

The capacity of a robot to plan and carry out its motions in a certain environment is known as navigation, and it is a fundamental component of robotics. An intricate interaction of sensors, algorithms, and actuators enables a robot to move around its environment on its own. This thesis offers a thorough investigation of robotic navigation, focusing on its fundamental ideas, working methods, and the crucial function it serves in allowing robots to carry out tasks across a range of industries.

## 2.5 Path finding

In the domain of pathfinding and graph traversal, Dijkstra's algorithm and A* Algorithm emerge as formidable cornerstones. This comprehensive endeavors to offer a profound comparative analysis of these two algorithms, illuminating their inception, underlying principles, diverse applications, strengths, and limitations. Through a meticulous exploration of the intricacies inherent to Dijkstra's and

A* algorithms, our objective is to foster a heightened comprehension of their applicability across a spectrum of scenarios. This endeavor, we believe, will empower researchers and practitioners alike to make informed decisions about the most fitting algorithm for their specific requirements.

The significance of pathfinding algorithms reverberates across a multitude of disciplines, spanning from robotics and interactive entertainment to transportation logistics and network routing. Within this array of algorithms, Dijkstra's and A* algorithms hold venerable positions, each distinguished by its unique attributes and functionalities.

Origins of Dijkstra's Algorithm, conceived by the eminent computer scientist Edsger W. Dijkstra in the seminal year of 1956, endures as a foundational pillar in the realm of graph theory. At its core, this algorithm is engineered to unearth the most concise route from a designated source node to all other nodes within a weighted graph. A distinguishing feature of this algorithm lies in its judicious utilization of a priority queue, enabling a methodical exploration of nodes in a breadth-first progression.

In a parallel vein, the A* Algorithm, introduced to the scientific community by Peter Hart, Nils Nilsson, and Bertram Raphael in the illustrious year of 1968, amalgamates the salient virtues of Dijkstra's approach with an astutely devised heuristic evaluation mechanism. This amalgamation bestows upon A* a remarkable efficacy, as it proficiently approximates the expense incurred in traversing from the commencement node to the intended terminus, rendering it exceptionally well-suited for a spectrum of pathfinding applications. The optimality of Dijkstra's Algorithm assures an optimal solution by meticulously examining all potential paths in a systematic manner. In contrast, the optimality of the A* Algorithm hinges on the admissibility of its heuristic; it guarantees the shortest path when the heuristic is admissible, but this assurance wavers if the heuristic proves inadmissible. The completeness of both algorithms demonstrates completeness, signifying that they unfailingly discover a solution if one exists. In time complexity computations Dijkstra's Algorithm operates at a time complexity of $O(V^2)$, with V denoting the count of vertices. In the expression $O(V^2)$, the O represents the order of growth or the upper limit of the algorithm's time complexity concerning the number of vertices (V) in a graph. Specifically, it indicates that the algorithm's time complexity grows no faster than a quadratic function of the number of vertices. This notation (O) is part of a family of notations used in algorithm analysis to denote how the time or space taken by an algorithm scales with the input size. Dijkstra's Algorithm is renowned for its straightforwardness and is particularly well-suited for situations

where achieving optimality is of utmost importance. However, it can encounter challenges when applied to extensive-scale applications due to its relatively higher time complexity.

On the other hand, the A* algorithm typically boasts superior time complexity, frequently surpassing Dijkstra's, particularly in scenarios involving extensive graphs. The A* Algorithm incorporates a heuristic function, a critical component that adeptly steers it towards the goal in an efficient manner. This is an attribute distinctly absent in Dijkstra's algorithm. Dijkstra's algorithm finds its niche in critical applications like network routing, robotics, and GPS systems, where precision in determining the absolute shortest path is paramount. Conversely, the A* algorithm excels in dynamic environments such as video games, robotics, and real-time systems. In these contexts, the emphasis lies on rapid and efficient pathfinding, even if it means compromising absolute optimality for expediency.In contrast, the A* algorithm strikes a balance between achieving optimality and operational efficiency. Nevertheless, its effectiveness is significantly contingent on the accuracy and effectiveness of the chosen heuristic.

Many hybrid algorithms combine elements of Dijkstra's and A* to optimize pathfinding in specific contexts.Dijkstra's algorithm and A* algorithm represent distinct approaches to pathfinding, each with its advantages and limitations. The choice between them depends on the application's requirements, computing resources, and the importance of optimality versus efficiency.Dijkstra's algorithm, conceived by Edsger W. Dijkstra in 1956, is hailed as a seminal contribution to the field of computer science, particularly in graph theory and pathfinding. However, this algorithm, while elegant and effective in many contexts, is not without its limitations. This literature review delves into a comprehensive analysis of these limitations, exploring the scenarios in which Dijkstra's algorithm falters and discussing alternative algorithms and techniques that address these shortcomings.

Dijkstra's algorithm is undeniably a cornerstone in the realm of pathfinding and graph traversal, celebrated for its ability to find the shortest path between two nodes in a graph. Nevertheless, understanding its limitations is crucial for making informed decisions when selecting pathfinding algorithms.One of the primary constraints of Dijkstra's algorithm lies in its inefficiency when confronted with large-scale graphs. The algorithm exhaustively investigates all potential pathways without any form of guidance, resulting in a time complexity of $O(V^2)$ . In scenarios involving graphs with a substantial node count, this computational demand becomes unfeasible.While Dijkstra's Algorithm assures optimality through its comprehensive exploration of all paths, it does so without the benefit of heuristic

guidance or estimated costs. In instances where a heuristic could offer a more streamlined search process, Dijkstra's algorithm exhibits limitations.Dijkstra's algorithm fundamentally operates within a static framework. It calculates paths predicated on the initial state of the graph, lacking the capacity to dynamically adjust to environments characterized by frequent alterations. Consequently, its applicability is limited in contexts demanding real-time responsiveness, such as robotics or systems necessitating continuous path adaptation.The introduction of the A* algorithm represented a pivotal advancement in pathfinding methodologies. Recognizing Dijkstra's deficiency in heuristic guidance, A* integrated a heuristic function into its framework. This strategic addition enabled A* to intelligently focus its search, effectively reducing the exploration of unpromising paths. Notably, when an admissible heuristic is employed, A* retains the crucial assurance of optimality, further enhancing its effectiveness in diverse applications.

Dijkstra has several modifications and hybrid approaches aim to mitigate the inefficiency of Dijkstra in dynamic environments. Dynamic Dijkstra variants adapt to changes in the graph over time, providing more efficient pathfinding. Dijkstra's algorithm, despite its limitations, retains its significance in scenarios where achieving optimality takes precedence and the graph in question is compact and stable. Its applications span a spectrum ranging from network routing to geographical information systems and static map-based pathfinding.As a trailblazer in path-finding algorithms, Dijkstra's algorithm provides a foundational understanding of graph traversal. Yet, its constraints, including the absence of an admissible heuristic and inefficacy in dynamic contexts, prompt the investigation of alternative algorithms. This comprehensive awareness of both its merits and limitations empowers researchers and practitioners to judiciously select the most apt pathfinding algorithm tailored to their specific requirements.

## 2.6 Global Planner

In the realm of autonomous robotics, effective navigation is a paramount concern. A critical component of this process is the global planner, tasked with charting a feasible path from the robot's current location to its designated goal within the environment[43, 44]. Among the plethora of algorithms at the disposal of roboticists, the A* algorithm stands as a beacon of efficiency and optimality. This comprehensive discourse embarks on an academic exploration of the A* algorithm, dissecting its core principles, mathematical underpinnings, variants, applications, strengths, and limitations.

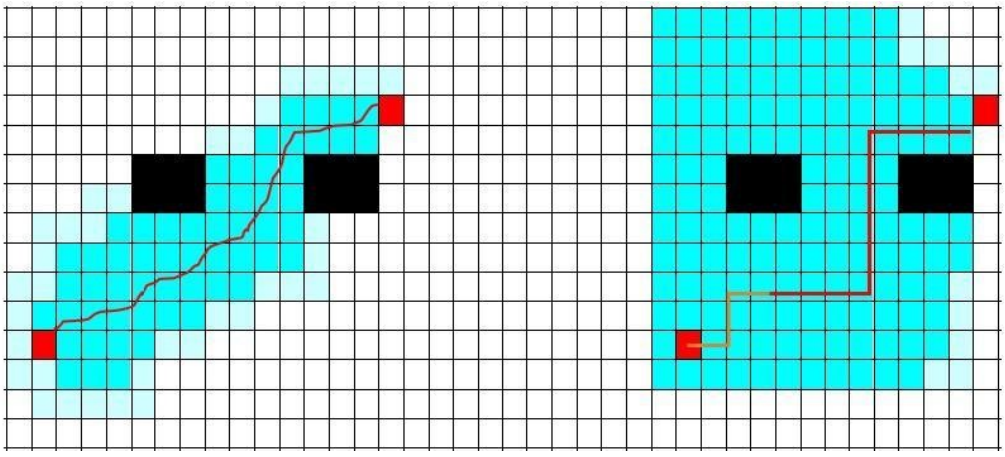


**Figure 2.9** The Dijkstra and A* models of exploratory behaviour are shown on the right and left, respectively [45]

The A* algorithm, which was formulated by Peter Hart, Nils Nilsson, and Bertram Raphael in 1968, is a product of a rich history of pathfinding algorithms. By using the advantages of both Dijkstra's algorithm and heuristic evaluation, this approach achieves a powerful combination of optimality and efficiency. The algorithm represents a groundbreaking advancement within the realm of artificial intelligence, fundamentally transforming the process of pathfinding across a range of domains, including robotics, video games, and network routing.

The A* algorithm utilizes a heuristic assessment as a fundamental component to direct its search procedure. The aforementioned heuristic serves as a means to approximate the cost between the initial node and the target node, functioning as a knowledgeable decision-making tool. By utilizing this heuristic in combination with a priority queue, the A* algorithm intelligently traverses the search space, methodically narrowing down the most favorable pathways[46].

### 2.6.1 Heuristic Functions

The selection of an appropriate heuristic function is a crucial part of the A* algorithm. The admissibility of this function is a necessary condition since it ensures that the cost estimation to attain the goal is never overestimated. Commonly used algorithms in grid-based applications encompass the Manhattan distance and the Euclidean distance. The selection of a heuristic significantly impacts the efficiency and optimality of the algorithm[47].

The A* algorithm ensures the attainment of an optimum solution when a heuristic function that satisfies the admissibility condition is utilized. This implies that the path it identifies is the most concise route from the initial point to the desired

destination. Moreover, the A* algorithm maintains completeness, guaranteeing that it will discover a solution, if one is there, as long as the search space is small.

A* algorithm's performance is contingent on the chosen heuristic and the nature of the search space. In the best-case scenario, with an ideal heuristic, A* can exhibit time complexity comparable to Dijkstra's algorithm, often outperforming it. However, in the worst case, when the heuristic is ineffective, A* may degrade to a breadth-first search, potentially incurring a high time complexity. Space complexity is also influenced by factors like the branching factor and depth of the search space.

Throughout its existence, the A* algorithm has undergone several changes and optimizations to address unique use situations. Weighted A* and jump point search (JPS) are alternative algorithms that provide improved efficiency in certain cases with distinct properties. Furthermore, the incorporation of strategies such as precomputation and hierarchical pathfinding has been implemented to enhance the efficiency of the A* algorithm.

Throughout its existence, the A* algorithm has seen several changes and optimizations in order to accommodate unique use situations. Weighted A* and jump point Search (JPS) are alternative algorithms that provide improved efficiency in certain cases with distinct properties. Furthermore, the use of methods such as precomputation and hierarchical pathfinding has been employed to enhance the efficiency of the A* algorithm.

Although the A* algorithm demonstrates exceptional performance in several contexts, it is not devoid of restrictions. The effectiveness of the system is highly dependent on the caliber of the heuristic employed, and in some scenarios, the algorithm may have difficulties in determining the optimal route. Current research efforts are focused on improving the process of heuristic selection and investigating the potential benefits of combining A* with other methodologies in a synergistic manner.

The A* algorithm is a fundamental component in the field of global planning, significantly transforming the process of pathfinding in autonomous systems. The combination of optimality and efficiency, together with its ability to be applied in a wide range of contexts, makes it an essential tool for professionals in the fields of robotics, game development, and routing engineering. As ongoing research endeavors further improve and enhance the aforementioned algorithm, the future of autonomous navigation has the potential for significant advancements in both efficiency and optimality.

### 2.6.2 Trajectory Planning

The control of trajectory planning methods employs kinematic equations to create the appropriate wheel velocities to achieve desired robot motions. These algorithms are frequently based on PID controllers or more sophisticated approaches such as Pure pursuit. Future kinematics research will focus on refining algorithms to account for uncertainties, dynamic situations, and non-ideal robot behaviors. Machine learning approaches are also being used to improve the precision of motion planning and control.ROS allows for seamless communication between different components of the system, enabling the wheeled mobile robot to perform accurate mapping and navigation tasks. A robotic operating system (ROS) solves the problem by serving as a flexible framework for robot software development. With easy access to code, robots with ROS became reachable to more around the world. This thesis proposes the implementation of an autonomous mobile robot based on ROS , control, and trajectory planning algorithms that use kinematic equations to generate the necessary wheel velocities to achieve desired robot motions. These algorithms are often based on global and local cost maps PID controllers and more advanced techniques like Pure pursuit. Navigation stack incorporating SLAM with G-Mapping and localization using AMCL navigation in robotics involves the autonomous movement of robots within an environment. Key to this is the integration of simultaneous localization and mapping (SLAM) for environment mapping and localization represented by the G-Mapping algorithm and Adaptive Monte Carlo Localization (AMCL), respectively. This navigation stack ensures accurate real-time mapping and localization for effective robot movement.

The process of simultaneous localization and mapping (SLAM) employing the G-Mapping algorithm initiates with the robot's sensors, such as lidar, cameras, or depth sensors, gathering comprehensive data about the surrounding environment. This acquired sensor data undergoes a meticulous feature extraction phase, where the system isolates significant landmarks, edges, or distinctive points crucial for constructing a comprehensive map of the environment. Subsequently, the G-Mapping algorithm leverages advanced probabilistic grid-based methodologies, utilizing the gathered sensor data to estimate the robot's precise location and generate a detailed map of the environment. Throughout this process, the map and the robot's position undergo continuous updates. Particularly noteworthy is the algorithm's capability to identify loop closures by recognizing previously visited locations, fine-tuning the map to rectify any positional deviations, and ensuring a consistent and precise representation of the mapped environment.

The localization utilizing adaptive monte carlo localization (AMCL) involves

leveraging the previously constructed map by the G-Mapping algorithm as a foundational guide for precise localization. AMCL initiates the process by establishing a set of particles, each representing a plausible robot pose within the map. Initially dispersed randomly, these particles serve as potential positions for the robot. Concurrently, the robot continuously captures sensor data, predominantly from lidar or comparable range sensors, integrating this incoming data into the particle filter for updating and thereby estimating the robot's location within the map. These particles undergo weighted sampling as the robot moves, altering their weights based on their compatibility with observed sensor data and the reference map. This process significantly bolsters the accuracy of localization estimations. Furthermore, AMCL incorporates adaptive particle resampling techniques, strategically concentrating particles in regions with higher probabilities of hosting the robot's genuine position. This adaptive approach intensifies the precision of localization, especially in environments marked by dynamic transformations or inherent uncertainties.

**PID Integration**

The amalgamation of Pure Pursuit with Proportional-Integral-Derivative (PID) controllers has emerged as a pervasive strategy in the field. This integration serves to augment tracking accuracy and bolster system stability, especially in scenarios demanding stringent precision. The incorporation of cutting-edge sensors, such as lidar and high-resolution cameras, has significantly broadened the algorithm's perceptual capacities. By leveraging these advanced sensory inputs, the algorithm gains a heightened ability to comprehensively perceive and interpret its surrounding environment. This extension equips the system to navigate through intricately detailed and dynamically changing terrains with greater efficacy. The integration of machine learning paradigms marks an exciting frontier in augmenting the capabilities of Pure Pursuit. The algorithm gains the ability to make more precise path predictions and dynamically navigate around obstacles. This integration has the potential to fundamentally transform the algorithm's adaptability and effectiveness across a wide spectrum of environments and scenarios.

Researchers are currently exploring hybrid methodologies that blend Pure Pursuit with complementary algorithms. This innovative approach is poised to yield outcomes of superior adaptability and performance. By leveraging the strengths of multiple techniques, these hybrid methodologies have the potential to overcome the limitations that are inherent in individual algorithms. This heralds a new era of navigational precision and reliability.The Pure Pursuit algorithm has solidified its position as a dependable and versatile tool in the toolkit of wheeled mobile robot

navigation. Its straightforward yet highly effective nature renders it an appealing choice for a diverse range of applications. While it does have its limitations, ongoing research and innovative strides are continuously broadening its capabilities, painting a promising future for this foundational algorithm.The interaction of the A* algorithm with Pure Pursuit for wheeled mobile robot navigation is an enthralling research and development topic. This study of the literature looks into the logic for this combination, its contributions to autonomous navigation, and the limits of other local planners in reaching equivalent outcomes.Autonomous navigation is a cornerstone in robotics, where wheeled mobile robots play a critical role. The pursuit of efficient, precise, and dynamic navigation has driven academics to experiment with novel algorithm combinations. Among them, the combination of the A* algorithm and Pure Pursuit has received a lot of interest. Peter Hart developed the A* algorithm in 1968, and it is now one of the most widely used pathfinding strategies in robotics and artificial intelligence. A* is well-known for its ability to locate the shortest path in a graph while accounting for the cost of traversal. Pure Pursuit, created by R.C. González in 1987, is developed for path tracking. It focuses on directing a robot along a preset path.

The geometric control by the algorithm that calculates the curvature of the route using geometric principles, ensuring smooth and exact tracking. The efficient of global route planning in A* is well-known for its global route planning skills, which can guide the robot across complex terrain. Pure Pursuit excels in local tracking and ensuring that the robot follows the set course. The dynamic adaptability comes from The combination enables robots to adjust to changing circumstances dynamically by recalculating the path using A* while retaining smooth tracking with Pure Pursuit. A*'s efficiency in computing global pathways supplements Pure Pursuit's real-time tracking, assuring prompt answers.

On the other hand, due of its conservative character, the dynamic window approach (DWA) is an alternative local planner that may struggle to adapt to extremely dynamic contexts. timed elastic band (TEB) provides flexibility, it may not always give ideal pathways, particularly in cases demanding strict path adherence. Machine learning approaches are being investigated in order to further improve the flexibility of the A*-Pure Pursuit combination.

Researchers are looking at multi-objective optimization approaches to balance global planning efficiency and local tracking precision. The combination of the A* algorithm with Pure Pursuit has enormous potential for developing wheeled mobile robot navigation. It blends global planning power with local tracking dexterity, allowing robots to navigate complicated settings effectively and securely.

While other local planners exist, their dynamic flexibility and real-time efficiency are sometimes limited, making the A*-Pure Pursuit combination a tempting choice for academics and roboticists alike.WMRs (wheeled mobile robots) have proven vital in a variety of fields, ranging from logistics and manufacturing to search and rescue. Effective trajectory tracking, achieved by sophisticated control algorithms, is a critical feature of their autonomous functioning. Among them, the Proportional-Integral-Derivative (PID) trajectory controller is a popular and well-liked option. This literature review investigates the several benefits that PID trajectory controllers provide in improving the navigation, stability, and efficiency of WMRs.Because of their adaptability, maneuverability, and efficiency, wheeled mobile robots have found applications in a wide range of sectors. For these robots to navigate complicated settings and do jobs reliably and securely, effective trajectory tracking is critical.The PID controller has a long history stretching back to the early twentieth century, making it one of the most mature control algorithms.The Three components of PID controllers include proportional, integral, and derivative terms, which enable varied control over a system's response. ID controllers help WMRs stay stable by constantly altering control inputs to maintain desired trajectories.PID controllers are appropriate for a wide range of robotic platforms due to their simple construction and intuitive calibration.

**Enhanced Path Tracking**

The Accuracy of PID controllers excel at accurate route tracking, ensuring WMRs follow prescribed trajectories.They effectively manage disturbances, ensuring path tracking remains dependable even in difficult conditions.PID controllers provide low-latency control responses, which are critical in dynamic and fast changing situations.Their deterministic nature assures predictable reactions, which is critical for safe navigation.PID settings may be modified to adapt to various payloads, terrains, and jobs, enabling adaptability in robot deployment. Changes in system dynamics may be handled by adjusting parameters, which simplifies system adaption. PID controllers seamlessly fuse with sensory data, allowing robots to adapt to environmental signals in real time.Sensory feedback allows PID controllers to make continual modifications even when there are uncertainties.PID controllers aid in the definition of energy-efficient pathways, which contributes to longer battery life and lower operational expenses. In dynamic Performance the Real-time modifications enable energy-efficient manoeuvres, particularly in dynamic settings. Despite PID benefits, The nonlinearities of PID controllers may encounter difficulties when dealing with nonlinearities. To overcome these difficulties, advanced PID variations and adaptive control techniques are emerging. Changing or unexpected situations may present difficulties. The research focuses

on integrating alternative control paradigms to deal with system regularity .PID trajectory controllers have shown to be quite useful in improving the navigation and control of wheeled mobile robots. Because of their benefits in terms of stability, simplicity, accuracy, and real-time control capabilities, they have become a standard in robotics. Current research, however, strives to overcome these shortcomings and improve their performance in dynamic and unpredictable contexts[24, 25, 48].

## 2.7 Local Planner

The present study examines the dynamics of local planners, specifically focusing on three widely used algorithms Pure Pursuit, Dynamic Window Approach (DWA), and Timed Elastic Band (TEB).Within the complex realm of robotics, local planners assume a crucial role in facilitating the real-time navigation of mobile robots. The algorithms are assigned the intricate duty of assuring the safety and accuracy of movements in dynamic conditions. Within the realm of local planners, three notable challengers have surfaced, namely Pure Pursuit, Dynamic Window Approach (DWA), and Time-Elastic Band (TEB). This extensive investigation undertakes an academic analysis of these acclaimed algorithms, diving into their basic concepts, mathematical underpinnings, applications, strengths, and potential limits.

### 2.7.1 Pure Pursuit Algorithm

The Pure Pursuit algorithm, introduced by R.C. González in 1987, remains a cornerstone of wheeled mobile robot navigation. At its core, Pure Pursuit employs principles of geometry and trigonometry to guide a robot along a predefined path while maintaining a constant look-ahead distance. This elegant concept forms the foundation of precise trajectory tracking in dynamic environments. The Pure Pursuit algorithm stands as a cornerstone in the realm of wheeled mobile robot navigation, offering fundamental and adaptable capabilities. This academic review sets out on a rigorous journey through the algorithm's inception, versatile applications, and evolutionary trajectories. It ventures into the theoretical foundations as well as real-world implementations, casting illumination on its robust attributes, identifiable constraints, and emerging paradigms within the field.In the realm of mobile robotics, particularly those endowed with wheels, the imperative of precise and efficient navigation is indisputable. Within this context, the Pure Pursuit algorithm has emerged as an illustrious choice, celebrated for its innate simplicity, potent effectiveness, and remarkable versatility across a spectrum of robotic platforms.Originating in 1987, the Pure Pursuit algorithm made its debut under the insightful guidance of R.C. González, initially conceived

as a guidance law tailored for autonomous vehicles [26, 27]. Over the years, it has evolved into a foundational pillar in the field of mobile robotics, particularly renowned for its prowess in trajectory tracking.Underpinning the Pure Pursuit algorithm lies a robust mathematical foundation, intricately woven with principles of geometry and trigonometry. This intricate framework empowers the algorithm to adeptly steer a robot along a predetermined path, all the while ensuring a steadfast adherence to a prescribed look-ahead distance. This amalgamation of mathematical principles endows Pure Pursuit with a unique and potent navigational prowess. At the crux of the Pure Pursuit algorithm lies the profound concept of look-ahead control, a pivotal principle in autonomous navigation. Within this paradigm, the robot adeptly identifies and designates a strategic point residing ahead along the predefined path. This prescient selection serves as the linchpin in orchestrating the robot's precise and efficient motion. Deep within the algorithm's mathematical core, a sophisticated interplay of geometric calculations unfolds. These intricate computations ingeniously determine the curvature of the path requisite for the robot's journey toward the selected look-ahead point. In essence, this dynamic process crafts a meticulously smooth trajectory, thereby ensuring the robot's harmonious traversal along its intended course. It is through this intricate fusion of mathematical ingenuity and anticipatory control that the Pure Pursuit algorithm achieves its exceptional navigational finesse.

The Applications of the Pure Pursuit algorithm span a diverse array of domains, each capitalizing on its precision and effectiveness, The path following in Pure Pursuit algorithm finds extensive utility in the realm of path following, solidifying its position as a cornerstone of autonomous navigation for ground-based vehicles. Its capacity to seamlessly track predefined trajectories ensures efficient and reliable motion. These applications underscore the algorithm's adaptability and underscore its role as an invaluable asset in diverse domains where precise and agile navigation is paramount.In complex environments characterized by high clutter or rapid, dynamic changes, the algorithm may face difficulties in swiftly and accurately adapting to evolving conditions. The need for precise path following can be hindered by the complexity and unpredictability of such surroundings. These limitations highlight scenarios where the Pure Pursuit algorithm may encounter hurdles, underscoring the importance of considering alternative approaches in certain contexts.

The fundamental principle of the Pure Pursuit algorithm revolves around the notion of look-ahead control, wherein the robotic system strategically identifies a point situated ahead of its current position along the intended trajectory. This particular position, commonly known as the "target point," determines the steering action

**Algorithm 2** Pure Pursuit with PID controller

**Require:** RobotPose, RobotGoal, GlobalPath
**Ensure:** v and w
  **while** true **do**
    Average_cost = calculate_cost(costmap)
    Goal = getTargetPoint(GlobalPath, RobotPose, LookAhead)
    Error_distance, Error_orientation ← calculate_errors(RobotPose, Goal)
    **if** Average_cost $< 80.0$ **then**
      **if** Error_distance $> 0.5$ **then**
        w ← saturate(PID($Error\_orientation$))
        v ← saturate(PID($Error\_distance$))
      **else**
        Goal reached
        **return** true
      **end if**
    **else**
      **return** false
    **end if**
  **end while**
  **Function** calculate_errors(RobotPose, RobotGoal)
  Error_distance ← get_euclidean_distance($RobotGoal, RobotPose$)
  Error_orientation ← get_error_orientation($RobotGoal, RobotPose$)
  **return** Error_distance, Error_orientation
  **End Function**
  **Function** getTargetPoint(GlobalPath, RobotPose, LookAhead)
  $N$ ← path_size($GlobalPath$)
  distance $= 0$
  **for** $i < N$ **and** distance $<$ LookAhead **do**
    distance ← get_euclidean_distance($RobotGoal, RobotPose$)
    **if** distance $>$ LookAhead **then**
      Target_point ← GlobalPath[$i$]
    **end if**
  **end for**
  **return** Target_point
  **End Function**

of the robot. The method computes the curvature of the trajectory required to reach the designated location, so assuring a seamless path.The Pure Pursuit algorithm is widely employed in several disciplines, such as route following tasks and robotics contests. The significant attention received by this approach may be attributed to its simplicity, efficacy, and application across many robotic platforms. The algorithm demonstrates exceptional performance in situations when accurate trajectory monitoring is of utmost importance.

Nevertheless, it is important to acknowledge that Pure Pursuit does have certain restrictions. Trajectory tracking might be challenging due to the presence of non-holonomic limitations, which are frequently encountered in wheeled mobile robots. In surroundings characterized by high levels of clutter or frequent changes, the algorithm may have difficulties in adapting rapidly, which might result in performance that is less than ideal as in Figure 2.10.

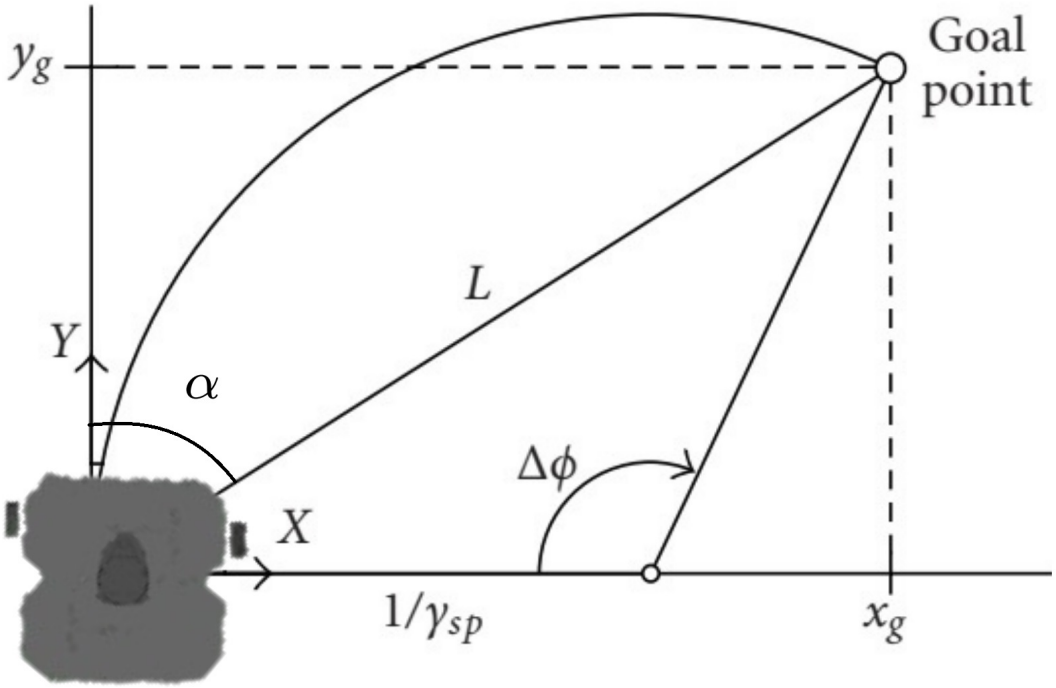



**Figure 2.10** Pure pursuit tracking concept

The curvature $\gamma$ of the vehicle is the reciprocal of the distance $r$ between the vehicle's frame origin and its instantaneous center of rotation (ICR), taking into account the sign. In other words, it is the inverse of $\gamma$ set point equal to the radius of the steering circle.

$\Delta\phi$ represents the change in direction of the vehicle as it moves along the arc.

The coordinates (xg, yg) of the target point could be determined geometrically as

follows:

$$
\begin{aligned}
x_g &= \frac{\cos(\Delta\phi) - 1}{\gamma_{sp}}, \\
y_g &= \frac{\sin(\Delta\phi)}{\gamma_{sp}},
\end{aligned}
\tag{2.10}
$$

The Pure Pursuit algorithm involves the calculation of the steering angle for a vehicle to follow a desired path defined by a set of waypoints. The mathematical expression for the Pure Pursuit algorithm's steering angle calculation is as follows:

**Algorithm 3** Steering Angle Calculation

**Require:** vehicle_base_length, $\alpha$, L
**Ensure:** steering_angle
$steering\_angle \leftarrow \arctan\left(\frac{2 \times \text{vehicle\_base\_length} \times \sin(\alpha)}{L}\right)$

Let $L$ be the look-ahead distance, which represents the distance between the vehicle's current position and the lookahead point on the path.

The formula for calculating the steering angle ($\delta$) in the Pure Pursuit algorithm where:

- $\alpha$ is the angle between the vehicle's orientation and the line connecting the vehicle to the lookahead point.
- `vehicle_base_length` is the distance between the vehicle's front and rear axles.

The lookahead point is determined by selecting a point on the path at a distance of $L$ ahead of the vehicle's current position.

$\Delta\phi$ represents the change in direction of the vehicle as it moves along the arc

The steering angle $\delta$ calculated by this formula is used to control the vehicle's steering mechanism to follow the desired path.

### 2.7.2 Dynamic Window Approach (DWA)

The Dynamic Window Approach (DWA) is a highly competitive challenger within the domain of local planners. The Dynamic Window Approach (DWA) was created in response to the shortcomings of grid-based approaches. It accomplishes this by

dynamically assessing the potential future states of a robot, taking into account both its kinematic restrictions and the surrounding environment. The ideal control input is determined using the Dynamic Window Approach (DWA) through the process of iteratively sampling and assessing possible velocities.

The DWA algorithm utilizes a velocity space with two dimensions in order to calculate viable control inputs. Through the process of superimposing the present velocity of the robot with a collection of measured velocities, the Dynamic Window Approach (DWA) is able to discern the range of permissible velocities, referred to as the "dynamic window." The evaluation of trajectories under this framework involves considering their closeness to the desired objective as well as the possibility of encountering obstacles.

DWA demonstrates exceptional performance in situations including non-holonomic limitations, enabling robots to efficiently navigate inside restricted settings. The capacity to flexibly adjust to the robot's functionalities and alterations in the surrounding area renders it a formidable selection for navigating in real time.

Nevertheless, the implementation of DWA presents several obstacles. The algorithm's performance is contingent on accurate models of the robot's dynamics and environment. The presence of inaccuracies or uncertainties within these models may result in trajectories that are sub optimal or potentially harmful as shown in Figure 2.11 .

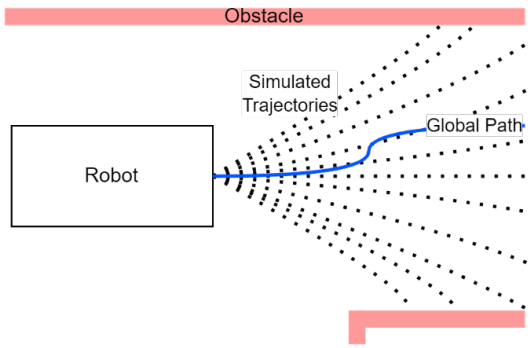



**Figure 2.11** Dynamic Window Approach DWA path tracking

### 2.7.3 Time-Elastic Band (TEB)

The Time-Elastic Band (TEB) algorithm represents a pioneering effort to address the challenges posed by time-varying environments. TEB employs a space-time representation to account for temporal aspects in addition to spatial constraints. By modeling the robot's trajectory as a "band" in space and time, TEB ensures safe and

**Algorithm 4** Dynamic Window Approach (DWA)

**Require:** Robot state: $x, y, \theta$
  Initialize best_v = 0, best_w = 0
  **for all** linear velocity $v$ in the range $[v_{\text{min}}, v_{\text{max}}]$ **do**
    **for all** angular velocity $\omega$ in the range $[\omega_{\text{min}}, \omega_{\text{max}}]$ **do**
      Predict robot's state: $x_{\text{new}}, y_{\text{new}}, \theta_{\text{new}}$
      Compute evaluation function $f(x_{\text{new}}, y_{\text{new}}, \theta_{\text{new}})$
      **if** $f(x_{\text{new}}, y_{\text{new}}, \theta_{\text{new}})$ is better **then**
        Update best_v = $v$, best_w = $\omega$
      **end if**
    **end for**
  **end for**
  **return** Best linear and angular velocities: best_v, best_w

efficient navigation.

TEB leverages mathematical formulations involving spatiotemporal constraints. The algorithm considers the robot's kinematic limits, along with obstacles in the environment, to compute a safe trajectory. By dynamically adjusting the size of the elastic band, TEB accommodates changes in the environment over time.TEB excels in scenarios where the environment is subject to temporal variations, such as moving obstacles or dynamic changes in terrain. Its ability to account for both spatial and temporal constraints makes it a robust choice for complex, time-varying environments.

Despite its efficacy, TEB is not without its complexities. Implementing the algorithm requires a detailed understanding of the robot's dynamics and the ability to model the environment accurately. Additionally, tuning parameters to achieve optimal performance can be a non-trivial task.

In the ever-evolving landscape of robotic navigation, local planners stand as the linchpin for safe and precise motion. Pure Pursuit, DWA, and TEB represent three distinguished approaches, each with its unique strengths and potential challenges. While Pure Pursuit emphasizes precision in trajectory tracking, DWA focuses on dynamic adaptation to the environment, and TEB addresses the intricacies of time-varying scenarios. Understanding the nuances of these algorithms empowers roboticists to select the most suitable approach for their specific applications, ultimately propelling the as shown below2.12

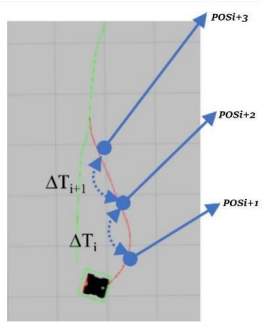



**Figure 2.12** Time Elastic Band TEB tracking

**Algorithm 5** Timed Elastic Band (TEB) Algorithm

**Require:** Robot state: $x, y, \theta$
**Require:** Time: $t$
Initialize trajectory: $T = \emptyset$
Generate feasible trajectories in the time interval $[t, t + \Delta t]$
**for all** feasible trajectory $t_i$ **do**
Compute cost: $c_i = \text{evaluateCost}(t_i)$
**end for**
Select trajectory with minimum cost: $T_{\text{min}} = \arg\min_i(c_i)$
**return** Best trajectory: $T_{\text{min}}$

# 3
# PROPOSED SYSTEMATIC DESIGN

In this section, our endeavor pivots around a meticulous analysis and synthesis of pivotal insights in the design of our proposed navigation model in both simulation and real platforms. Our core aim is the formulation of a navigation model, underscoring the indispensable need for an adaptable control framework. This envisioned system is poised to significantly enhance the navigational prowess of automated guided vehicles (AGVs) employed within the realms of smart warehousing and manufacturing domains. A paramount objective is to fortify the dependability of the non-collision system, effectively curbing the probability of mishaps within these intricately dynamic operational settings. so the proposed design has to be implemented in both simulation and real platform.

## 3.1 Settling on an operating system for the robot

ROS1 is chosen over ROS2, due to the availability of a wide range of packages that are important for this project. For instance, rf2o is a laser odometry package, the move_base stack provides Dijkstra and A* algorithms which are used as a reference implementation.

## 3.2 Proposed Low-Resolution Grid

Low-resolution grid maps constitute a pivotal concept within the realm of robotics and navigation systems, serving as indispensable tools for simplifying environmental representations. These maps play a crucial role in facilitating path planning and decision-making processes. This comprehensive exploration aims to delve deeply into the realm of low-resolution grid maps, delving into their defining attributes, creation methodologies, manifold advantages, and diverse applications in the domain of robotics. Essentially, a low-resolution grid map functions as a discretized representation of the environment, wherein each grid cell corresponds to a distinct area or region, marked as either accessible or inaccessible. Unlike

their high-resolution counterparts, which offer intricate details and precise accuracy, low-resolution grid maps adopt larger grid cells, encompassing broader regions and consequently portraying a less detailed representation of the environment. These lower resolution maps prove particularly beneficial in resource-constrained robotic systems owing to their reduced granularity, which, in turn, enables more efficient computational processes and optimized memory usage3.1.

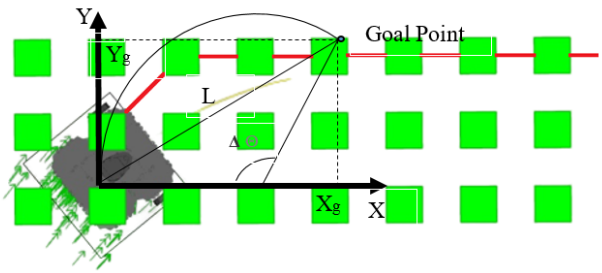



**Figure 3.1** Low-resolution grid of markers

The creation of low-resolution grid maps involves a critical process of discretizing the continuous environmental data into discrete cells. This intricate process encompasses the partitioning of the environment into a grid-based structure, assigning specific values or labels to each cell based on the accessible or not accessible corresponding area. These attributes typically encompass the occupancy status of the area (whether occupied, free, or unknown), terrain characteristics, or other pertinent information crucial for facilitating robot navigation within the environment. One of the major benefits of adopting low-resolution grid maps is that they lower the computational complexity of path-planning algorithms. Because low-resolution grid maps include fewer cells than high-resolution grid maps, planning algorithms can run quicker, allowing for real-time planning and execution in dynamic contexts. Furthermore, because low-resolution grid maps need less memory, they are suited for implementation on resource-limited platforms such as embedded computers and mobile robots. This benefit is especially relevant for autonomous robots that must function in real-world contexts where processing power and memory resources are frequently limited. Low resolution grid maps have a wide range of applications in robotics and navigation. They're frequently employed in route planning algorithms to generate collision-free courses for robots, allowing them to explore complicated surroundings while avoiding obstacles. Low-resolution grid maps are also used in localization and mapping activities, where they can assist robots in estimating their position and creating maps of their surroundings. Low-resolution grid maps are critical in robotic exploration missions for route planning and coverage optimization. Robots can effectively explore and map huge regions with minimum processing cost by employing low-resolution grid

maps.

Low-resolution grid maps are very useful in multi-robot systems, where communication and coordination between robots are vital. The use of low-resolution grid maps facilitates robot communication and data sharing, allowing for effective collaboration and work distribution. Low-resolution grid maps are an effective tool in robotics and navigation systems, providing a good mix of processing efficiency and representation accuracy. They are useful for resource-constrained robotic systems because of their ability to decrease computing and memory needs, and their applications range from path planning and navigation to mapping and exploration. Low-resolution grid maps will definitely remain an important component in enabling robots to move, explore, and interact in a restricted area with their environments as robotics advances.

Auto Markers is a code of C++ language program used to generate markers for any given map. It reads the map topic from the map server and publishes visualization markers that are away from obstacles. There is nothing related to the map to change in auto markers. One parameter can be tuned to generate markers far or close to an obstacle.

#### 3.2.0.1 Functions

**main:** This function creates the ROS node and sets up the subscribers and publishers. It uses the map callback function to receive the map once and generates the markers on free space. Later it continuously publishes the marker to be visible on Rviz.

**mapCallback:** This callback function is used to receive the map from map_server and save it in a list.

**isClosetoObstacle:** This function is used to check if a cell in the map is too close to the obstacle. If it is not, and if the cell is free, it is marked using a marker.

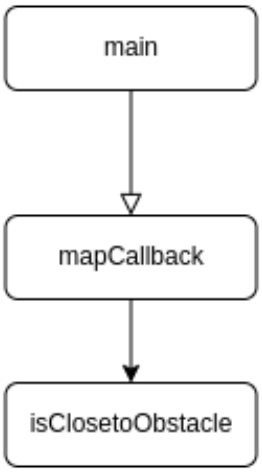



**Figure 3.2** Auto marker functions

#### 3.2.0.2 Combining Auto Markers with Dijkstra Planner

This planner is the same as the Euclid planner but works with auto markers. It has the same functions, the only addition is the mapCallback and isClosetoObstacle functions from the automarkers code. It has the same parameters as Euclid's Planner

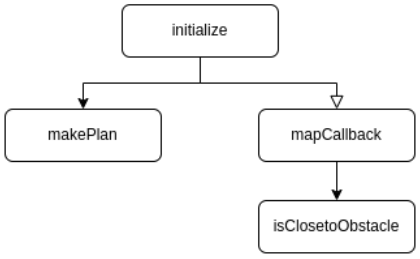



**Figure 3.3** Euclid planner with marker functions

#### 3.2.0.3 Combining Auto Markers With Astar Planner

This planner is the same as A* planner but works with auto markers. It has the same functions, so, The only addition is the mapCallback and is ClosetoObstacle functions from the automarkers code. It has the same parameters as Astar planner .

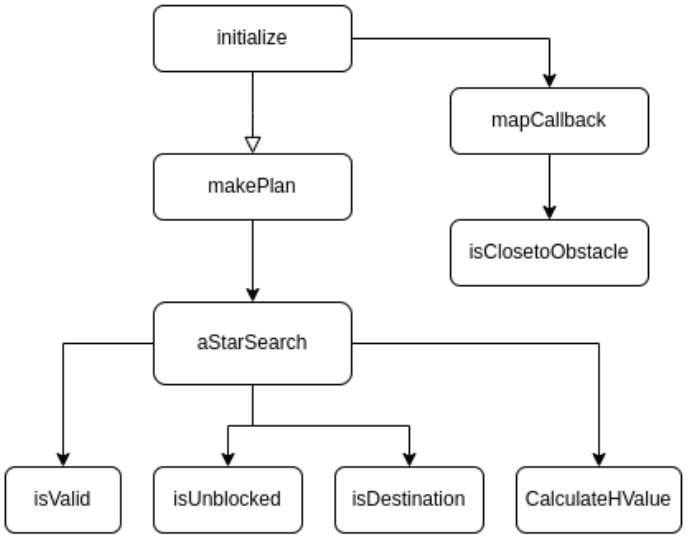



**Figure 3.4** A* planner with marker functions

## 3.3 Proposed Navigation Model Design

To delineate our navigation system, it is imperative to meticulously outline and define the algorithms that will be instrumental in crafting an enhanced non-collision system.

The odometry node (laser odometry or wheel odometry) will calculate the movement of the robot. This is a rough estimate of the actual movement, the SLAM node will take this estimate and the laser data. It will combine multiple laser data to form a map of the place. The odometry node (laser odometry or wheel odometry) will calculate the movement of the robot. This is a rough estimate of the actual movement. The AMCL node will take this rough estimate. It will use a particle filter to match the current laser data with the existing map. This way it will calculate an improvement to the position. This is a more accurate estimate of the actual movement.

The Global planner will take the accurate estimate from the AMCL node, and calculate a path from the robot's current position to the goal. It will use global costmap and markers to avoid obstacles and plan a path.

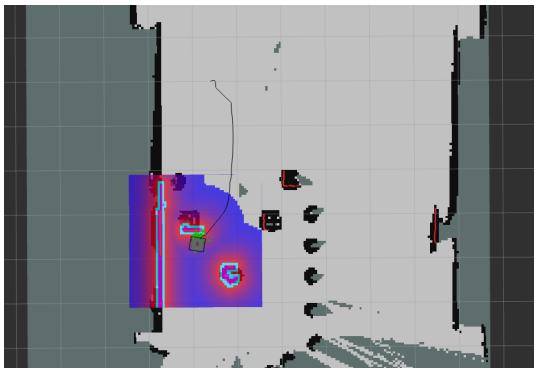


**Figure 3.5** Local cost map and path planning

The local planner will take the path from the global planner and the position estimate from the AMCL, and generate cmd_vel to control the robot and reach the goal. It will use a local costmap to avoid obstacles close to the robot.

The velocity control will take the cmd_vel from the local planner, and convert it to signals for the left and right wheel as shown in Figure 3.6.

### 3.3.1 A Comparative Analysis of Dijkstra's Algorithm and A* Algorithm

Dijkstra's algorithm and the A* algorithm are two fundamental methods in the field of path finding and graph traversal. While both serve the purpose of finding the shortest path between nodes in a graph, they employ different strategies and heuristics, leading to distinct strengths and weaknesses.

#### 3.3.1.1 Dijkstra's Algorithm

Dijkstra's algorithm is a classic approach to finding the shortest path between nodes in a weighted graph. It achieves this by maintaining a priority queue of nodes and continuously selecting the node with the smallest distance from the source. It then relaxes the neighboring nodes, updating their distances if a shorter path is found. One of the advantage*s* of Dijkstra is completeness, Dijkstra's algorithm is guaranteed to find the shortest path in any graph with non-negative edge weights. It explores all possible paths, ensuring optimality. Dijkstra's applicability in handling graphs with varying edge costs makes it suitable for a wide range of real-world applications. On the other hand, the Dijkstra limitation is an Inefficiency with negative weights, It is not designed to handle graphs with negative edge weights. In such cases, it may produce suboptimal or incorrect results. Dijkstra's algorithm time complexity is O((V + E) log V) with a priority queue, which can be impractical for large graphs.

**Algorithm 6** Dijkstra's Algorithm

**Require:** Graph $G = (V, E)$, source vertex $s$
  $dist[s] \leftarrow 0$
  $Q \leftarrow$ priority queue initialized with vertices and their distances ($\infty$ initially)
  **while** $Q$ is not empty **do**
    $u \leftarrow$ vertex in $Q$ with the minimum distance
    **for all** neighbor $v$ of $u$ **do**
      **if** $dist[u] + w(u, v) < dist[v]$ **then**
        $dist[v] \leftarrow dist[u] + w(u, v)$
      **end if**
    **end for**
  **end while**

#### 3.3.1.2 A* Algorithm

The A* algorithm is an informed search algorithm that combines elements of both uniform-cost search (similar to Dijkstra's) and a heuristic evaluation function. It estimates the cost of reaching the goal from a given node and uses this estimate to prioritize exploration. The advantages of A* is the efficiency with heuristic its typically more efficient than Dijkstra's algorithm, especially when a well-informed heuristic is available. It guides the search towards the goal, reducing the number of nodes to explore.Moreover the choice of heuristic allows for flexibility in the algorithm's behavior, enabling it to be tailored to specific problem domains. But on the other hand, The limitations of heuristic accuracy and the effectiveness of A* heavily relies on the quality of the heuristic function. An inaccurate heuristic can lead to sub optimal paths. Furthermore, the completeness under heuristic Conditions, while A* is optimal with an admissible heuristic, it may not always find the shortest path if the heuristic is inadmissible. The choice between Dijkstra's algorithm and A* algorithm depends on the characteristics of the problem at hand. Dijkstra's algorithm is a versatile option for scenarios without a well-defined heuristic, while A* shines in situations where a good heuristic can significantly improve efficiency. When implemented with proper consideration, both algorithms are powerful tools for solving various pathfinding problems. The determinations of the above mentioned that A* is better in path planning due to the heuristic function, but the heuristic function does not choose the shortest path in all of the cases relies on the quality of that function. So we are trying to combine markers as a low-resolution grid to be taken into account as a pathway during path planning.

### 3.3.2 Proportional Wheels Velocity Controller:

**Proportional Velocity Wheels Speed Controller has the following functions:**

The controller calculates the error difference between the target wheel speed (setpoint) and the actual wheel speed (process variable) on a continuous basis. The difference, known as the mistake, serves as the foundation for the control action.

**Proportional Control:** The proportional controller component is in charge of generating a control output proportionate to the error. It changes the wheel speed dependent on the amount of the mistake, resulting in a quicker reaction as the error rises. The proportional gain (Kp) controls the strength of this corrective action.

**Output Calculation:** The controller computes the control output by the proportional components. This output is sent to the motor of the robotic platform, which adjusts the wheel speed and aligns it with the desired setpoint.

**Closed-Loop Control:** The Proportional velocity wheels speed controller operates in a closed-loop mode, in which the controller continually checks the wheel speed and changes the control output in real-time. The controller may use this closed-loop mechanism to adjust any deviations from the target speed, resulting in accurate and steady wheel motion.

**Proportional Velocity Wheels Speed Controller's Philosophy:**

The Propotional velocity wheels speed controller is based on the principle of attaining accurate and efficient motion control in robotic systems. The major element of control philosophy is the Propotional controller is built to be stable and robust in the face of shocks and uncertainty. The integral component aids in the elimination of steady-state errors produced by variables like as friction, inertia, or external disturbances, ensuring the robot's mobility is precise and stable. The propotional controller's component responds quickly to changes in the target speed or external effects. The controller responds with a greater corrective action as the mistake rises, allowing for fast modifications to attain the target speed.The propotional controller balances simplicity and efficacy. It is simple to build and tune, making it a popular choice for a variety of robotic applications.An inherent component of the controller allows it to adjust to changing conditions and maintain accurate speed control under variable load situations or environmental changes.

**Real-Time Control:** The Propotional velocity wheels speed controller runs in real-time, updating the control output constantly depending on the detected speed and setpoint. This real-time feedback loop allows the controller to make quick modifications and ensure accurate velocity control while the robot is in operation. The propotional velocity wheels speed controller is a crucial control approach for regulating wheel speed in robotic systems. Its functions include error

calculation, proportional and integral control actions, and providing a control output to accomplish the required speed. Because the controller's philosophy emphasizes robustness, responsiveness, simplicity,flexibility, and real-time control, it is a vital component in obtaining accurate and efficient motion control in robotic systems as shown in Figure3.6.

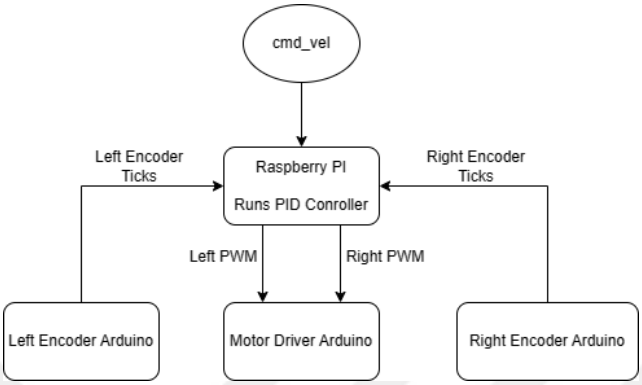



**Figure 3.6** Wheel velocity controller

### 3.3.3 PID Trjectory Contoller

The PID (Proportional-Integral-Derivative) controller holds a prevalent position in mechanical control systems owing to its simplicity, reliability, and ease of implementation. Its constitution encompasses three integral constituents,the proportional, integral, and differential components. Functionally, the proportional control aspect within the PID controller serves to adjust the error value's magnitude, the integral control component adeptly eradicates the persistent error emerging during the process control, while the differential control component stipulates the rate at which errors are rectified. The schematic flow of the PID controller is visually elucidated in Figure 3.7.

### 3.3.4 Pure Pursuit with PID Controller

Motion planning and trajectory tracking are key parts of mobile robot navigation, allowing robots to move across complicated situations effectively and precisely. Pure Pursuit is a well-known trajectory tracking technique for steering wheeled mobile robots along a chosen path while minimizing tracking error , The Euclidean distance formula is used to calculate error distance as below .

$$\boldsymbol{d}(\boldsymbol{p},\boldsymbol{q})^2 = (q_1 - p_1)^2 + (q_2 - p_2)^2 \tag{3.1}$$

A PID (Proportional-Integral-Derivative) controller is frequently included to

manage the robot's velocity and provide a smoother trajectory following to improve the performance and stability of the Pure Pursuit algorithm. This section delves into the idea, implementation, and benefits of employing Pure Pursuit in conjunction with a PID trajectory controller in wheeled mobile robots.

**Linear PID Velocity Controller**

$$\nu(t) = K_{P_{\text{lin}}} \cdot e_{\text{lin}}(t) + K_{I_{\text{lin}}} \cdot \int e_{\text{lin}}(t)\, dt + K_{D_{\text{lin}}} \cdot \frac{de_{\text{lin}}(t)}{dt} \tag{3.2}$$

**Angular PID Velocity Controller**

$$\omega(t) = K_{P_{\text{ang}}} \cdot e_{\text{ang}}(t) + K_{I_{\text{ang}}} \cdot \int e_{\text{ang}}(t)\, dt + K_{D_{\text{ang}}} \cdot \frac{de_{\text{ang}}(t)}{dt} \tag{3.3}$$

A PID controller is a type of feedback control system that governs the behavior of a system based on the difference between the planned setpoint and the actual output. The PID controller is in charge of altering the robot's velocity to minimize the tracking error between the actual location and the planned path in the setting of trajectory tracking. The PID controller consists of three components proportional, integral, and derivative, which work together to provide accurate and steady control.

**Proportional (P) Component:** The proportional component generates a control output proportionate to the difference between the actual location of the robot and the planned route. The stronger the corrective action, the greater the mistake, ensuring a speedy response to deviations from the course.

**The integral (I) component:** collects prior mistakes over time and modifies the control output to eliminate any steady-state faults. This feature ensures that the robot can correctly maintain the specified trajectory.

**Derivative (D) Component:** The derivative component predicts future mistakes by monitoring the error's rate of change. It dampens the robot's reaction to abrupt changes, reducing overshooting and oscillations as shown in Figure 3.7.

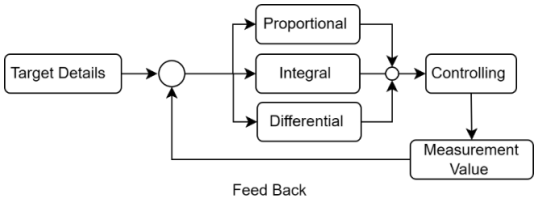



**Figure 3.7** PID loop

Combining the Pure Pursuit algorithm with a PID trajectory controller provides considerable benefits in terms of trajectory following accuracy and stability. The Pure Pursuit algorithm ensures that the robot maintains the target velocity while minimizing mistakes and departures from the course3.8.

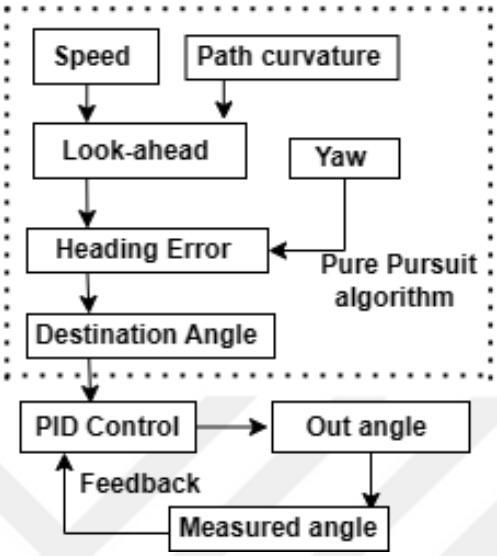



**Figure 3.8** Pure pursuit with PID control

**The Benefits of Pure Pursuit with PID Trajectory Control:**

Pure Pursuit algorithm is a model-based technique for route following in wheeled mobile robots. It works by choosing a point on the intended path, known as the look ahead point, and directing the robot towards it. The look ahead point is dynamically updated based on the position and velocity of the robot, allowing it to follow the path smoothly. The program computes the curvature of the path necessary to reach the look ahead point and changes the steering angle of the robot appropriately. The operation is repeated when the robot reaches the look-ahead point, guaranteeing continuous trajectory monitoring.

The Smooth Trajectory Following by combining the Pure Pursuit algorithm and the PID controller, the robot may follow the intended path with fewer oscillations and smoother trajectory tracking.

The PID controller minimizes error actively, tracking errors between the actual location of the robot and the planned path, resulting in more precise navigation. The PID controller's capacity to adapt to changing conditions and minimize steady-state faults helps to the trajectory tracking system's overall stability and robustness.The Pure Pursuit and PID combo is adaptable to a variety of robotic platforms, giving it a versatile option for trajectory tracking jobs.The Pure Pursuit with PID trajectory controller runs in real-time, allowing the robot to react quickly to changes in the

environment or planned course.

A strong strategy for providing precise and reliable trajectory tracking in wheeled mobile robots is Pure Pursuit with PID trajectory controller. This technique delivers efficient and precise navigation in complicated situations by combining the route following capabilities of Pure Pursuit with the error correction and velocity regulation of the PID controller. The use of these strategies aids in the growth of robotic systems, allowing them to execute a variety of tasks in a variety of applications such as autonomous cars, warehouse automation, and industrial robots. Pure Pursuit with PID is a local planner that takes the global plan from planner like A* or euclid and sends the robot command velocity. It uses the pure pursuit algorithm with PID control to achieve this. There is nothing related to the map to change in Pure Pursuit with PID. However, there are parameters in the launch file, that can change the PID Parameters (Kp, Ki, Kd) for linear and angular velocities, and also the max velocities (max_v and max_w). This can be changed to improve the behavior of the robot, to make it follow the path better, or to increase or decrease the speed. we will explore the different functions and their use in the Pure Pursuit with PID _planner_real.cpp file.

**printPose:** Helper function to print a pose .

**getPose:** Helper function to convert PoseStamped type to Eigen Vector3f type .

**calcErrors**: This function calcultes the linear error and angular error based on current robot pose and nearest goal (based on look ahead). It calculates euclidean distance as linear error, and the angle difference between the current robot direction and goal direction as the angular error.

**saturate:** this function limits a value between a min and max range. If its outside the range, it sets it to the max or min.

**calcDistance:** Calculates euclidean distance between two poses.

**scan_cb:** Lidar callback Not used.

**initialize:** This function initializes the Pure Pursuit with PID planner, it subscribes to odometry and local costmap in ROS, and sets variables to default values (0.0).

**getTargetPoint:** Based on the input global plan, this function finds the point closes to the robot that is higher than the lookahead distance. This point is used as the target to find linear and angular errors for trajectory control.

**setPlan:** this function is called when the global planner sends a new plan. Here the global plan is updated .

**computeVelocityCommands**: This is the function with the main logic, its input is the current robot position, and its output is the control velocity that has to be applied to the robot. This function first finds the look ahead point, uses it as target to compute linear and angular errors, applies PID look on the errors, and returns the required linear and angular velocity.

**isGoalReached:** Helper function to check if the robot has reached the goal. It uses tolerance parameter to check.

**worldToMap:** Helper function to convert world coordinates to map index number as shown in Figure3.9.

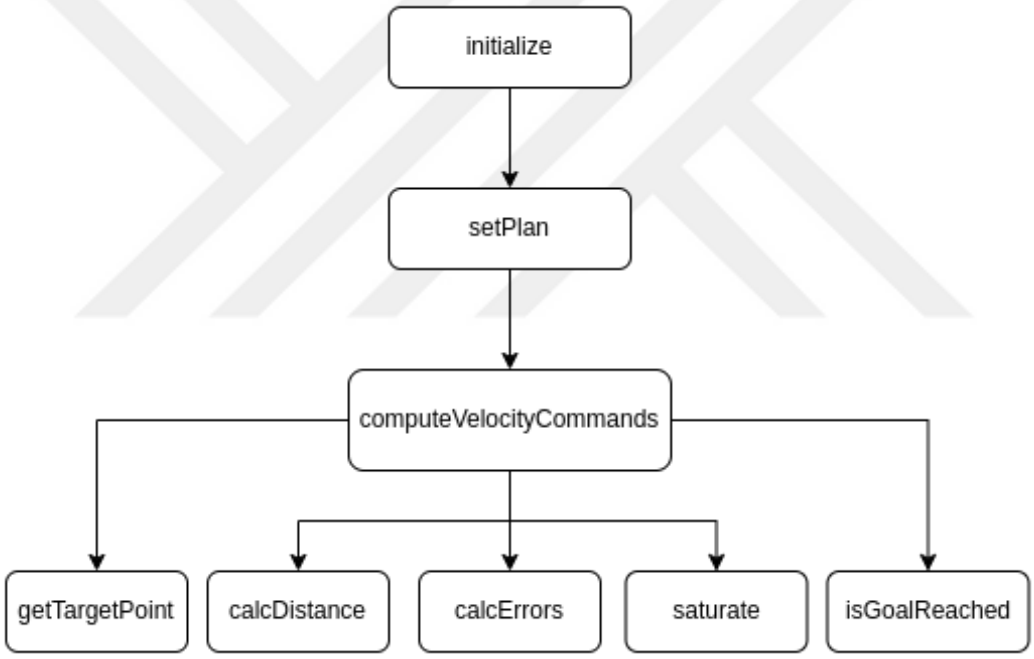



**Figure 3.9** Pure pursuit with PID planner algorithm

## 3.4 Proposed Real Platform Odometry

### 3.4.1 Wheel Tick Odometry vs. Laser Odometry A Comparative Analysis

Wheel tick odometry and laser odometry are two distinct methods used in robotics for estimating a robot's position and movement. While both serve this purpose, they employ fundamentally different approaches, each with its own strengths and weaknesses. In this comparison, we'll explore the key differences between the two methods and discuss why wheel odometry is generally considered more precise.

### 3.4.2 Robot Wheel Odometry

Robot wheel odometry is a fundamental technique used in mobile robotics to estimate a robot's position and orientation based on the movement of its wheels. It serves as a critical component for navigation systems, allowing robots to determine their relative position in a given environment. This thesis will explore the concept of wheel odometry, how it is collected, and the types of messages associated with it. Wheel tick odometry relies on wheel encoders, which generate pulses for each rotation of the wheel. These pulses are used to estimate the distance traveled by the robot.It assumes that the wheels do not slip, providing a reliable measure of linear displacement.Wheel tick odometry is commonly used in differential drive robots with simple wheel configurations.Wheel encoders are relatively inexpensive, making this method accessible for a wide range of robotic platforms. The principle behind wheel tick odometry is straightforward, making it easy to implement and understand.If the wheels experience slippage, for example on a non-ideal terrain, the accuracy of wheel tick odometry can be compromised.Over time, small inaccuracies in the wheel encoder measurements can accumulate, leading to a drift in the estimated position.

#### 3.4.2.1 What is Wheel Odometry?

Wheel odometry is a method of estimating a robot's pose (position and orientation) by measuring the rotation of its wheels. Odometry refers to the utilization of data obtained from motion sensors to calculate the variation in position during a specific period. Legged or wheeled robots utilize it in the field of robotics to determine their position with relation to an initial location. This method is susceptible to mistakes caused by the integration of velocity measurements over time in order to obtain position estimates. Efficient and precise data collection, calibration of instruments, and processing are essential prerequisites for the optimal use of odometry in most scenarios. It relies on the assumption that the wheels do not slip, providing a straightforward way to calculate the robot's movement in a 2D plane. The technique is commonly used in differential drive robots, where two wheels are the primary means of locomotion.

#### 3.4.2.2 How Wheel Odometry Works

Wheel odometry operates on the principle of calculating the change in position based on the wheel's rotation. By measuring the angular displacement of each wheel, it's possible to infer how far the robot has moved and in which direction. This information is then used to update the robot's estimated pose. Wheel odometry

relies on wheel encoders, which are sensors attached to the robot's wheels to measure their rotation. These encoders generate pulses as the wheel turns, allowing the system to track the distance traveled. The number of pulses and the wheel's circumference are used to calculate the linear displacement.

The Incremental Encoders provide information about the change in position, but not the absolute position. They are widely used in robotics due to their simplicity and cost-effectiveness.Wheel odometry data is typically transmitted as nav_msgs/Odometry messages in ROS (Robot Operating System). These messages contain essential information about the robot's pose and velocity in free space. The message structure includes:

- Header header: Contains timestamped data in a specific coordinate frame.
- string child_frame_id: Specifies the coordinate frame associated with the pose and twist information.
- geometry_msgs/PoseWithCovariance pose: Contains position and orientation estimates along with covariance data representing uncertainty.
- geometry_msgs/TwistWithCovariance twist: Contains linear and angular velocity estimates along with covariance data.

#### 3.4.2.3 Transformations

Wheel odometry data is transformed into the robot's base link frame. This transformation is crucial for accurately integrating odometry with other sensor data and ensuring that the information is provided in a consistent reference frame.

#### 3.4.2.4 Fusion with Other Sensors

To improve accuracy, wheel odometry data is often fused with information from other sensors like IMUs (Inertial Measurement Units) and LIDAR (Light Detection and Ranging). IMUs provide data on the robot's orientation and angular velocity, while LIDAR offers precise measurements of the environment's structure. Fusion algorithms, such as those provided by the robot_localization package, are used to combine data from multiple sources.Therefore ,wheel odometry is a foundational technique in mobile robotics, allowing robots to estimate their position and orientation based on wheel movement. By using wheel encoders to measure rotation, odometry data is collected and transmitted as nav_msgs/Odometry messages in ROS. This information is crucial for tasks like navigation, localization,

and mapping. Additionally, integrating wheel odometry with data from other sensors enhances accuracy and robustness in complex environments.

wheel odometry serves as a cornerstone in robotics, enabling robots to navigate and interact with their surroundings effectively. Its integration with other sensor data and fusion algorithms represents a significant step towards creating autonomous and capable robotic systems.

### 3.4.3 Laser Odometry

Laser odometry, on the other hand, uses data from a LIDAR (Light Detection and Ranging) sensor. This sensor scans the environment, creating a detailed map of the surroundings.Sophisticated algorithms process the LIDAR data to detect features and track their movement. This allows for accurate estimation of the robot's position. Laser odometry can provide highly accurate estimates of a robot's position, especially in environments with complex structures or objects. It is less susceptible to issues like wheel slippage, making it more reliable in diverse environments.LIDAR sensors tend to be more expensive than wheel encoders, which can increase the overall cost of the robotic system. Implementing laser odometry requires more sophisticated hardware and software, which can be challenging for beginners.

#### 3.4.3.1 rf2o_laser_odometry

Rf2o_laser_odometry is a ROS package used for laser odometry. Laser odometry is a technique used in mobile robotics to estimate the position and orientation of a robot by processing laser scan data. rf2o stands for "Range Flow Odometry". It's a method that fuses range information from laser scans over time to estimate the robot's motion.The package employs a technique called Range Flow, which computes dense optical flow from two consecutive laser scans. This flow information is then used to estimate the robot's motion.The primary input to rf2o_laser_odometry is laser scan data. This data is usually obtained from a laser scanner mounted on the robot.

The raw laser scan data is preprocessed to remove noise and outliers,this might involve filtering and cleaning the data.

**Range Flow Estimation**

The main algorithmic step is to compute the range flow. This is done by comparing consecutive laser scans to find correspondences between points. It essentially

estimates how each point in the laser scan moves from one scan to the next. The odometry estimation using the computed range flow, the package estimates the robot's motion. This includes both translational and rotational components. Over time, the odometry estimates are integrated over time to provide a continuous motion estimate. The output of rf2o_laser_odometry is typically a pose estimate, which includes the position (x, y, z) and orientation (roll, pitch, yaw) of the robot at each time step as mentioned in Figure 3.10.

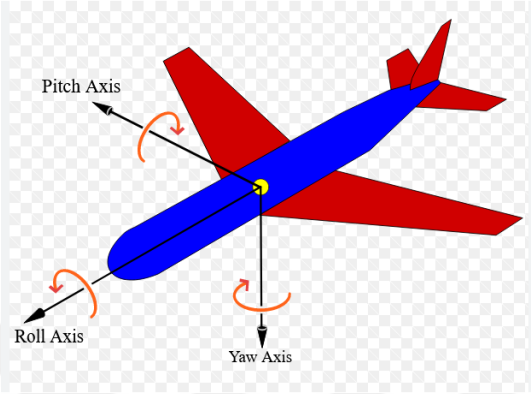



**Figure 3.10** Orientation (roll, pitch, yaw)

because its a wheeled mobile robot, we are going to use yaw angles to change the robot's heading toward the waypoint, Since it's based on range information, it's less affected by dynamic objects in the environment compared to methods based on visual odometry the laser reading as shown in Figure 3.11.

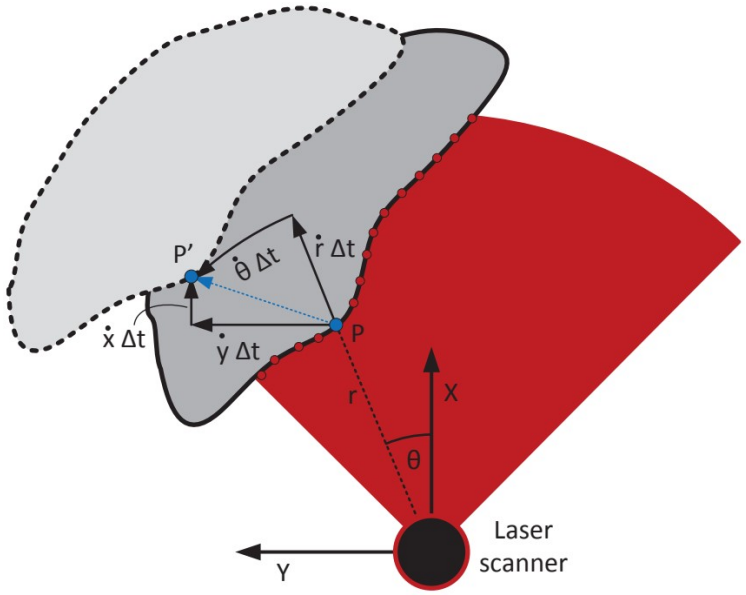



**Figure 3.11** Rf2O laser odometry

Top-down depiction of a laser scan that intersects with a specific object. Point $P$

undergoes a displacement relative to the scanner, resulting in its new position $P_0$ after a time period $\Delta t$ [49].

Laser scanners work well in indoor environments with controlled lighting conditions.`rf2o_laser_odometry` is commonly used in mobile robotics for tasks like simultaneous localization and mapping (SLAM) and navigation. Depending on the specific hardware setup and environment, there might be parameters that need to be tuned for optimal performance as shown in rf20 algorithm below .

**Algorithm 7** RF2O - Robust Feature-Based 2D Laser Odometry

**Require:** Laser scan data $L$, odometry data $O$
  Initialize odometry-related variables and feature-based map $M$
  Extract features from the initial scan $L_0$
  **for** each received laser scan $L_t$ and odometry data $O_t$ **do**
    Extract features from the current scan $L_t$
    Match features between $L_t$ and $L_{t-1}$
    Estimate the relative pose $\Delta x_{t,t-1}$ based on feature correspondences
    Update the odometry pose $x_t$ using $\Delta x_{t,t-1}$ and $O_t$
    Refine the odometry pose using optimization techniques
    Update the feature-based map $M$ using $L_t$ and $x_t$
  **end for**

### Integration with ROS

Since it's a ROS package, rf2o_laser_odometry can be easily integrated into larger robotic systems that use ROS for communication and coordination. Remember, the specifics of how rf2o_laser_odometry works might vary based on the version and any customization made by the user or the development team. Always refer to the official documentation and source code for the most accurate and up-to-date information. Wheel odometry is often considered more precise than laser odometry in specific scenarios. This is primarily due to its direct measurement of wheel rotations, which can be very accurate when wheels do not slip. Additionally, wheel odometry can provide continuous and real-time updates on a robot's position, making it suitable for tasks that require rapid feedback. However, in environments with intricate structures or a high likelihood of wheel slippage, laser odometry tends to outperform wheel odometry. The detailed point cloud data provided by lidar allows for a more robust estimation of the robot's position, even in complex surroundings. The choice between wheel tick odometry and laser odometry depends on the specific requirements and constraints of the robotic application. While wheel

odometry is generally more precise under ideal conditions, laser odometry excels in complex and dynamic environments. A combination of both methods, known as sensor fusion, is often employed to leverage the strengths of each approach and mitigate their respective weaknesses.

**Proposed design Software requirements**

we should summarize the whole requirements to our proposed design as below in Table 3.1

| **Proposed Paradigm** | |
|---|---|
| **Criteria** | **Description** |
| Operating system | Ubntu 20 with Ros1 Noetic |
| Global Planner | A* Algorthim |
| Local Planner | Pure Pursuit Algorthim |
| Map | Low-Resolution Grid of markers , rf20 |
| Controller 1 | Trajectory Proportional, Integral Derivative Controller. Linear PID. Angular PID. |
| Controller 2 | Velocity Controller proportional controller. |

**Table 3.1** The proposed options

### 3.4.4 Elements of The Robotics System

Robots exhibit varying designs based on their ability to fulfil certain objectives and satisfy particular specifications. Certain robots prioritise material cost above quality, while others provide a balance between the two. The robot's geometric configuration influences its navigation by effectively distributing weight on the wheels and ensuring smooth passage around turns, barriers, and obstacles without becoming stuck.

#### 3.4.4.1 Simulation Sensor

```
<sensor type="ray" name="lds_lfcd_sensor">
    <update_rate>5</update_rate>
    <ray>
        <scan>
            <horizontal>
                <samples>360</samples>
                <resolution>1</resolution>
                <min_angle>0.0</min_angle>
```

```
                <max_angle>6.28319</max_angle>
            </horizontal>
        </scan>
        <range>
            <min>0.120</min>
            <max>3.5</max>
            <resolution>0.015</resolution>
        </range>
        <noise>
            <type>gaussian</type>
            <mean>0.0</mean>
            <stddev>0.01</stddev>
        </noise>
    </ray>
</sensor>
```

**Update_rate:** How many laser scan messages should be generated in 1 second .

**Horizontal samples:** How may points should be present in one scan .

**Horizontal resolution:** Angle between two points .

**Horizontal min angle, max angle:** Starting and ending angle of scan.

**Range min, max:** Minimum and maximum distance measured by laser .

**Range resolution:** Minimum change in distance measurement. (If laser can give 35cm or 36cm, then resolution is 1cm) .

**Noise:** How much noise should be added to measurement

Noise is added using gaussian sampling. We can set mean and deviation parameters.

That means the laser can create 360 points in 360 degrees which is impossible in hardware representation as shown in Figure 3.12.

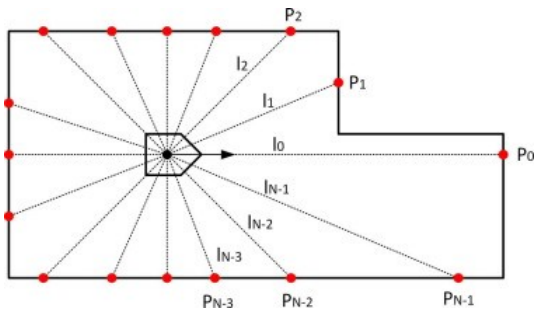



**Figure 3.12** Simulation sensor points distribution

#### 3.4.4.2 Real Platform Sensor

In simulation, we can change the parameters as we require. The gazebo simulator will read this and create the required laser sensor. But , In hardware, the parameters are fixed based on the model type. It is provided by manufacturers in the data sheet.

**Update_rate:** How many laser scan messages should be generated in 1 second .

**Horizontal samples:** How may points should be present in one scan .

**Horizontal resolution:** Angle between two points .

**Horizontal min angle, max angle:** Starting and ending angle of scan .

**Range min, max:** Minimum and maximum distance measured by laser .

**Range resolution:** Minimum change in distance measurement. (If laser can give 35cm or 36cm, then the resolution is 1cm) .

Noise parameters are not used in hardware. This is because the noise is generally non-gaussian, and cannot be modeled easily.

**Hokuyo URG laser sensor:** In the realm of robotics and automation, is a frequently used and highly rated sensor. The Hokuyo URG sensor has revolutionized perception systems with its enhanced capabilities and dependability, enabling precise and efficient detection and mapping of the surroundings. In this statement, we will look at the many features of the Hokuyo URG laser sensor, such as its operation, uses, advantages, and future possibilities.

The URG Hokuyo laser sensor is a range sensor that measures distances and detects objects in its surroundings using laser technology. According to Hokuyo Automatic Co., Ltd., a major japanese sensor producer, the URG sensor series has acquired widespread acceptance and use throughout sectors and research disciplines Figure

3.13 .

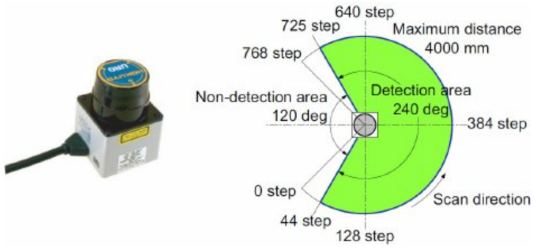



**Figure 3.13** URG hokuyo LIDAR

We may find applications for the Hokuyo URG sensor in several industries, including robotics, autonomous driving, industrial automation, security systems, and SLAM research. We will examine how the sensor is applied in each of these fields, going over practical examples and the benefits it provides for boosting perception and judgment. Researchers and practitioners choose the Hokuyo URG sensor because it has several benefits. These advantages include its small size, excellent measurement precision, wide operating range, dependability, low power need, and interoperability with various robotic platforms. We can comprehend the appeal of the sensor in the robotics and automation sector by looking at these benefits. Understanding the Hokuyo URG sensor's integration and programming elements is crucial for efficient use. This part will go into the many programming interfaces, including ROS (Robot Operating System), and offer instructions on how to connect to the sensor, get data, and process it for additional analysis and decision-making. Even though the Hokuyo URG sensor has several advantages, it's necessary to be aware of its drawbacks and restrictions while using it. Limitations in outdoor settings, difficulty with reflecting surfaces, occlusion concerns, and difficulties with data processing and interpretation are a few examples of these. Researchers and practitioners may make wise choices and create suitable solutions by being aware of these limits.

In this part, we'll contrast the Hokuyo URG laser sensor with some of the other well-liked ones now on the market. We may learn more about the special qualities that distinguish the Hokuyo URG sensor from its rivals by looking at aspects like performance, price, adaptability, and applicability for particular applications. The Hokuyo URG sensor is no exception to the ongoing evolution in the field of laser sensors. In this part, we'll look at some recent improvements in sensor technology, such as improvements in range, resolution, and scanning speed. Additionally, we'll talk about potential future developments like new apps and better integration.

Hokuyo Sensor requires to be identified in the system, so, the URG node has

been installed on the raspberry PI3 side to get the sensor reading, and a physical connection is to be provided with a 5VDC power supply and connecting the USB data cable with raspberry PI3, with topic name /urg node we can test the received data from Hokuyo URG on raspberry PI3 side.

#### 3.4.4.3 Motor

Powerful and adaptable, the 12V DC RS-37-555 Side Shaft Geared Motor - 100 RPM is made for a variety of industrial systems, including robotics and automation. This motor provides dependable and accurate performance for a variety of jobs thanks to its high efficiency, small size, and sturdy construction. The RS-37-555 motor is compatible with common battery systems and power sources thanks to its 12V direct current (DC) power supply, which is a key component of the device. This voltage range makes it simple to integrate into different electrical configurations and streamlines power management. The motor's design includes a geared arrangement that offers more torque and slower output speed. This motor's gear ratio is designed to reach a rotating speed of 100 revolutions per minute (rpm), making it suited for uses that call for a high output of torque at a modest pace. The motor is more effective and can easily manage big loads thanks to the geared system. The RS-37-555 motor's side shaft arrangement provides more design and installation options. The side shaft is perfect for driving systems in robotics, conveyor belts, and many automation activities because it enables direct attachment to wheels, pulleys, or other mechanical components.

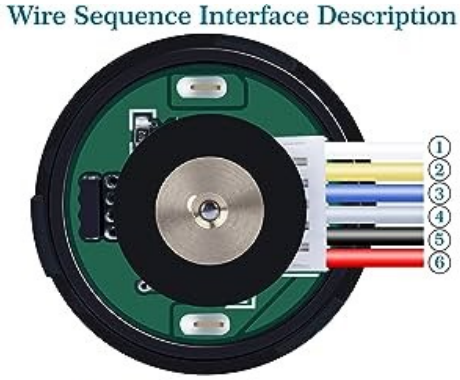
Aside from having great durability, the RS-37-555 motor also has a long service life thanks to its premium building materials and careful engineering. Usually constructed of tough metal or resilient plastic, the motor's exterior housing protects it from the weather and ensures dependable performance in a variety of settings. Additionally, the motor has ball bearings that lessen wear and friction during use, resulting in a quiet and smooth performance. The motor's extended lifespan is a result of the usage of ball bearings, which makes it an affordable option for industrial applications and long-term projects. The interface between a motor controller and an electronic speed regulator for the RS-37-555 motor is simple. This feature enables smooth and accurate motion control in robotic systems and other automation activities by permitting exact control of the motor's rotational speed and direction. The motor's small size and lightweight construction make it simple to integrate into mobile platforms and systems with limited space. In battery-powered applications, its low power consumption also helps with overall energy economy and increased battery life. The 12V DC RS-37-555 Side Shaft Geared Motor - 100 RPM is a flexible and dependable motor appropriate for a

variety of applications in industrial systems, robotics, and automation. This motor offers a practical and affordable option for numerous motion control jobs because of its high torque, accurate speed control, and robust construction. The RS-37-555 motor is a crucial component in contemporary engineering and design projects because it offers dependable performance whether employed in robotics, conveyor systems, or other automation applications. increased battery life Figure 3.14.

**Figure 3.14** Wheel DC motor

#### 3.4.4.4 DC Morotr_Encoder

A key component that provides accurate motion control and feedback in a variety of robotic and automation applications is the Motor_Encoder coupled with the 12V DC RS-37-555 Side Shaft Geared Motor @ 100 rpm as in Figure3.15.




| | |
|---|---|
| White | Encoder B Output |
| Yellow | Encoder A Output |
| Blue | Encoder VCC (3.3V to 24V) |
| Gray | Encoder GND |
| Black | Motor Power (Connects to the Other Motor Terminal) |
| Red | Motor Power (Connects to One Motor Terminal) |

**Figure 3.15** Wheel encoder

The encoder improves the overall performance and dependability of the motor-driven system by offering precise position and speed information. When combined with the RS-37-555 motor, the Motor_Encoder is a high-resolution rotary

with an encoder at a gearbox 1:12 ratio that can produce 6000 ticks for raising and lowering throughout a full revolution and 30,000 ticks per meter find Figure 3.16.

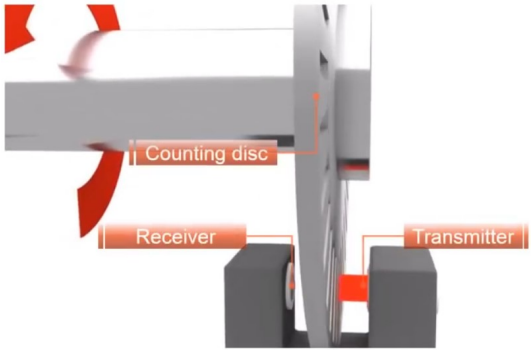



**Figure 3.16** DC Motor wheel encoder

It is excellent for applications that demand precise motion control, such as robotics, CNC machines, and conveyor systems, due to its high resolution, which enables fine-grained position detection and rotational control of the motor. The encoder and motor work together flawlessly, with the encoder's shaft connected to the RS-37-555 motor's side shaft. The encoder produces electrical ticks or pulses as the motor turns that represent minute changes in the location of the motor. The control system receives the encoder's output, which enables real-time feedback and alterations to the speed and location of the motor. The arduino UNO and encoder have an easy-to-follow connection. The arduino UNO card supplies a 5V DC power supply to the encoder, ensuring a steady and continuous power source for dependable operation. Ticks A and B, the encoder's output signals, are then connected, respectively, to pins 2 and 3 on the arduino UNO.

The incremental encoder's output signals are in quadrature format, which means that the A and B signals are 90 degrees phase-shifted. This quadrature encoding allows the arduino UNO to identify both the direction and amount of the motor's rotation, improving the precision and responsiveness of the control system by checking whether A channel is leading or lagging B channel as shown in Figure 3.17.

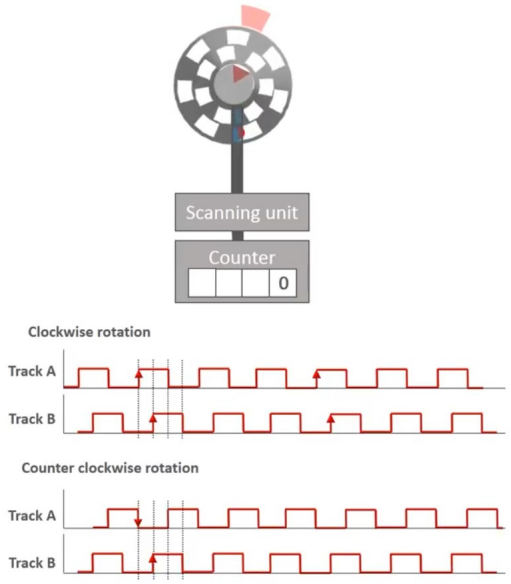



**Figure 3.17** Forward and reverse rotation through A and B channels

As the microcontroller in this configuration, the arduino UNO continually monitors the encoder output and processes the ticks to compute the motor's speed and position. The arduino UNO can create control signals to modify the motor's speed and position by comparing the encoder's feedback with the required setpoints, essentially establishing closed-loop control of the motor. This closed-loop control method has various advantages, including accurate position monitoring, enhanced stability, and the capacity to adapt to disturbances or external variables impacting the functioning of the motor. It guarantees that the motor retains its intended position and speed even when subjected to external pressures or environmental changes. The Motor_Encoder's integration with the 12V DC RS-37-555 Side Shaft Geared Motor @ 100 rpm and the arduino UNO allows for complex motion control capabilities. Because of the encoder's high resolution and precision, fine-tuning the motor's motions is possible, making it suited for applications requiring accurate placement and smooth motion.

#### 3.4.4.5 Arduino UNO

The arduino Uno is a famous open-source microcontroller board that is frequently used in academic and DIY projects. It has become one of the most accessible and adaptable platforms for testing and experimenting with electronics and programming, thanks to the arduino community. The Atmel ATmega328P microcontroller is at the core of the Arduino Uno, providing the processing power and memory necessary to run arduino sketches (programs). The board also has digital input/output connectors, analog input pins, built-in components such as

LED indicators, and a USB-to-serial converter for simple contact with a computer. One of the main advantages of the arduino Uno is its simplicity and ease of use, which makes it a good starting point for those new to electronics and programming. The arduino integrated development environment (IDE) provides an easy-to-use environment for composing, developing, and uploading sketches to the board. The code is developed in arduino, a simplified version of C++. The arduino Uno is very expandable, allowing users to enhance its capabilities by adding numerous shields and modules. Shields are extra boards that may be placed on top of the arduino Uno to give features like as Ethernet connectivity, wireless communication (Wi-Fi, Bluetooth), motor control, and others. Because of its versatility, the board may be easily customized for individual projects and applications.

The arduino Uno is frequently used in academic contexts for teaching and studying electronics, programming, and robotics in educational institutions and research facilities. It's a fantastic tool for exposing kids to the world of embedded systems and allowing them to put their creative ideas into action through practical projects.

One of the most useful advantages of utilizing arduino Uno in academic settings is the extensive community support and resources available. There are several online courses, forums, and guidelines available to assist students and researchers in getting started and overcoming obstacles in their work. Furthermore, the open-source nature of arduino fosters knowledge exchange and cooperation among users all around the world. Researchers and students frequently utilize the arduino Uno for quick prototyping and proof-of-concept development. Its cheap cost and simplicity furthermore, the availability of many sensors, actuators, and modules that are compatible with the arduino Uno expands its capabilities and allows researchers to gather data and interact with the real world in their investigations. The arduino Uno's influence in academia extends beyond academic institutions. Many new initiatives and businesses have sprung up as a result of academic study that began with the arduino Uno as a prototype tool. Its importance as a stepping stone in the innovation process cannot be overestimated. Overall, the arduino Uno has changed the way students, academics, and enthusiasts think about electronics and programming. It has democratized access to embedded systems and created a forum for ideas to become reality. As technology advances, the arduino Uno remains an important and relevant tool, enabling individuals to create, explore, and contribute to society as shown in Figure 3.18.

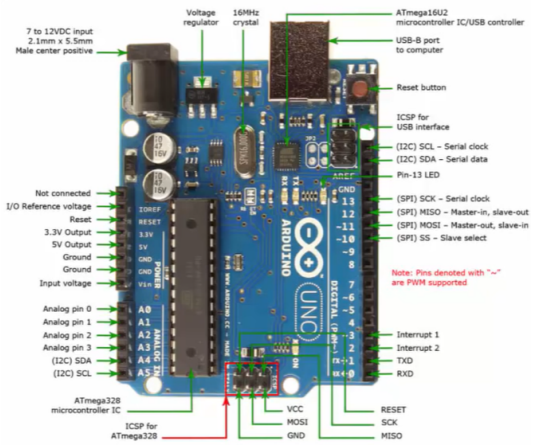



**Figure 3.18** Arduino UNO

### 3.4.4.6 Motor Drive VNH 5019 Arduino uno

The arduino Uno with VNH5019 is a strong combo that expands the arduino board's capabilities, allowing it to drive high-current loads such as motors and actuators. The VNH5019 is a twin high-current motor driver module that enables the arduino Uno to operate two motors separately while also offering bidirectional control and current sensing. The VNH5019 motor driver can handle high currents of up to 12A per channel and voltages of up to 24V. As a result, it is perfect for driving high-power motors used in robotics, electric cars, and industrial automation. Because of its tiny and user-friendly design, the VNH5019's integration with the arduino Uno is straightforward. The module connects to the arduino Uno through standard digital output pins, enabling simple motor programming and control.

The VNH5019 motor driver has a variety of safeguards to ensure the system's safety. It includes overcurrent protection, thermal shutdown, and undervoltage lockout to help prevent motor and driver damage in the case of failure or underutilized operating conditions. The arduino Uno becomes a flexible platform for constructing numerous robotics and automation applications with the VNH5019 motor driver. Its capacity to drive high-power motors allows for the development of durable and nimble robotic systems capable of performing complicated tasks in difficult settings. The combination of arduino Uno and VNH5019 is commonly utilized in academic contexts for teaching and research in robotics, automation, and control systems. Students and researchers may easily investigate motor control ideas, investigate motion planning techniques, and create sophisticated robotics applications.

The availability of detailed documentation and community assistance is one of the primary benefits of utilizing arduino Uno with VNH5019 in academic projects. The arduino community and different online forums provide a multitude of materials,

tutorials, and sample projects to help students and researchers with their projects. Furthermore, the open-source nature of arduino and the VNH5019 motor driver encourages cooperation and information exchange among users all over the world. This allows for the sharing of ideas and best practices, which enriches the learning and research experience even more. The arduino Uno with VNH5019 is also a fantastic hands-on learning platform, allowing students to acquire practical experience in motor control, power electronics, and embedded systems. Working with real-world hardware and constructing robotic systems improves students' knowledge of theoretical ideas and prepares them for future robotics and automation difficulties. arduino Uno with VNH5019 can be used in research in a variety of disciplines, including driverless cars, industrial automation, and smart home systems. Because of its scalability and versatility, it is ideal for developing and testing new ideas before moving on to more specialized and production-grade gear. The combination of arduino Uno with VNH5019 is also cost-effective, making it accessible to educational institutions and research laboratories with limited budgets. This affordability allows for more students and researchers to have access to state-of-the-art motor control technologies and robotics development 3.19.

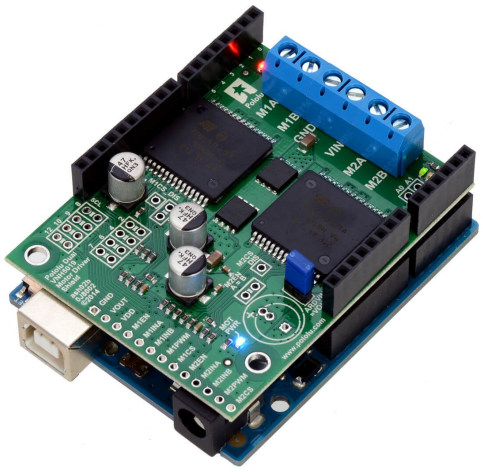



**Figure 3.19** Motor drive VNH5019

The arduino Uno with VNH5019 is a strong and adaptable platform that offers up new possibilities in motor control and robotics. Its simplicity of use, strong community support, and interoperability with a wide range of sensors and actuators make it a good choice for academic robotics and automation projects and research. Students and researchers may use this combination to explore new frontiers in motor control, motion planning, and robotics, contributing to the growth of the subject. The VNH5019 is an integrated circuit that controls DC motors. It is intended to power high-current motors and has several functions for adjusting motor speed and direction.

These are some of the commonly accessible functions in a VNH5019 library, however, the precise collection of functions and their names may vary based on the individual library implementation. It is critical to consult the library documentation or header files for exact function names and use. Furthermore, while dealing with motor drivers and high-current systems, make sure you grasp the specs and safety factors to avoid harming the motors and driver. The VNH5019 is an integrated circuit that controls DC motors. It is intended to power high-current motors and has a number of functions for adjusting motor speed and direction. The VNH5019 library features allow you to communicate with the motor driver and operate the associated DC motors. Some of the important functions that are commonly accessible in a VNH5019 library are as follows:

**VNH5019():** This is the constructor function for the VNH5019 library. It establishes a connection with the motor driver and configures it for usage.

**brake():** The brake() function applies a brake to the motor, bringing it to a stop quickly. It achieves this by shorting the motor terminals, effectively creating a dynamic brake.

**stop():** This function stops the motor by setting its speed to zero. Unlike the brake() function, the motor gradually stops due to friction and inertia.

**getSpeed():** This function allows you to retrieve the current speed of the motor. It can be useful for feedback control or monitoring the motor's behavior.

**getDirection():** The getDirection() function is used to determine the current rotation direction of the motor. It returns a value indicating whether the motor is rotating forward or backward.

**setBrakeMode(mode):** The VNH5019 driver offers different brake modes, such as coasting or dynamic braking. The setBrakeMode() function is used to configure the desired brake mode. For example, you might set it to BRAKE_MODE_COAST for coasting or BRAKE_MODE_BRAKE for dynamic braking.

**setCurrentLimit(current_limit):** This function is used to set the maximum current allowed to flow through the motor driver. It can help protect the motor and driver from excessive currents that could damage them.

**enable() and disable():** These functions are used to enable or disable the motor driver, respectively. When disabled, the motor will not receive any power, effectively stopping it.

**isFault():** The VNH5019 driver can detect certain faults, such as over-temperature or over-current conditions. The isFault() function allows you to check if any fault has occurred.

**VNH5019():** This is the constructor function used to initialize the VNH5019 library. It sets up the communication with the motor driver and prepares it for use.

**setSpeed(speed):** This function controls the speed of the attached motor. The speed parameter represents the intended speed and can be either positive or negative to influence the motor's direction. A positive number represents forward rotation, whereas a negative value represents backward rotation. The function changes the voltage applied to the motor to obtain the specified speed.

#### 3.4.4.7 RASPBERRY-PI3

Raspberry Pi 3 Model B is a versatile and capable single-board computer that has gained significant popularity across various domains, including education, hobbyist projects, and industrial applications. It is the third iteration of the raspberry Pi series and comes with impressive capabilities, making it a sought-after choice for diverse applications.

The powerful 1.2GHz quad-core ARM Cortex-A53 CPU at the core of the raspberry Pi 3 Model B offers substantial processing power for a wide range of computing tasks. This improved CPU enables the raspberry Pi 3 to handle complex applications and multitask effectively, although it does have certain limitations.

Featuring 1GB of RAM, the raspberry Pi 3 Model B provides sufficient memory capacity to ensure the smooth and efficient execution of software programs. This increased RAM capacity compared to earlier models enhances the performance of memory-intensive tasks and improves system responsiveness.

One of the standout features of the raspberry Pi 3 Model B is its built-in Wi-Fi and Bluetooth Low Energy (BLE) capabilities. With integrated Wi-Fi and BLE functionalities, the raspberry Pi 3 can connect to wireless networks, access the internet, communicate with other devices, and transmit data wirelessly. This wireless connectivity adds flexibility and convenience to projects, making the raspberry Pi 3 an ideal choice for IoT applications and remote monitoring systems.

The raspberry Pi 3 Model B additionally includes 40 General Purpose Input-Output (GPIO) connectors for connecting a variety of sensors, actuators, and other peripheral devices. The raspberry Pi's GPIO ports allow it to interface with the real

environment, making it perfect for robotics, home automation, and sensor-based projects. One of the raspberry Pi 3 Model B's main advantages is its small size and low power usage. The single-board computer's tiny form factor allows it to be incorporated into space-constrained projects, and its energy-efficient design assures minimal power usage, making it appropriate for battery-powered applications and distant deployments.Raspbian, a Debian-based Linux distribution optimized for the raspberry Pi, is one of the operating systems supported by the raspberry Pi 3 Model B. This feature-rich and user-friendly operating system provides a familiar environment for newcomers and enables easy program creation and deployment.The raspberry Pi 3 Model B's flexibility extends to its support for a variety of programming languages, including Python, C++, Java, and others. This extensive language support allows developers and hobbyists to construct a wide range of apps and experiment with various programming paradigms. The raspberry Pi 3 Model B has become a popular teaching tool due to its low cost and ease of use. It is frequently used in schools and institutions to teach ideas such as programming, electronics, and computer science. Its hands-on learning method helps students to get practical experience and problem-solving abilities.

Aside from education, the raspberry Pi 3 Model B has found use in a variety of industries, including home automation, media centers, gaming consoles, weather stations, and more. Its versatility, performance, and low cost make it appealing to enthusiasts, manufacturers, and professionals alike. The raspberry Pi 3 Model B is an incredible single-board computer that has transformed the field of embedded computing. Its strong processor, enough memory, wireless connection, and GPIO capabilities make it an adaptable platform for a variety of applications. Whether utilized for educational reasons, hobbyist projects, or industrial applications, the raspberry Pi 3 Model B continues to inspire innovation and creativity in the world of technology as shown in Figure 3.20.

**Figure 3.20** Raspberry-pi3

To create communication between an arduino and a raspberry Pi, it is necessary to provide a translation mechanism for commands and feedback via the serial ports. This may be achieved by utilizing the ROSserial node, which is responsible for creating a connection and is considered one of the most effective approaches for this purpose.

#### 3.4.4.8 Rosserial Node

ROSserial is a Robotic Communication Bridge that connects arduino to ROS.In the complex world of robotics, communication between many components is critical for efficient and coordinated activities. ROSserial acts as a connection between the power of the Robot Operating System (ROS) and the variety of arduino microcontrollers, allowing for the integration of hardware components such as motor drives and encoders into a coherent robotic system. This scholarly investigation goes into the complexities of ROSserial, revealing its mechanics, relevance, uses, and effect on the field of robotics.ROSserial: ROSserial is an interface that allows bidirectional communication between ROS and arduino-based systems. This collaboration enables the combination of ROS's powerful computing skills with the precise control and real-time sensing capabilities provided by arduino microcontrollers.

**Mechanics of Communication:**

ROSserial facilitates communication since it is simple and effective. ROSserial uses the ROS Serial protocol, which provides the framework for sending messages through serial connections. These messages are written in accordance with the ROS message definitions. On the arduino side, a ROSserial client analyses these messages and converts them into actions or sensor data. This mutual contact creates a unified environment in which data and commands flow effortlessly between ROS and arduino. Communication Mechanics is Important in Robotic Applications, the combination of ROSserial and arduino is extremely important in a variety of robotic applications. One noteworthy use is motor control. In the case of motor drives controlled by arduino, such as the VNH 5019, ROSserial serves as a channel for ROS to steer and handle these motors. This integration is useful in mobile robots, drones, and robotic arms, improving their manoeuvrability and reactivity.

In improving sensing and feedback the accurate sensory information is essential for educated decision-making in robotics. Encoders are essential for delivering input on the motion and location of robotic components. ROSserial integrates encoders with ROS, allowing for real-time monitoring and control of the robot's motions.

This paves the way for complex algorithms to use this accurate data for tasks like as localization, path planning, and obstacle avoidance.The introduction of ROSserial has resulted in a paradigm change in the development of robotic systems. It makes it easier for developers to create complicated robotic behaviours by allowing them to leverage the capabilities of both ROS and arduino. Furthermore, it speeds up prototyping by combining the reliability of ROS's libraries with the quick prototyping capabilities of arduino. This collaboration speeds up the conversion of creative concepts into practical robots.

While ROSserial has transformed robotics, issues remain. Due to the inherent differences between ROS and arduino, issues like latency, synchronization, and compatibility may emerge. Future developments might concentrate on optimizing these features, raising the performance of robotic systems even higher. Furthermore, integrating ROSserial with upcoming hardware platforms might broaden its possibilities. The ROSserial develops as a conduit that connects ROS's digital domain with the physical manifestation of arduino-based electronics. This interface orchestrates a symphony of data and commands, allowing robots to do astonishing feats. The seamless integration of motor drives, encoders, and other hardware components into the ROS environment highlights the strength of software-hardware collaboration. ROSserial, as the field of robotics grows, serves as a testimony to the limitless possibilities that emerge when technology converges.

### 3.4.5 The General Proposed Design will be as below

In the case of the beginning, you'll need to gather the necessary power sources: 12V for the motors and 5V for the raspberry pi3, arduino UNO, and Hukoyo lidar. Then, you'll need to connect all of the electronic components to the raspberry pi3 via its USB serial connector as shown in Figure 3.21.

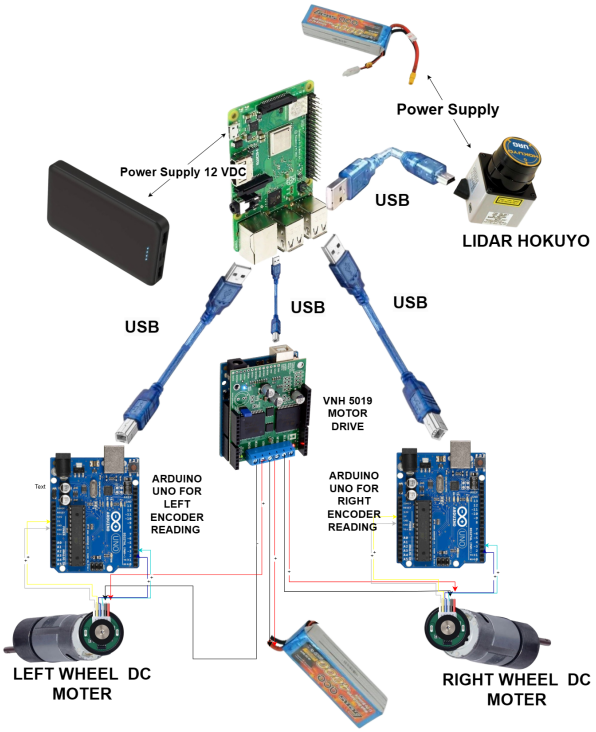



**Figure 3.21** Robot general electrical design

### 3.4.6 Robot Body Hardware Design

The robot machinery design will comprise the distance between the wheels, the diameter of the wheels, and the height of the LiDAR as shown in Figure 3.22. The geometric shape of the robot allows it to move better due to the size of the robot .

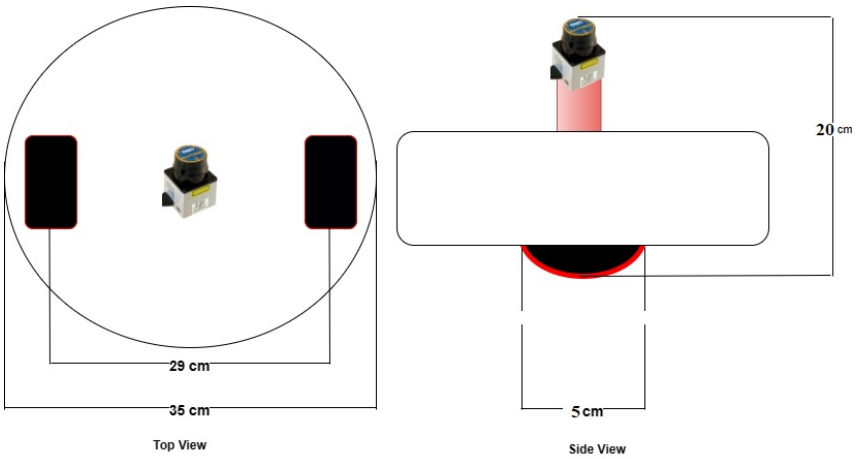



**Figure 3.22** Robot hardware body

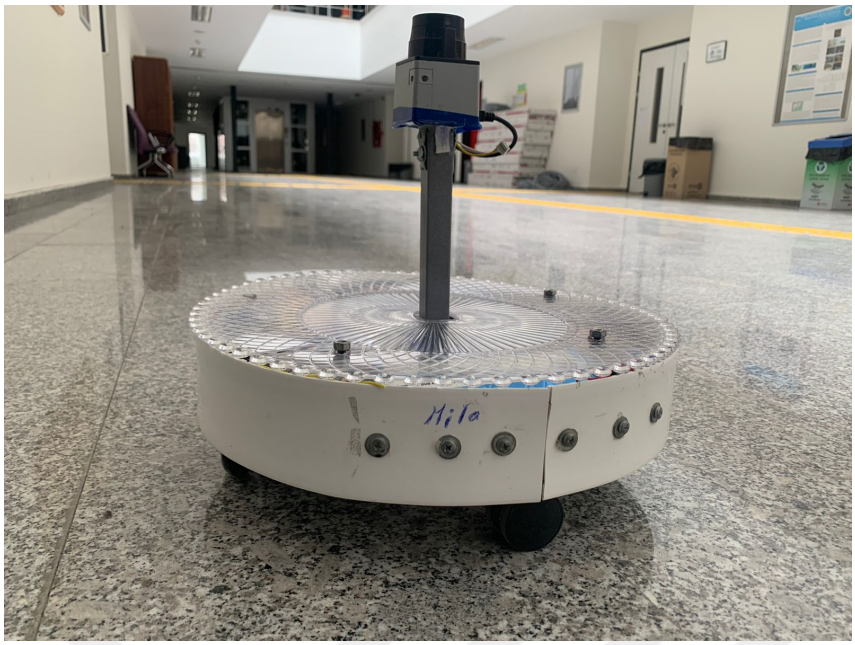



**Figure 3.23** Robot real body

Also this design more convenient to fit the robot footprint on simulation which led to avoiding unwanted stuck in the crowded environment .

## 3.5 Overall Proposed Design of Nodes and Packages

The overall proposed design of nodes and packages will be as follows in Figure 3.24 ,

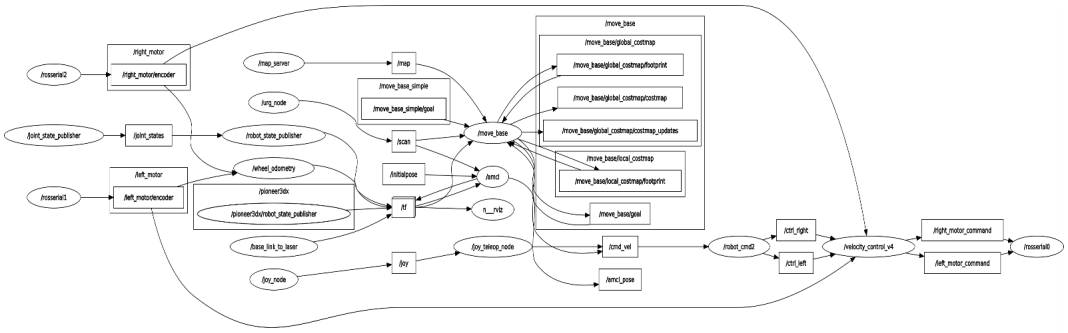



**Figure 3.24** Proposed robot general nodes connections

The default node messages name has been used to write encoder readings using rosserial and arduino, ROS node created on the arduino that reads encoder data and publishes it as a ROS message (e.g., std_msgs/Int32).

**Wheel Velocity Controller Node** geometry_msgs/Twist messages are often used

to send velocity commands, specifying linear and angular velocities for the robot's movement. It interprets the desired velocities received via ROS messages and generates appropriate control signals (PWM, motor voltage, etc.) for the robot's wheels.

**RF2O Odometry Node**

An RF2O odometry node collects data from an RF2O sensor to calculate the robot's odometry, providing information about its position and orientation. nav_msgs/Odometry messages are commonly used to represent odometry data in ROS, containing information about the robot's position and velocity.

**URG Node**

The URG node interfaces with Hokuyo URG laser range finders in ROS, allowing the robot to gather laser scan data for mapping or navigation purposes. sensor_msgs/LaserScan messages are used to represent laser scan data, providing information about distances measured by the laser sensor at various angles.

**Robot_cmd2 Node**

The robot_cmd2 node likely subscribes to the /ctrl_left and /ctrl_right topics to receive control commands for the left and right wheels of the robot, respectively. The messages published on /ctrl_left and /ctrl_right topics might be custom messages or standard messages (e.g., std_msgs/Float64) containing velocity commands or control signals for the respective wheels.

**Rviz Node**

rviz is a visualization tool in ROS used to visualize sensor data, robot models, and trajectories, aiding in debugging and understanding the robot's behavior.rviz can display various visualizations such as laser scans, point clouds, odometry, 3D robot models, maps, and paths.

# 4
# EXPERIMENT SETUP AND RESULTS

This experimental setup chapter delves into a multifaceted exploration that bridges the virtual and physical realms of robotics. This chapter's focus encompasses simulation work, lab work, and the experimental setup's pivotal components, showcasing the intricate connection between theoretical simulations and real-world applications. Simulation Work comprises the utilization of the Turtle Bot 3 World in Gazebo, coupled with RViz as a visualization tool for real-time insights into the robot's environment and behavior. Furthermore, it involves fine-tuning the Proportional-Integral-Derivative (PID) controller, crucial for motion control. Transitioning to Lab Work, emphasis is placed on the physical manifestation of the robotic platform, translating algorithms and simulations into practical scenarios. Mapping the environment and employing RViz for visualization and validation play central roles in understanding the robot's perception and its interaction with the surroundings. Integration with Mapping and the demonstration video in the Experimental Setup showcase the interplay between mapping techniques, enhancing the robot's spatial awareness and decision-making abilities during navigation. The Results and Analysis section juxtaposes scenarios with and without grid-based maps, emphasizing the PID controller's robustness and the profound impact of PID settings on navigation efficiency and accuracy. This comprehensive exploration underscores the symbiotic relationship between simulation and practical experimentation, highlighting the PID controller's crucial role and the significance of mapping techniques in shaping the robot's performance and decision-making in varied scenarios.

## 4.1 Simulation Tools Setup and Testing

### 4.1.1 Gazebo

A simulation tool offering a warehouse environment equipped with a robot model, specifically the TurtleBot3 waffle given in Figure 4.1.

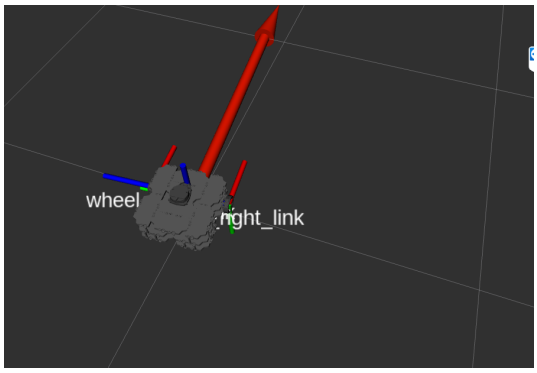



**Figure 4.1** Robot model

It provides a platform for testing and experimenting with robotic algorithms and behaviors within a controlled virtual setting, to be utilized on a warehouse environment given in Figure 4.2.

**Figure 4.2** Gazebo simulation warehouse environment

### 4.1.2 RViz

A visualization tool used in conjunction with Gazebo to offer a graphical representation of the robot's environment. It aids in real-time visualization of sensor data, robot states, and simulated environments, facilitating debugging and algorithm validation given in 4.3.

**Figure 4.3** Warehouse environment map in rviz tool

#### 4.1.2.1 Sensor

The sensor reading in RVIZ tool representation as below in Figure 4.4.

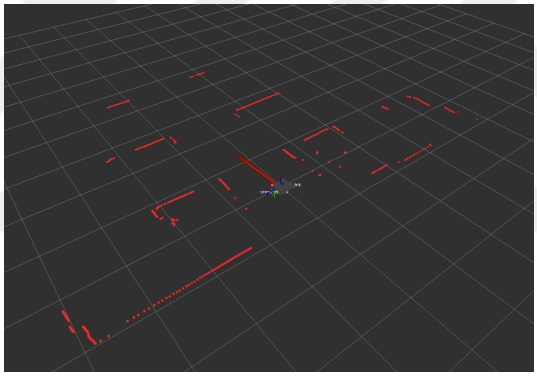


**Figure 4.4** Laser in simulation

### 4.1.3 Global Planner

A* and Low Resolution An amalgamation of the A* algorithm with a low-resolution grid map. This planner is primarily responsible for devising an optimal path from the start to the goal while utilizing a low-resolution map for broader area coverage and efficient computation as shown in Figure 4.5.

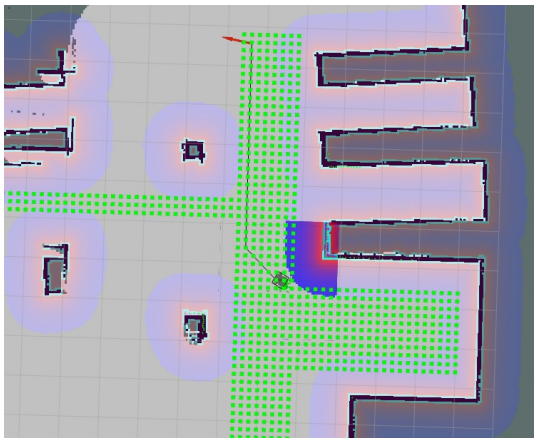


**Figure 4.5** Global planner simulation

### 4.1.4 Local Planner

Pure Pursuit + PID Controller: Combining the Pure Pursuit path tracking algorithm with a Proportional-Integral-Derivative (PID) controller for precise and stable motion control. It defines the robot's local path following strategy by adjusting linear and angular velocities based on the desired trajectory.

**Pending**

### 4.1.5 Testing Procedure in Simulation Platform

After creating the required nodes and packages on our workspace,then creating four main launch files for loading the required nodes packages and the needed steps as below

**SLAM and Mapping:** Create a launch file for the G-Mapping algorithm to create the map and to be loaded to the map server after saving as .pgm and .yamal format.

**Gazebo and robot model:** A launch file has been created To load the gazebo environment from SDF file and then the robot model from the URDF file description.

**Rviz Tool:** To load the map we need to load the RVIZ viewing tool in this launch file which includes Robot Model launch file , Map Server , AMCL Node Rviz ode,Markers Node,Global Planner Node Packages,Local Planner Node Packages,Joystick Controller Node .

**Navigation testing** : A launch file should be created to include a Python navigation order node to navigate the robot from the initial pose to the target .

A set of tests has been driven starting from the Robot's initial pose RINP along between the marked 10 points of x and y , each point's test set has been driven between the initial pose and the set point goal as shown in Figure 4.6 , and then we take the average of 10 results for each point individually to get a table of 10 averages measures resulted of the three metrics ATAG[s], APE[M],AOE[degrees] and , Collision, OSG at final we will get three average values for our metrics.


**Figure 4.6** Warehouse environment map

### 4.1.6 Path Planning based Complicated environment

Employing the parameters outlined above, we endeavored to quantify the duration of path planning precisely at position 5 within the corridor map. We conducted multiple trials, yielding the recorded outcomes depicted in Table 4.1.

**Table 4.1** Dijkstra and A* in implementation time comparison

| | **Dijkstra Algorithm** $(\mu s)$ | **A* Algorithm** $(\mu s)$ |
|---|---|---|
| **Path Planning** | 39524 | 1749 |
| | 39600 | 1743 |
| | 39621 | 1745 |
| | 39841 | 1747 |
| | 39326 | 1741 |
| | 39518 | 1736 |

The aforementioned outcomes were derived from an intricate map configuration, encompassing free of obstacles. The objective was to discern the variance among algorithms regarding the exploration of adjacent cells within the map, striving to identify the most efficient path for the robot to navigate and ultimately reach its designated goal.

### 4.1.7 Optimal PID Parameters in Simulation-Based Platform

The parameters associated with the PID controller and motion control, encompass coefficients for linear velocity control in the simulation platform. Additionally, maximum linear (VMax) and angular (W Max) velocities, along with the look ahead parameter, are considered for path tracking and navigation as shown in Table 4.2.

**Table 4.2** Optimal PID parameters in simulation platform

| **Parameter** | **Value** |
|---|---|
| KPlin | 0.7 |
| KDlin | 0.2 |
| KIlin | 0.25 |
| KP ang | 1.9 |
| KD ang | 0.97 |
| KIang | 0.45 |
| Vmax | 0.26 |
| W Max | 1.28 |
| **Look Ahead** | 12.90 |

### 4.1.8 Comparison of Proposed Design with Alternative Global and Local Planners in Simulated Grid-Based System

Following an extensive examination involving simulated mobile robot navigation, the integration of A* with PID-based Pure Pursuit yielded significant outcomes, specifically in averting collisions and overshooting the set goals. The obtained results, which encompassed the performance evaluation of global planners such as Dijkstra/A* and local planners post-driving tests, Dynamic Window Approach DWA and Time Elastic Band (TEB) where DPPD refers to a combination of Dijkstra with pure pursuit and PID controller, APPD refers to a combination of A* with pure pursuit and PID controller, APP refers to a combination of A* without pure pursuit and PID controller, ADWA refers to a combination of A* with Dynamic Window Approach, and ATEB refers to a combination of A* with Time Elastic Band detailed in Table 4.3.

**Table 4.3** Global and local planners comparison in simulation

| | DPPD | APPD | APP | ADWA | ATEB |
|---|---|---|---|---|---|
| ATAG[s] | 75.091 | 74.125 | 83.847 | 94.884 | 79.756 |
| APE[M] | 0.187 | 0.152 | 0.163 | 0.466 | 0.467 |
| AOE[degrees] | 1.434 | 1.534 | 0.146 | 0.135 | 0.318 |
| Collision | 1 | 0 | 1 | 0 | 0 |
| OSG | 0 | 0 | 0 | 0 | 16 |

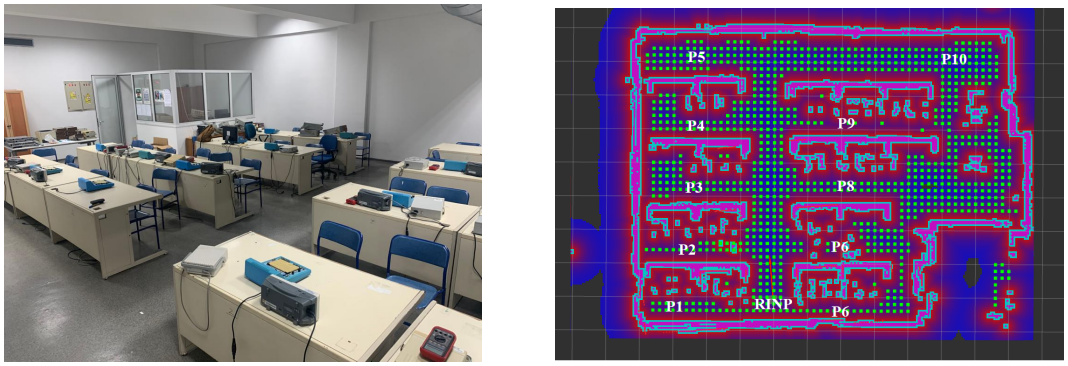
## 4.2 Real Platform Tools Setup and Testing

### 4.2.1 RViz

A visualization tool used in conjunction with Gazebo to offer a graphical representation of the robot's environment. It aids in real-time visualization of sensor data, robot states, and simulated environments, facilitating debugging and algorithm validation.

#### 4.2.1.1 Environment - A

After mapping the real environment in Figure 4.6 a The lab environment has been represented on Figure 4.6 b this map is the most complex map that has been tested due to the narrow ways between obstacles 4.7.



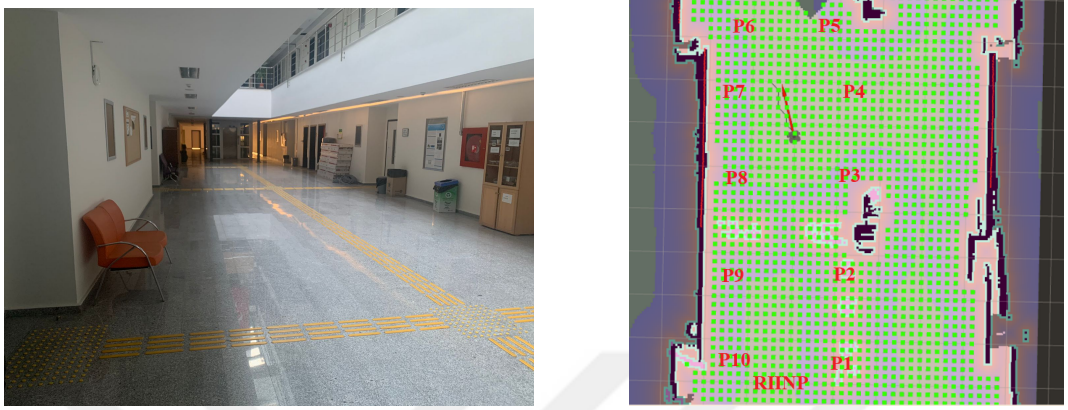

a b

**Figure 4.7** Corridor environment

The green markers have been the auto distribution of low resolution grid in the non-occupied free spaces, also the distributed ten difference targets from P1to P10 to be obtained on the map with the Robot initial pose RINP .

#### 4.2.1.2 Environment -B

was resulted from the mapped real corridor environment shown if Figure 4.8 a to be represented in RVIZ tool as shown in Figure 4.8 b for more study on a set of tests in the corridor open area environment free of obstacles.




a b

**Figure 4.8** Corridor environment

Also there are a green markers of low resolution grid to be distributed over the non-occupied free spaces .

### 4.2.2 Global Planners Real Platform

A* + Low Resolution: An amalgamation of the A* algorithm with a low-resolution grid map. This planner is primarily responsible for devising an optimal path from the start to the goal while utilizing a low-resolution map for broader area coverage and efficient computation.

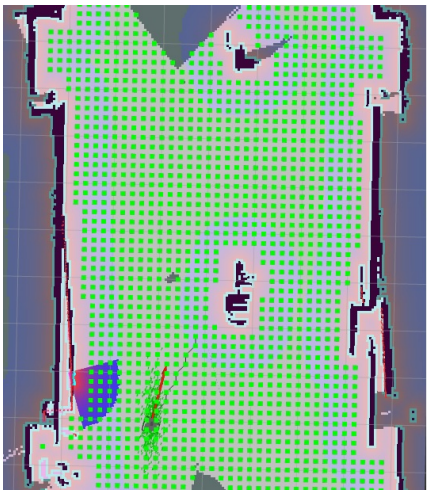


**Figure 4.9** Global planner A* with low resolution grid

### 4.2.3 Local Planners Real Platform

Pure Pursuit + PID Controller: Combining the Pure Pursuit path tracking algorithm with a Proportional-Integral-Derivative (PID) controller for precise and stable motion control. It defines the robot's local path following strategy by adjusting linear and angular velocities based on the desired trajectory.

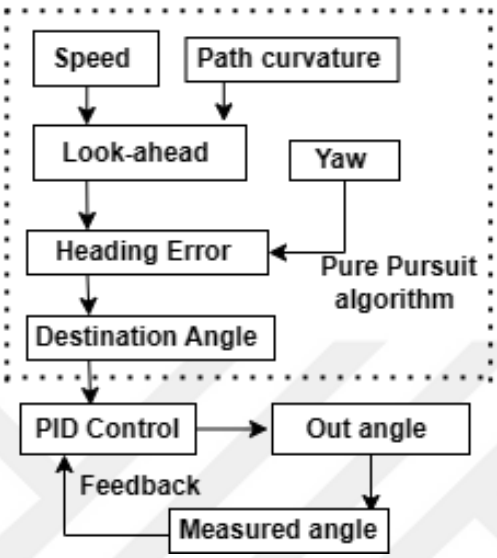



**Figure 4.10** Pure Pursuit with PID controller

### 4.2.4 Odometry Setup

The proposed wheel odometry is calculated from encoder ticks,the Robot travelled distance is calculated using left and right wheel distance. Here, Dl and Dr are the wheel distance, and d is the robot travelled distance. Dl and Dr are calculated from the encoder ticks. We know the distance per tick value (metres per tick).

For example, we get 7000 ticks from the right wheel and 7000 ticks from the left wheel.

Assuming ticks per metre is 30,000 :

**Right wheel distance** = Right wheel ticks / Ticks per metre = 7000/30000 = 3.333 m = $D_r$

**Left wheel distance** = Left wheel ticks / Ticks per metre = 7000/30000 = 3.333 m = $D_l$

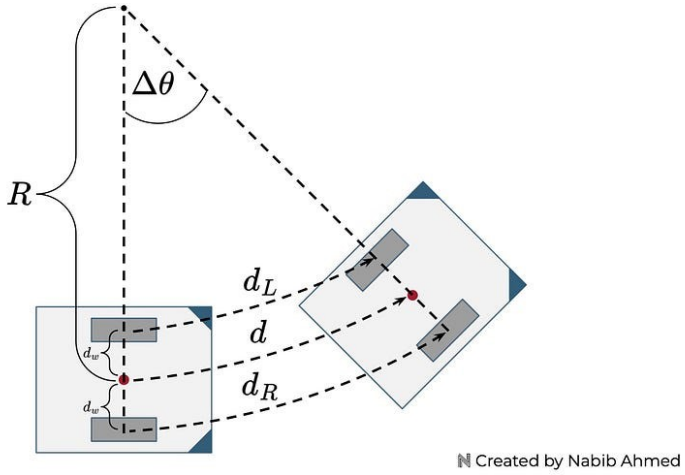



**Figure 4.11** Right and Left wheel travel distance calculation

$d_L =$ distance traveled by the left wheel

$d_R =$ distance traveled by the right wheel

$d_w =$ distance between the reference point and the wheels

$d =$ distance traveled by the reference point

$\Delta\theta =$ change in the angle of rotation

$R =$ radius of the curve containing the reference point

The first three variables can be directly measured (using the encoder for the first two variables and a ruler for the third variable). The last three variables are not directly measurable — instead, we need to use geometry to relate these variables to the measurable quantities.

$$\begin{aligned} d &= \frac{d_L + d_R}{2} \\ \Delta\theta &= \frac{d_R - d_L}{2d_w} \end{aligned} \tag{4.1}$$

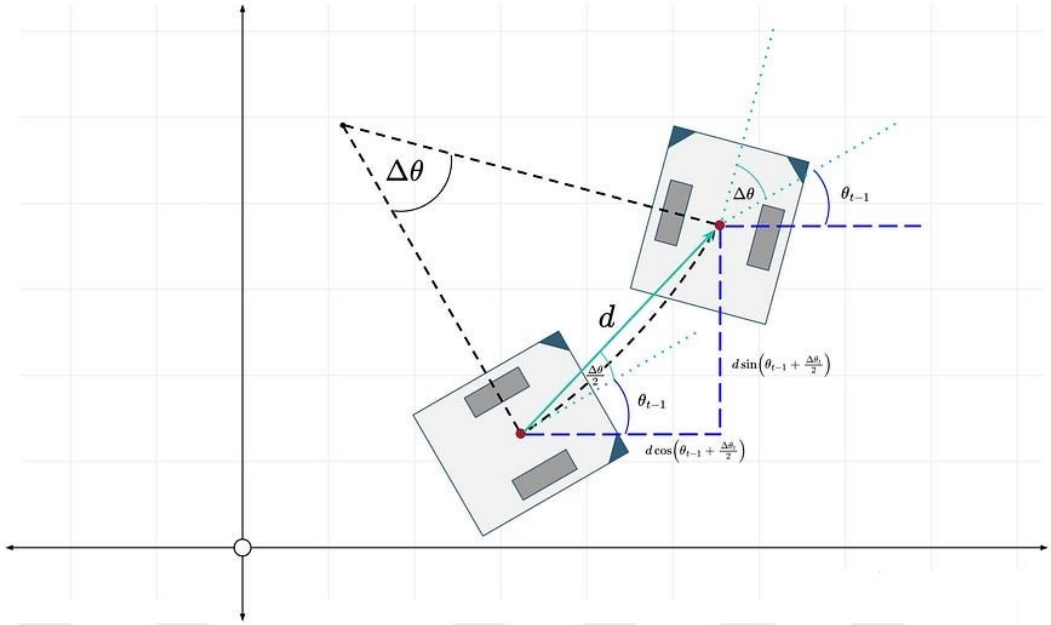



**Figure 4.12** Wheel odometry

$$x_t = x_{t-1} + d\cos\left(\theta_{t-1} + \frac{\Delta\theta_t}{2}\right)$$
$$y_t = y_{t-1} + d\sin\left(\theta_{t-1} + \frac{\Delta\theta_t}{2}\right)$$
$$\theta_t = \theta_{t-1} + \Delta\theta_t$$

1.57 Radian = 1.57 * 180/3.14 = 90 Degrees

Have to check transformation from radians to degrees between wheel tick odometry and velocity controller.

### 4.2.5 Optimal Parameters Real Based Platform

We tried to utilize the same simulation parameters that associated with the PID controller and motion control, encompass coefficients for linear velocity control. Additionally, we revised the maximum angular (W Max) velocity from 1.28 to 0.26 because we got some control system failarity, along with the Look Ahead parameter, to be considered for path tracking and navigation as below Table 4.4.

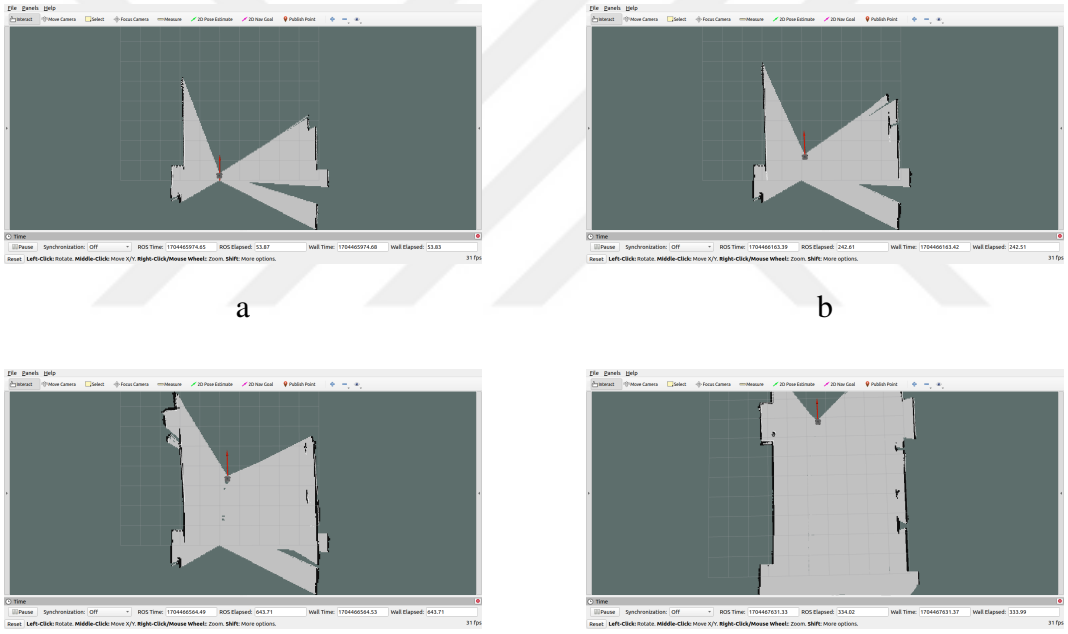
### 4.2.6 Testing Procedure in Real Platform

In real platform we will use the same creating the required nodes and packages on our workspace,then creating four main launch files for loading the required nodes packages and the needed steps as below .

**Table 4.4** Optimal parameters in real platform

| **Parameter** | **Value** |
|---|---|
| KPlin | 0.7 |
| KDlin | 0.2 |
| KIlin | 0.25 |
| KP ang | 1.9 |
| KD ang | 0.97 |
| KIang | 0.45 |
| Vmax | 0.26 |
| W Max | 0.26 |
| **Look Ahead** | 5.0 |

**SLAM and Mapping:** Create a launch file for the G-Mapping algorithm to create the map and to be loaded to the map server after saving as .pgm and .yamal format 4.13.



a b

c d

**Figure 4.13** Mapping steps

**Gazebo and robot model:** A launch file has been created To load the gazebo environment from SDF file and then the robot model from the URDF file description.

**Rviz Tool:** To load the map we need to load the RVIZ viewing tool in this launch file which also includes the robot model launch file, map server, AMCL node, rviz node, markers node, global planner node packages, local planner node packages, joystick controller node.

**Navigation testing** : A launch file should be created to include a Python navigation order node to navigate the robot from the initial pose to the target .

A set of tests has been driven starting from the Robot's initial pose RINP along between the marked 10 points of x and y , each point's test set has been driven between the initial pose and the set point goal as shown in Figures 4.8a and b , and then we take the average of 10 results for each point individually to get a table of 10 averages measures resulted of the three metrics, average time to reach the goal in seconds ATAG[s],average position error in meters APE[M], average orientation error AOE[degrees] and, Collision, overshooting around the goal OSG at final we will get three average values for our metrics.

### 4.2.7 Proposed Design with Alternative Global and Local Planners in Real Platform Grid-Based System Comparison

The tests have been driven in a free obstacle corridor environment as shown in Table 4.5.

**Table 4.5** Real platform global and local planners comparison

| | DPP | APPD | APP | ADWA | ATEB |
|---|---|---|---|---|---|
| ATAG[s] | 27.5905 | 28.723 | 36.849 | 36.1563 | 25.8939 |
| APE[M] | 0.76195 | 0.526452 | 0.60936 | 0.692565 | 0.55076 |
| AOE[degrees] | 1.52505 | 1.44126 | 0.224 | 1.51735 | 1.80465 |
| Collision | 1 | 0 | 2 | 0 | 10 |
| OSG | 0 | 0 | 10 | 5 | 15 |

The results obtained from the amalgamation of five models yielded five metrics, identifying the A* with Pure Pursuit and PID controller (APPD) amalgamated model as the most count on it with a focus on an effective collision-free system.

### 4.2.8 Multiple PID Parameters in Real Platform Grid-Based Environment Comparison

The purpose of the experiment was to assess how different PID controller parameters affected the navigation system on a platform grid-based setting. Five tiers with different PID parameter settings were examined to determine how they impact the efficiency and stability of navigation.to show the effect of the PID controller parameters, we conducted a set of tests altering the PID Linear and angular trajectory PID controller, In Tier1 the Linear Velocity K PID Parameters

have been set to zeros. In Tier 2 the Angular Velocity K PID Parameters have been set to zeros. The Linear K Derivative and Integral have been set to zeros in Tier 3. while K PID magnitude have been set to zeros in Tier 4 Linear and angular. Tier 5 Through many trials and analysis K Integral has been set to zeros as shown in Table 4.6 .

**Table 4.6** Real platform PID parameters comparison

| **Tier No** | **Tier1** | **Tier2** | **Tier3** | **Tier4** | **Tier5** |
|---|---|---|---|---|---|
| **Parameter** | **Value1** | **Value2** | **Value3** | **Value4** | **Value5** |
| KPlin | 0 | 0.35 | 0.35 | 0 | 0.72 |
| KDlin | 0 | 0.01 | 0 | 0 | 0.9 |
| KIlin | 0 | 0.01 | 0 | 0 | 0.2 |
| KPang | 1 | 0 | 1 | 0 | 0.2 |
| KDang | 3 | 0 | 3 | 0 | 0.01 |
| KIang | 0.1 | 0 | 0.1 | 0 | 0 |
| Vmax | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| Wmax | 0.3 | 0.3 | 0.3 | 0.3 | 0.3 |
| **LH** | 5 | 3 | 5 | 3 | 3 |
| **Efficiency** | 0 | 0.15 | 0.25 | 0.0 | 0.90 |

### 4.2.9 Proposed Local Planner and Other Navigation Local Planner Models in Complex Environment Comparison

The optimal Tier 5's PID controller parameters have been utilized with increasing the complexity of the environment by placing 3 obstacles in a low resolution-based environment, with a distance of 120 cm between the obstacles we got the below results shown in Table 4.7.

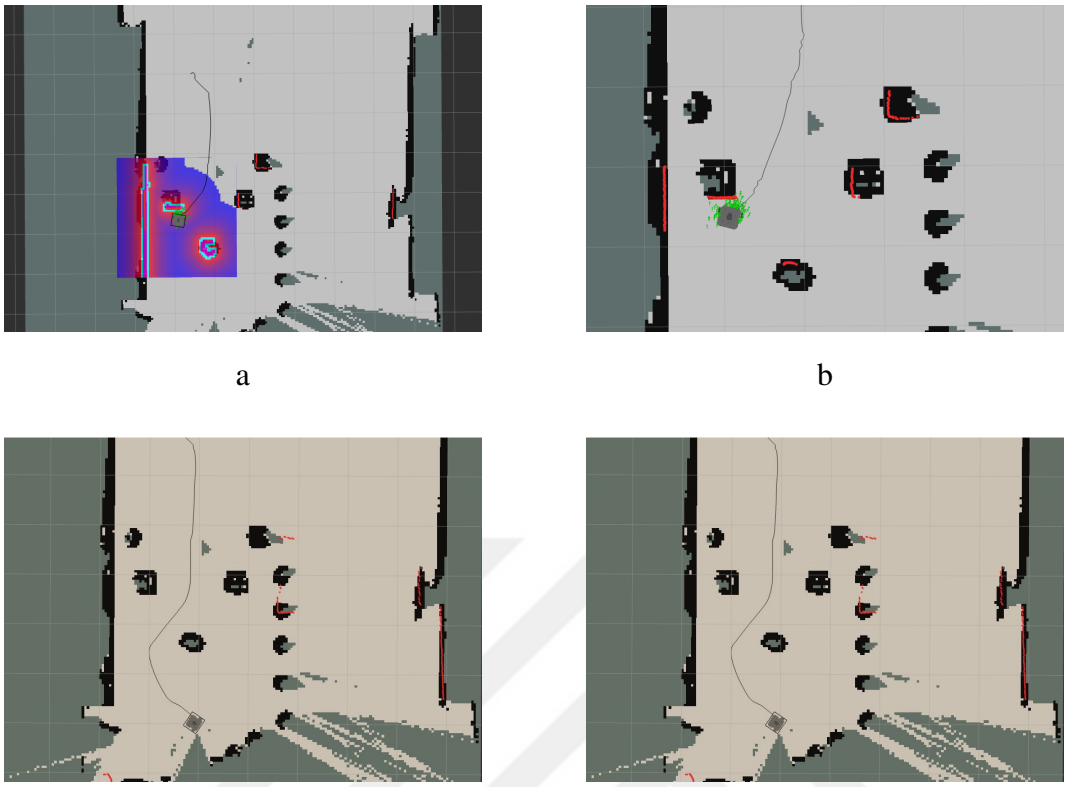
**Table 4.7** Assessment of sophisticated models within a complex environment

| | DPPD | APPD | APP | ADWA | ATEB |
|---|---|---|---|---|---|
| Tier 1 | Passed | Passed | Failed | Passed | Failed |
| Tier 2 | Passed | Passed | Failed | Passed | Failed |
| Tier 3 | Passed | Passed | Failed | Passed | Failed |
| Tier 4 | Failed | Passed | Failed | Failed | Passed |
| Tier 5 | Failed | Passed | Failed | Failed | Failed |

### 4.2.10 Lack of Low Resolution Grid

We tried to navigate the robot using the same parameters resulted from the tires in Table 4.7 but without a grid-based map so we got fail and crashed with the obstacle

as per the four cases shown in below 4.14.



a b

c d

**Figure 4.14** Crashing in system without grids

### 4.2.11 Proposed System Development

Based on that combination development of our model we evaluated the steps of model development as shown in Table 4.8.

**Table 4.8** Proposed model development

| | NAVIGATION IMPROVEMENT IMPACT PERCENTAGE |
|---|---|
| A* with Pure pursuit Low-resolution Grid Based | %25 |
| A*, Pure pursuit with PID Low-resolution Grid Based | %75 |
| A*, Path re-planning, Pure pursuit with PID Low-resolution Grid Based | %90 |

# 5
# DISCUSSION

In this chapter, we will review the parameters and the impact of their changes on the system, a pre-comparative analysis between the Dijkstra and A* global planners was conducted. The aim was to discern variances in planning time across intricate environments using the metric. Our choice to utilize a global planner configured with a grid of discrete points was instrumental in this comparative evaluation, our paradigm of Astar with a low-resolution grid and Pure pursuit with PID controller we call it (Pure Pursuit with PID). Furthermore, motivated by the adaptability inherent in the pure pursuit paradigm, we endeavored to integrate a PID (Proportional-Integral-Derivative) controller into our trajectory controller. This integration was intended to facilitate the tracking of the path generated by the global planner within the context of a low-resolution grid structure. Comparing our paradigm with other local planners using those three metrics average time to arrive at the goal [s], average position error [meters], and Average orientation error [ degrees] , we got the shortest time with TEB but with a lot of overshoots around the goal while we got best time to reach the goal without overshoots with following path precisely, but we got highest average position error in Dijkstra with Pure Pursuit with PID .the lowest Average orientation error in degree happened with A* and normal pure pursuit while the worst value Dijkstra.

## 5.1 Simulation Outcomes Analysis

### 5.1.1 Comparing A* with Dijkstra

a comparative analysis between A* and Dijkstra in terms of path planning time, conducted amidst an environment populated with four obstacles, culminated in the results of a series of adjustments that led to the development of a stable navigation system, as evidenced by the outcomes of path planning tabulated in Table4.1.

The reason why A* performs better than Dijkstra is due to the use of heuristics, which guides the search algorithm to explore cells that are potentially closer to the

destination as shown in Figure 2.9. This avoids the exploration of far away cells as in Dijkstra, which reduces the time taken by A* by a factor of 20. It should be noted that the heuristic function's efficiency in A* depends on the complexity of map being used, and the time improvement may reduce in more complicated maps.

### 5.1.2 Parameter Tuning - Simulation

Analysis of the simulation data depicted in Table 4.2 illuminated the substantial impact of augmenting both look-ahead distance and angular velocity. This augmentation notably reduced the time taken for the waypoint approach and overall traversal time towards the goal.

The optimal linear and angular PID values were found using the ZN tuning method and multiple trial and error runs. The max linear and angular velocity were chosen based on the specifications of the motors, and the response time of the system to dynamic obstacles. It was crucial to find the highest possible velocity that the robot can travel in while making sure to follow the local plan and avoid collisions.

The proportional term allows the robot to align itself to the required orientation, and move forward towards the goal. Having a high P value will make the corrections more aggressive and faster.

The differential term is responsible for preventing any overshoot that might happen during navigation.

The integral value is responsible for preventing any constant error in orientation of the robot. Having a low I value is optimal for the application, since a high value can cause the robot to oscillate unnecessarily.

An optimal lookahead parameter value needs to be chosen to make sure the robot can detect curves in the local plan with sufficient time left. Having a small lookahead parameter reduces the amount of processing done on the local planner, but the robot reacts suddenly to turns and curves in the path. This leads to overshoots in the robot, therefore increasing the navigation time. Having a large lookahead parameter makes the robot follow the path in a smoother and more precise manner, but increases the amount of processing done on the local planner. A suitable value was chosen such that the robot follows the path precisely while making sure the time taken to reach the goal is less.

### 5.1.3 The Effect of Look-Ahead Distance and Angular Velocity Augmentation on Navigation Efficiency

Analysis of the simulation data depicted in Table 4.2 illuminated the substantial impact of augmenting both look-ahead distance and angular velocity. This augmentation notably reduced the time taken for the waypoint approach and overall traversal time towards the goal. The utilization of our low-resolution grid demonstrated pronounced efficacy in minimizing instances of overshooting and averted system crashes. The navigation system underwent meticulous testing, yielding the outcomes detailed in Table 4.3.

## 5.2 Real Platform Discussion

### 5.2.1 Challenges Encountered Throughout The Process of Converting Simulations Optimal Parameters to Real-World Scenarios

Translating the simulated parameters onto the physical platform uncovered a significant issue arising from excessively high angular speed. This phenomenon led to recurrent instances of self-overshooting by the robot, resulting in wheel slippage and subsequent deviation from the intended path. Consequently, a reduction from 1.28 to 0.26 in angular speed was implemented, as depicted in Table 4.4. Subsequently, a comparative analysis between A* and Dijkstra in terms of path planning time, conducted amidst an environment populated with four obstacles, culminated in the results presented in Table 4.5. This series of adjustments led to the development of a stable navigation system, as evidenced by the outcomes tabulated in Table 4.6.

### 5.2.2 Path Planning based Complicated Environment Real Platform Outcomes

Path planning in intricate environments is a crucial element of robotics, especially in situations where robots maneuver through difficult places, such as tight hallways, crowded areas, or environments with moving impediments. The efficacy of path planning algorithms in such circumstances has a direct impact on a robot's capacity to move securely and optimally. The complexity of path planning in such contexts is influenced by multiple aspects. These factors encompass the existence of barriers, restricted area, variations in landscape, ambiguities in sensor information, and the requirement for immediate decision-making. In order to tackle these difficulties, academics and engineers frequently employ advanced algorithms, such as A*, Dijkstra's, or other heuristic-based methods. Considering the given corridor map, various parameters were employed to evaluate the effectiveness and precision of

the path planning procedure. The parameters encompass factors such as the robot's position (x, y, theta), the dimensions of the corridor, the presence and movement of obstacles, sensor accuracy, robot velocity limits, and the algorithm's capacity to adapt and identify an optimal path while evading collisions.The collected results from numerous experiments provide significant data for assessing the efficacy of the path planning algorithm. The algorithm's performance may have been evaluated by assessing metrics such as the time taken for path planning, the smoothness of the trajectory, the successful completion of navigation tasks, and the ability to manage dynamic obstacles. Furthermore, the discussions may explore the compromises between the time it takes to perform calculations and the level of precision achieved, the constraints of certain algorithms in intricate settings, the possible improvements or adjustments required for the algorithm, and the importance of reliable path planning for practical uses like self-driving cars, warehouse robots, or other mobile robots operating in limited areas. Path planning in intricate environments is a multidimensional field that requires thorough assessment, optimization, and adjustment of algorithms to guarantee secure and efficient navigation for autonomous systems. An examination of recorded results is used to comprehend the algorithm's performance and gain insights on enhancing path planning techniques for difficult surroundings. The biggest difference between A* and Dijkstra lies in their approach to examining neighboring cells in order to determine the most efficient path, as shown in Figure 2.9. Our analysis, as presented in Table 4.1 , reveals that A* requires significantly less planning time compared to Dijkstra, with a reduction of 20-fold.

### 5.2.3 Performance Analysis of Global and Local Planner Combinations in Real Platform Grid-Based Systems

The experiment involved evaluating various combinations of global and local planners in a real platform grid-based system.Table 4.5 showcases the outcomes obtained from these trials. The metrics analyzed included average time to arrive at the goal (ATAG), average position error (APE), average orientation error (AOE), instances of collision or getting stuck, and cases of overshooting the goal (OSG).

These performance indicators were used to evaluate every combination of planners. Notably, positive outcomes were shown when A* was combined with Pure Pursuit and PID Controller (APPD). When compared to alternative setups, it showed the shortest average time to reach the goal as well as the lowest average position and orientation mistakes.

The combination of A* and Time Elastic Band (ATEB), on the other hand,

demonstrated competitive performance, particularly in lowering orientation mistakes, while taking a little longer on average to reach the objective. In contrast, combinations that exhibited a lower number of collisions or stuck times, such as A* with Dynamic Window Approach (ADWA), were significantly overshooting the aim.

The data presented in Table 4.5 sheds important light on how well various planner combinations work in the free obstacle corridor scenario. Although certain configurations performed better than others in particular metrics, there was a trade-off between various characteristics. Configurations that performed exceptionally well at minimizing errors, for example, took longer on average to attain the objective or had more overshooting.

In a real platform grid-based system, the outcomes highlight the need of carefully choose and fine-tuning global and local planner combinations utilizing the default setting of local planners to be contrasted with our model combination of A* with pure pursuit and PID controller. These results can help guide future decisions about robot navigation strategy optimization, taking into account the many trade-offs between measures like accuracy, time efficiency, and avoiding collisions or overshooting.

TEB planning component would almost be the shortest in time if it were not for the swing around the target and hesitating back and forth on it to correct the path and match the robot's body with the target point. The reason for that was the small value of goal tolerance in the TEB default setting.Therefore, the observed swing and hesitancy around the target point in the TEB planning component are often due to the fine-tuned nature of the goal tolerance parameter. Adjusting this parameter to allow a small margin of error or providing additional constraints could help mitigate the excessive corrections and reduce the swing effect, enhancing the algorithm's efficiency in matching the robot's body with the target while avoiding unnecessary oscillations.

### 5.2.4 A Study on Varied Linear and Angular PID Controller Settings in a Real Platform Environment

In a real platform grid-based environment, the experiment aimed to assess the effects of different PID controller parameters on the navigation system. To determine their impact on navigation efficiency and stability, five tiers with different PID parameter settings were examined.

**Proportional parameter** ($K_P$) in a PID (Proportional-Integral-Derivative)

controller used for robot steering is essential for regulating the steering control's precision and response. It has a significant impact on robot steering and influences several critical factors.

**Responsiveness to Error:** The proportional term directly responds to the current error between the desired heading or orientation and the robot's actual heading. A higher proportional gain increases the corrective action taken in response to steering errors. Consequently, the robot tends to respond more quickly to deviations from the desired path.

**Rate of Correction:** Adjusting the proportional parameter changes the rate at which the steering control corrects the robot's heading error. A higher K_P leads to a faster rate of correction, allowing the robot to react swiftly to deviations.

**Overshoot and Oscillations:** While increasing the proportional gain can enhance the system's responsiveness, it can also cause overshooting or oscillations around the desired heading. If the gain is too high, the steering may become too aggressive, leading to overshoot or unstable oscillations.

**Stability:** Properly tuning the proportional gain is crucial for maintaining stability during steering. An optimal K_P value ensures the robot can effectively correct errors without introducing instability or erratic behavior.

**System Specificity:** The ideal proportional gain depends on the specific characteristics of the robot's steering system. Different robots, environments, and steering mechanisms may require varying K_P values to achieve optimal performance.

**Error Reduction:** The proportional term contributes to reducing heading errors. It acts in proportion to the current error, providing an immediate correction force to bring the robot closer to the desired heading.

The proportional parameter in the PID controller for robot steering influences the system's responsiveness, rate of correction, stability, and error reduction. Fine-tuning the proportional gain is crucial to strike a balance between rapid error correction and preventing overshoot or instability, ensuring accurate and stable steering performance for the robot.

**Integral parameter (K_I)** in a PID (Proportional-Integral-Derivative) controller for robot steering has several notable impacts on the control system:

**Error Reduction Over Time:** The integral term in the PID controller accumulates

the error over time. It is responsible for addressing any residual steady-state error that persists even after the proportional and derivative terms have responded to the immediate error. This integration of error over time helps in continuously reducing the steady-state error, ultimately aiming for zero error.

**Eliminating Steady-State Error:** When there is a constant error in the system, such as a consistent bias or drift in the robot's steering, the integral term steadily accumulates this error and applies a corrective action. This correction helps to eliminate the steady-state error, ensuring that the robot eventually reaches and maintains the desired trajectory or heading.

**Response to Constant Disturbances:** K_I is particularly effective in addressing constant disturbances or biases in the system. It continuously adds a corrective signal that increases with time if there is a persistent error, contributing to sustained corrections to counteract these disturbances.

**System Stability:** While the integral term helps in reducing steady-state error, excessive K_I values can lead to instability or overshoot in the system. Inappropriately high integral gain can cause the robot to oscillate or exhibit overshoot around the desired trajectory.

**Slower Response to Sudden Changes:** Unlike the proportional term, which responds immediately to changes in error, the integral term's impact on the steering control is gradual. It takes time to accumulate enough error for a substantial corrective action, making it slower to respond to sudden changes.

**Tuning Challenges:** Tuning the integral parameter requires careful consideration. Setting K_I too high can lead to instability, while setting it too low may not effectively eliminate steady-state errors.

The integral parameter in a PID controller for robot steering plays a significant role in reducing steady-state errors by continuously accumulating and integrating the error over time. It ensures a more accurate and stable trajectory or heading control, but its tuning is crucial to prevent instability while addressing constant errors in the system.

**Derivative parameter (K_D)** in a PID (Proportional-Integral-Derivative) controller for robot steering has several impacts on the control system:

**Damping Oscillations:** The derivative term in the PID controller is responsible for predicting the system's future behavior based on the rate of change of error. It helps in reducing oscillations or overshoot by providing damping to the system's

response. When there's a rapid change in error, the derivative term acts to counteract this change, thereby reducing the system's response to sudden variations.

**Improved Stability:** The derivative term's primary function is to anticipate the error's rate of change, contributing to the stability of the control system. By mitigating rapid changes in error, it prevents the system from reacting too aggressively to sudden disturbances, leading to a more stable response.

**Faster Response to Changes:** Unlike the integral term, which responds slowly to accumulated errors, the derivative term reacts to changes in error immediately. It enables the controller to respond quickly to changes in the error signal, which can be particularly useful when the system encounters sudden disturbances or changes in the environment.

**Suppression of Overshoot:** K_D helps in minimizing overshoot by generating a damping effect. By controlling the speed of the system's response to rapid changes in error, the derivative term can help reduce the extent of overshoot that might occur when the system is subjected to sudden disturbances.

**Noise Amplification:** In some cases, the derivative term can amplify high-frequency noise present in the error signal, leading to instability or erratic behavior. Careful tuning of the derivative gain is necessary to prevent amplification of noise in the system.

**Tuning Challenges:** Properly tuning the derivative parameter is essential. Too high a K_D value can lead to excessive damping, slowing down the system's response and making it unresponsive to rapid changes. On the other hand, setting K_D too low may not effectively dampen the oscillations or overshoot.

The derivative parameter in a PID controller for robot steering influences the system's response to rapid changes in error. It helps in damping oscillations, improving stability, suppressing overshoot, and providing a faster response to sudden changes, but it requires careful tuning to avoid amplification of noise or excessive damping.

The linear velocity parameters ($K_P^{lin}$, $K_D^{lin}$, and $K_I^{lin}$) were all set to zero in Tier 1, which had a significant effect on the navigation system's efficiency. Similar to Tier 1, which emphasized the importance of angular velocity in system performance, Tier 2 included setting the angular velocity parameters ($K_P^{ang}$, $K_D^{ang}$, and $K_I^{ang}$) to zeros.

Tier 3 adjusted the integral parameters and linear velocity derivative to zeros,

indicating how these factors affect stability. On the other hand, Tier 4 required setting the angular and linear PID parameters to zero, demonstrating the significance of these elements in attaining effective navigation.

Tier 5 yielded the best design when the angular integral parameter ($K_I^{ang}$) was zero. As seen in Table 4.6, this modification reduced oscillations and swinging around the path, making the navigation system more stable.

Analyzing the values across the tiers, it is evident that variations in the proportional, derivative, and integral parameters significantly impact navigation efficiency and stability. For instance, altering the proportional and integral components played a crucial role in refining the navigation system's performance.

The efficiency indicator showed different levels of influence depending on the parameters in all of them. Tier 5, which was found to be the best configuration after trials, demonstrated a significant increase in efficiency, up to 90 percentage. This indicates that a significant portion of the system's effectiveness was derived from the precise parameters, specifically from setting the angular integral parameter to zero.

### 5.2.5 Evaluation of Local Planner Configurations in Obstacle-Enriched Complex Environments with Tiered PID Parameters

A comprehensive comparison of various local planner models within a complex environment was undertaken, employing the Tier 5 PID controller parameters. This investigation was conducted within a low-resolution-based environment, augmented by the introduction of three obstacles spaced 120 cm apart. The outcomes are meticulously detailed in Table 4.7.

Table 4.7 portrays the performance evaluation of sophisticated models within this complex environment. Each tier denotes a specific configuration of the local planner models tested in this environment. The models evaluated include DPPD (Dijkstra with Pure Pursuit and PID Controller), APPD (A* with Pure Pursuit and PID Controller), APP (A* with Pure Pursuit without PID Controller), ADWA (A* with Dynamic Window Approach), and ATEB (A* with Time Elastic Band).

In Tier 1, Tier 2, and Tier 3, the models DPPD, APPD, and APP demonstrated a successful navigation performance, as evidenced by the "Passed" designation. However, the APP model encountered difficulties and failed to perform adequately, indicated by the "Failed" designation. This trend persisted across these tiers, affirming the robustness and efficacy of the DPPD, and APPD models in navigating

the complex environment.

Moving to Tier 4, while APPD and ADWA exhibited favorable performance, the DPPD and ATEB models encountered challenges and were unable to navigate effectively, resulting in a "Failed" designation. Finally, in Tier 5, only the APPD model demonstrated successful navigation, whereas the other models, including DPPD, APP, ADWA, and ATEB, faced significant hurdles, leading to a "Failed" designation for their performance.

These findings underscore the varying degrees of success and failure among the tested local planner models within the complexity of this environment. Notably, the consistent performance of APPD across multiple tiers signifies its reliability, whereas other models exhibited mixed success and limitations. Further analysis and refinement of these models are warranted to ascertain their viability in addressing challenges within intricate environments.

### 5.2.6 Development Consider for the Suggested System

An attempt was made to navigate the robot using Tier 5 parameters without relying on a grid-based map, which resulted in failure as the robot collided with an obstacle. This outcome is visually depicted through four distinct cases illustrated in Figure 4.14.

Reflecting on the developmental process of our model and its evolution, a comprehensive evaluation of the model's stages is presented in Table 4.8.

Table 4.8 provides insight into the proposed model's developmental stages and their corresponding impact percentages on navigation improvement. Initially, employing the A* algorithm coupled with Pure Pursuit in a low-resolution grid-based environment led to a modest improvement of 25%.

Advancing the model by integrating the A* algorithm, Pure Pursuit, and PID controller within the same low-resolution grid environment significantly enhanced navigation performance, yielding a substantial 75% improvement.

Further refinement ensued with the integration of path re-planning alongside the A* algorithm, pure pursuit, and PID controller within the low-resolution grid environment. This augmentation led to a remarkable 90% improvement in navigation performance, signifying a substantial advancement in the model's efficacy.

The delineation of these stages in Table 4.8 highlights the iterative nature of model

development and the consequential enhancement in navigation efficiency with each stage.

Additionally, Table 4.8 supplements this developmental trajectory by presenting a comparative analysis of the respective model stages. It portrays the evolution of the model's functionality, underscoring the progression from an initial moderate improvement to a significantly augmented navigational capability through strategic iterations and refinements.

These tables collectively underscore the iterative process and incremental improvements made throughout the model's development. They elucidate the pivotal role of integrating path re-planning strategies and the PID controller within the A* and pure pursuit framework, ultimately culminating in a substantial enhancement in the robot's navigation capabilities within a challenging environment.

# 6
# CONCLUSION

Following an extensive examination involving real platform mobile robot navigation, the integration of A* with PID-based Pure Pursuit yielded significant outcomes, specifically in averting collisions and overshooting the set goals. The obtained results, which encompassed the performance evaluation of global planners such as Dijkstra or A and local planners post driving tests, are detailed in Table 4.5 .

The evolution of Wheeled Mobile Robots (WMRs) and Autonomous Ground Vehicles (AGVs) has witnessed a transformative journey, evolving from conceptual ideation to sophisticated technological systems that are now indispensable across various industries including manufacturing, logistics, healthcare, and exploration. This study provides an extensive exploration into the historical trajectory of WMRs, charting significant milestones, technological advancements, and pivotal events that have shaped their evolution.

The primary aim of this research endeavor has been to optimize the navigation system by incorporating a low-resolution mapping technique, specifically targeting the challenge of overshooting encountered in the fusion of A* with pure pursuit. The proposed solution involves the strategic placement of dots within the generated map, meticulously spaced at intervals of 0.2 meters to form a grid structure, effectively delineating a restricted navigation area within manufacturing facilities.

The integration of the low-resolution grid system serves a dual purpose, focusing on enhancing the precision of waypoint navigation while addressing the issue of overshooting. The envisioned strategy involves the incorporation of a PID control technique in tandem with the pure pursuit model. This entails the implementation of a PID Controller governing both linear and rotational velocities, acting as an auxiliary trajectory controller. The strategic merger aims to alleviate challenges associated with surpassing set goals and minimize encounters with navigational barriers. By exploring various PID parameter configurations and

predictive conditions, the study seeks to identify an optimal setup, enhancing AGV maneuverability in intricate and confined environments.

Evaluation of the developed framework encompasses both simulation-based and physical platform experiments. These assessments involve comparisons among local planners, specifically Pure Pursuit, Dynamic Window Approach (DWA), and Time Elastic Band (TEB) models. The amalgamation of these local planners with A* and Dijkstra algorithms operating on a low-resolution grid-based system enables a comprehensive assessment of navigational efficacy.

Simulation tests underscored the critical role of adjusting look-ahead, linear, and angular velocities in minimizing time and distance to goal achievement, alongside reducing orientation errors. In contrast, real platform studies provided essential insights into steering parameter optimization within complex landscapes, shedding light on the most effective parameters for real-world navigational challenges.

This research contributes significantly to advancing the understanding of AGV navigation in intricate environments. The strategic combination of PID control techniques, local planners, and algorithms operating on a low-resolution grid offers a promising approach to enhance AGV navigational precision and efficiency in complex real-world scenarios. The findings and methodologies presented here serve as a foundational framework for future research endeavors aimed at further improving AGV navigation and autonomy.

Building upon the findings and insights gleaned from this study's comprehensive historical analysis and technical experimentation, several recommendations emerge to enhance the navigational capabilities and efficiency of AGVs in complex environments.

## 6.1 Implement Low-Resolution Mapping Techniques

The application of low-resolution mapping techniques has exhibited promise in addressing the issue of overshooting when combining A* with pure pursuit algorithms. It is recommended to refine and expand this methodology by employing advanced mapping techniques that enable more precise dot placement. Furthermore, exploring adaptive strategies to dynamically adjust the resolution of the grid based on environmental complexities could potentially optimize navigation.

## 6.2 Optimize PID Control Techniques

The incorporation of PID control techniques in tandem with pure pursuit demonstrates substantial potential in mitigating navigation issues related to surpassing set goals and encountering barriers. To maximize effectiveness, it is advised to conduct rigorous parameter tuning and performance assessments, leveraging iterative simulation-based analyses, to pinpoint the optimal PID configurations for different operational scenarios.

## 6.3 Enhance Integration of Local Planners and Algorithms

The comparison and evaluation of various local planners such as Pure Pursuit, DWA, and TEB models, combined with A* and Dijkstra algorithms in a low-resolution grid-based system, highlight the need for further exploration. There is an opportunity to delve deeper into the synergistic integration of these planners and algorithms, possibly through adaptive hybridization approaches that harness the strengths of each method for specific navigation challenges.

## 6.4 Real-World Validation and Experimentation

Given the promising results observed in simulations, it is imperative to validate and further refine these findings through rigorous real-world experimentation. Practical testing in varied and challenging environments will offer invaluable insights into the scalability, robustness, and adaptability of the proposed navigation system. This includes addressing real-world constraints, sensor noise, and dynamic environmental changes to validate the system's performance and reliability.

## 6.5 Continuous Iterative Development and Collaboration

To ensure the continual advancement of AGV navigation, a collaborative approach among researchers, industry experts, and stakeholders is recommended. Encouraging collaboration and knowledge exchange can foster innovation and facilitate the implementation of cutting-edge technologies. Embracing an iterative development approach enables continual refinement, adaptation, and enhancement of AGV navigation systems.

## 6.6 Ethical and Safety Considerations

As AGVs become more prevalent in industrial and public settings, it is crucial to prioritize safety and ethical considerations. Robust safety protocols, fail-safes, and ethical guidelines must be integrated into the design and deployment of AGV navigation systems to prevent accidents and ensure responsible use in shared spaces.

## 6.7 Future Directions and Research Avenues

To propel the field of AGV navigation forward, it is essential to explore emerging technologies, such as machine learning, artificial intelligence, and sensor fusion. Further investigations into novel algorithms, adaptive control strategies, and human-robot interaction paradigms can open new frontiers in AGV navigation efficiency and adaptability.

The recommendations outlined above offer a roadmap for the continual enhancement and refinement of AGV navigation systems. By combining innovative technological approaches with collaborative research efforts and a commitment to safety and ethical considerations, the trajectory of AGVs in intricate environments can be further optimized, ultimately revolutionizing their role in various industries.

# REFERENCES


[1] Y. Chen, S. Shi, Z. Chen, T. Wang, L. Miao, H. Song, "Optimizing port multi-agv trajectory planning through priority coordination: Enhancing efficiency and safety," *Axioms*, vol. 12, no. 9, 2023, issn: 2075-1680.

[2] G. Fragapane, R. de Koster, F. Sgarbossa, J. O. Strandhagen, "Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda," *European Journal of Operational Research*, vol. 294, no. 2, pp. 405–426, 2021, issn: 0377-2217.

[3] R. Van Parys *et al.*, "Distributed coordination, transportation & localisation in industry 4.0," in *2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2018, pp. 1–8.

[4] X. Li, Z. Sun, D. Cao, D. Liu, H. He, "Development of a new integrated local trajectory planning and tracking control framework for autonomous ground vehicles," *Mechanical Systems and Signal Processing*, vol. 87, pp. 118–137, 2017, Signal Processing and Control challenges for Smart Vehicles, issn: 0888-3270.

[5] M. A. Javed, F. U. Muram, S. Punnekkat, H. Hansson, "Safe and secure platooning of automated guided vehicles in industry 4.0," *Journal of Systems Architecture*, vol. 121, p. 102 309, 2021, issn: 1383-7621.

[6] M. Boehning, "Improving safety and efficiency of agvs at warehouse black spots," in *2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing (ICCP)*, 2014, pp. 245–249.

[7] *Industrial Workplace Safety for Automated Guided Vehicles*.

[8] D. Bibbo *et al.*, "Simulation and validation of optimized pid controller in agv (automated guided vehicles) model using pso and bas algorithms," *Journal Title*, vol. 2022, p. 7 799 654, 2022.

[9] L. F. Rocha, A. P. Moreira, A. Azevedo, "Flexible internal logistics based on agv system's: A case study," *IFAC Proceedings Volumes*, vol. 43, no. 17, pp. 248–255, 2010, 5th IFAC Conference on Management and Control of Production Logistics, issn: 1474-6670.

[10] B. Malone. "Devol: A life devoted to invention, and robots." [Accessed: Dec. 27, 2023]. (Mar. 2023).

[11] H. Neradilová, G. Fedorko, *Open Engineering*, vol. 6, no. 1, 2016.

[12] D. Speck, C. Dornhege, W. Burgard, "Shakey 2016—how much does it take to redo shakey the robot?" *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1203–1209, 2017.

[13] U. Hanebeck, N. Saldic, G. Schmidt, "A modular wheel system for mobile robot applications," in *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No.99CH36289)*, vol. 1, 1999, 17–22 vol.1.

[14] "Ieee transactions on robotics information for authors," *IEEE Transactions on Robotics*, vol. 21, no. 1, pp. c4–c4, 2005.

[15] "Ieee transactions on autonomous mental development information for authors," *IEEE Transactions on Autonomous Mental Development*, vol. 3, no. 1, pp. C4–C4, 2011.

[16] F. Lamnabhi-Lagarrigue *et al.*, "Systems & control for the future of humanity, research agenda: Current and future roles, impact and grand challenges," *Annual Reviews in Control*, vol. 43, pp. 1–64, 2017, issn: 1367-5788.

[17] U. Hanebeck, N. Saldic, G. Schmidt, "A modular wheel system for mobile robot applications," in *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No.99CH36289)*, vol. 1, 1999, 17–22 vol.1.

[18] N. Hassan, A. Saleem, "Neural network-based adaptive controller for trajectory tracking of wheeled mobile robots," *IEEE Access*, vol. 10, pp. 13 582–13 597, 2022.

[19] *Autonomous Distributed AGV System Based on Taxi Transportation Strategy: Effect of Multiple-Load AGVs on Conveyance Efficiency*, vol. ASME 2010 International Manufacturing Science and Engineering Conference, Volume 2, International Manufacturing Science and Engineering Conference, Oct. 2010, pp. 553–560.

[20] A. Sudmann, *The Democratization of Artificial Intelligence: Net Politics in the Era of Learning Algorithms* (AI critique). Transcript, 2019, isbn: 9783837647198.

[21] J. Bayer, P. Čížek, J. Faigl, "Autonomous multi-robot exploration with ground vehicles in darpa subterranean challenge finals," *Field Robotics*, vol. 3, pp. 266–300, Jan. 2023.

[22] J. Chen, X. Zhang, X. Peng, D. Xu, J. Peng, "Efficient routing for multi-agv based on optimized ant-agent," *Computers & Industrial Engineering*, vol. 167, p. 108 042, 2022, issn: 0360-8352.

[23] J. Meng, A. Liu, Y. Yang, Z. Wu, Q. Xu, "Two-wheeled robot platform based on pid control," in *2018 5th International Conference on Information Science and Control Engineering (ICISCE)*, 2018, pp. 1011–1014.

[24] S. Koo, K. Jo, "Performance improvement of pure persuit alogorithm using pid control," in *2023 International Workshop on Intelligent Systems (IWIS)*, 2023, pp. 1–2.

[25] Y. Huang, Z. Tian, Q. Jiang, J. Xu, "Path tracking based on improved pure pursuit model and pid," in *2020 IEEE 2nd International Conference on Civil Aviation Safety and Information Technology (ICCASIT*, 2020, pp. 359–364.

[26] M. K. Diab, H. H. Ammar, R. E. Shalaby, "Self-driving car lane-keeping assist using pid and pure pursuit control," in *2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT)*, 2020, pp. 1–6.

[27] Y. Chen, Y. Shan, L. Chen, K. Huang, D. Cao, "Optimization of pure pursuit controller based on pid controller and low-pass filter," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 3294–3299.

[28] S. Macenski, S. Singh, F. Martín, J. Ginés, "Regulated pure pursuit for robot path tracking," *Autonomous Robots*, pp. 1–10, 2023.

[29] E. Horváth, C. Hajdu, P. Kőrös, "Novel pure-pursuit trajectory following approaches and their practical applications," in *2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, 2019, pp. 000 597–000 602.

[30] H. Bin, L. W. Zhen, L. H. Feng, "The kinematics model of a two-wheeled self-balancing autonomous mobile robot and its simulation," in *2010 Second International Conference on Computer Engineering and Applications*, vol. 2, 2010, pp. 64–68.

[31] E. Maulana, M. A. Muslim, A. Zainuri, "Inverse kinematics of a two-wheeled differential drive an autonomous mobile robot," in *2014 Electrical Power, Electronics, Communicatons, Control and Informatics Seminar (EECCIS)*, 2014, pp. 93–98.

[32] E. Savaee, A. Rahmani, Y. Anabestani, "Kinematic analysis and odometry-based navigation of an omnidirectional wheeled mobile robot on uneven surfaces," *Journal of Intelligent & Robotic Systems*, vol. 108, May 2023.

[33] Z. Chi, Z. Yu, Q. Wei, Q. He, G. Li, S. Ding, "High-efficiency navigation of nonholonomic mobile robots based on improved hybrid a* algorithm," *Applied Sciences*, vol. 13, no. 10, 2023, issn: 2076-3417.

[34] Mecharithm. "Holonomic vs. nonholonomic constraints for robots | mecharithm." Accessed: Dec. 28, 2023. (2023).

[35] M. B. Alatise, G. P. Hancke, "A review on challenges of autonomous mobile robot and sensor fusion methods," *IEEE Access*, vol. 8, pp. 39 830–39 846, 2020.

[36] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, eabm6074, 2022.

[37] Y. Abdelrasoul, A. B. S. H. Saman, P. Sebastian, "A quantitative study of tuning ros gmapping parameters and their effect on performing indoor 2d slam," in *2016 2nd IEEE International Symposium on Robotics and Manufacturing Automation (ROMA)*, 2016, pp. 1–6.

[38] S. Kumar NT, M. Gawande, N. P. B. M, H. Verma, P. Rajalakshmi, "Mobile robot terrain mapping for path planning using karto slam and gmapping technique," in *2022 IEEE Global Conference on Computing, Power and Communication Technologies (GlobConPT)*, 2022, pp. 1–4.

[39] B. Balasuriya *et al.*, “Outdoor robot navigation using gmapping based slam algorithm,” in *2016 Moratuwa Engineering Research Conference (MERCon)*, 2016, pp. 403–408.

[40] S. Romaniuk, A. Wolniakowski, A. Pawłowski, C. Kownacki, “Adaptation of ultra wide band positioning system for adaptive monte carlo localization,” in *2022 26th International Conference on Methods and Models in Automation and Robotics (MMAR)*, 2022, pp. 238–243.

[41] X. Tian, B. Li, Y. Du, M. Wang, “Robot autonomous exploration and map building method,” in *2023 International Conference on Robotics and Automation in Industry (ICRAI)*, 2023, pp. 1–6.

[42] A. M. Kaneko, H. Hisatsugu, J. Hashizume, “Stata center frame monte carlo localization: A new global self-localization method based on world assumptions,” in *2023 IEEE/SICE International Symposium on System Integration (SII)*, 2023, pp. 1–8.

[43] C. Rösmann, F. Hoffmann, T. Bertram, “Planning of multiple robot trajectories in distinctive topologies,” in *2015 European Conference on Mobile Robots (ECMR)*, 2015, pp. 1–6.

[44] Y. Liu, Y. Du, S. Dou, L. Peng, X. Su, “Research summary of intelligent optimization algorithm for warehouse agv path planning,” in *LISS 2021*, X. Shi, G. Bohács, Y. Ma, D. Gong, X. Shang, Eds., Singapore: Springer Nature Singapore, 2022, pp. 96–110.

[45] A. Al-Naseri, E. Uslu, “Autonomous mobile robot navigation using lower resolution grids and pid-based pure pursuit controller,” in *Advances in Intelligent Manufacturing and Service System Informatics*, Z. Şen, Ö. Uygun, C. Erden, Eds., Singapore: Springer Nature Singapore, 2024, "200–213".

[46] B. Fu *et al.*, “An improved a* algorithm for the industrial robot path planning with high success rate and short length,” *Robotics and Autonomous Systems*, vol. 106, pp. 26–37, 2018, issn: 0921-8890.

[47] B. J. Oommen, L. G. Rueda, “A formal analysis of why heuristic functions work,” *Artificial Intelligence*, vol. 164, no. 1, pp. 1–22, 2005, issn: 0004-3702.

[48] S. Ramane, P. Shah, N. Doshi, A. Madankar, B. Narkhede, “Hardware and software development of a small scale driverless vehicle,” in *2022 IEEE Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI)*, 2022, pp. 1–6.

[49] M. Jaimez, J. G. Monroy, J. Gonzalez-Jimenez, “Planar odometry from a radial laser scanner. a range flow-based approach,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 4479–4485.

# PUBLICATIONS FROM THE THESIS

## Conference Papers

1. Al-Naseri, A., Uslu, E. (2024). Autonomous Mobile Robot Navigation Using Lower Resolution Grids and PID-Based Pure Pursuit Controller. In: Şen, Z., Uygun, Ö., Erden, C. (eds) Advances in Intelligent Manufacturing and Service System Informatics. IMSS 2023. Lecture Notes in Mechanical Engineering. Springer, Singapore.