People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

University of August 20, 1955 Skikda Faculty of Technology
Department of Electrical Engineering

Ref : D012123029D

THESIS
Submitted for the award of the Doctorate degree in Automation

By Mr. Mouad Boumediene

Contribution to Mobile Robots Command

Defended on: 19 - Dec - 2023

Before the Jury composed of:

President	Pr. GUECHI El-Hadi	Pr	University of 20 août 1955, skikda
Rapporteur	Dr. MEHENNAOUI Lamine	MCA	University of 20 août 1955, skikda
Examiner	Pr. GHERBI Sofiane	Pr	University of Badji Mokhtar, Annaba
Examiner	Dr. BOUCHAREB Faouzi	MCA	University of Badji Mokhtar, Annaba
Examiner	Pr. INEL Fouad	Pr	University of 20 août 1955, skikda
Guest	Pr. LACHOURI Abderezzak	Pr	University of 20 août 1955, skikda

Supervised by:
Dr. Lamine Mehennaoui , University of 20 août 1955, skikda

ملخص :

تتعمق هذه الأطروحة في المجال الرائع للروبوتات المتنقلة المستقلة(AMRs)، وهي مستوحاة من التقدم الكبير في الصناعة. يركز هذا البحث على تعزيز تقنيات الملاحة المستقلة لـ AMRs مع التركيز على التخطيط الفعال للمسارات في البيئات الصناعية. هنا تظهر مساهمتان محوريتان: طريقة هجينة لتخطيط المسار التي تجمع بين وقت الحساب السريع للخوارزميات العشوائية مع الدقة والمثالية في الأساليب القائمة على الشبكة، و أيضا تم تصميم سير عمل مبتكر وفعال من حيث التكلفة لتصميم وبناء الروبوتات المتنقلة المستقلة. وباستخدام التصميم بمساعدة الكمبيوتر، عملت الدراسة على تحسين تصميم الروبوت ذاتي التوازن، والذي تم التحقق من عمله في محاكاة باستعمال برنامجي ROS وGazebo ، وتم إنشاؤه باستخدام مكونات متاحة على نطاق واسع. ويسعى هذا العمل إلى توسيع نطاق تطبيقات الروبوتات، مما يجعلها مجدية لكل من الشركات الصغيرة والأسر.

Abstract :

This dissertation delves into the fascinating domain of autonomous mobile robots (AMRs), inspired by significant industry advancements. This research centers on enhancing AMRs' autonomous navigation techniques with a focus on efficient path planning in industrial environments. Two pivotal contributions emerge: a hybrid path-planning method combining the fast computation time of stochastic algorithms with accuracy and optimality of grid-based approaches, and an innovative, cost-effective workflow for AMR design and construction . Utilizing CAD, the study refines a self-balancing robot design, validated in a ROS + Gazebo simulation, and constructed with widely available components. This work endeavors to broaden AMR applicability, making it feasible for both small-scale businesses and households.

Résumé :

Cette thèse explore le domaine fascinant des robots mobiles autonomes (AMR), inspiré par des avancées significatives de l'industrie. Cette recherche se concentre sur l'amélioration des techniques de navigation autonomes des AMR en mettant l'accent sur une planification efficace des trajectoires dans les environnements industriels. Deux contributions cruciales émergent : une méthode hybride de planification de chemin combinant le temps de calcul rapide des algorithmes stochastiques avec la précision et l'optimalité des approches basées sur une grille, et un flux de travail innovant et rentable pour la conception et la construction d'AMR. À l'aide de la CAO(Conception Assistée par Ordinateur), l'étude affine une conception de robot auto-équilibré, validée dans une simulation ROS (Robot Operating system) + Gazebo et construite avec des composants largement disponibles. Ce travail vise à élargir l'applicabilité de la RAM, la rendant réalisable à la fois pour les petites entreprises et les ménages.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to express my deepest gratitude to God Almighty, for the strength, wisdom, and guidance bestowed upon me, making this journey possible.

My heartfelt thanks to my parents, whose love, support, and sacrifices have been the cornerstone of my achievements. Their unwavering belief in me has been my constant source of motivation and resilience.

I am deeply grateful to my supervisor, for his invaluable guidance, patience, and expert advice throughout my research. His mentorship has been crucial in shaping my work and this achievement.

I extend my sincere appreciation to my family members for their endless encouragement and understanding throughout the duration of my studies. Their support has been invaluable.

A special thanks to my teachers, from elementary school through to university. Each one of them has contributed to my academic growth and personal development, shaping me into the person I am today.

Finally, I would like to acknowledge my friends who have been with me on this journey. Their companionship, support, and shared moments of joy and struggle have enriched my PhD experience immeasurably.

This achievement is not just mine, but a reflection of the collective effort and sacrifice of all those mentioned above.

Nomenclature

- A* - A Star
- AB - Adaptive Boosting
- AGV - Automated Guided Vehicle
- AI - Artificial Intelligence
- AOA - Angle Of Arrival
- AR - Augmented Reality
- BIT* - Batch Informed Trees
- BMW - Bayerische Motoren Werke (Bavarian Motor Works)
- BRISK - Binary Robust Invariant Scalable Keypoints
- BRIEF - Binary Robust Independent Elementary Features
- CAD - Computer-Aided Design
- CANON - (Likely specific to the context of the document)
- CNC - Computer Numerical Control
- CNN - Convolutional Neural Network
- CVOG - (Likely specific to the context of the document)
- D* - D Star
- EKF - Extended Kalman Filter
- EOS - Electro-Optical System
- FAST - Features from Accelerated Segment Test
- FDA* - Focused Discretization A*
- FMT* - Fast Marching Tree

- FREAK - Fast Retina Keypoint
- GLONASS - Global Navigation Satellite System
- GNSS - Global Navigation Satellite System
- GPS - Global Positioning System
- GTSAM - Georgia Tech Smoothing and Mapping
- HMI - Human-Machine Interface
- ICR - Instantaneous Center of Rotation
- IDA - Iterative Deepening A*
- IMU - Inertial Measurement Unit
- JPS - Jump Point Search
- LIDAR - Light Detection and Ranging
- LOAM - Lightweight and Ground-Optimized Lidar Odometry and Mapping
- LPA* - Lifelong Planning A*
- LQ - Linear Quadratic
- LS - Least Squares
- LS-NET - Learning to Solve Nonlinear Least Squares for Monocular Stereo
- LSD-SLAM - Line Segment Detector Simultaneous Localization and Mapping
- LSTM - Long Short-Term Memory
- MPC - Model Predictive Control
- ORB - Oriented FAST and Rotated BRIEF
- PCB - Printed Circuit Board
- PF - Particle Filter
- PID - Proportional-Integral-Derivative
- PRM - Probabilistic Roadmap Method
- PTAM - Parallel Tracking and Mapping
- RANSAC - Random Sample Consensus
- RCB - Robot Controller Board

- ROS - Robot Operating System
- RRT - Rapidly-exploring Random Tree
- RRT* - RRT Star
- SLAM - Simultaneous Localization and Mapping
- SMA - Simplified Memory-bounded A*
- SDF - Simulation Description Format
- STOMP - Stochastic Trajectory Optimization for Motion Planning
- TDOA - Time Delay of Arrival
- UAV - Unmanned Aerial Vehicle
- UGV - Unmanned Guided Vehicle

Chapter 1

introduction

Contents

1.1 Thesis introduction

The progressive transformation of industries towards automation is indisputable, and it is not just a trend but an inevitable trajectory that industries worldwide are embracing. Be it the automobile sector with automated assembly lines, agriculture with self-driving harvesters, or logistics with automated warehouses; automation is becoming a core component of their operational strategy. This transformation is propelled not only by the desire to enhance operational efficiency—automated systems can operate around the clock without fatigue, and with a high degree of precision—but also by the quest to maintain competitiveness in a rapidly evolving technological environment.

A central player in this industrial metamorphosis is the .AMRs have risen to prominence due to their versatility and autonomy. As opposed to their predecessors, the Automated Guided Vehicles (AGVs) that followed predefined paths, AMRs can independently navigate complex environments. They have the capacity to adapt to dynamic environments, avoid obstacles, and plan efficient paths, thus making them invaluable assets for industries.

For instance, in an Amazon warehouse, AMRs like 'Kiva' [14] are changing the traditional dynamics of the warehouse processes. These robots can move around the warehouse, pick up shelves, and bring them to the human pickers, significantly reducing the time and effort spent by humans in walking around the warehouse. In the manufacturing industry, companies like BMW [15] and General Motors [16] are using AMRs in their manufacturing process, enhancing efficiency and safety.

However, to achieve their maximum potential, AMRs must confront and resolve some challenging problems related to path planning, localization, and mapping. Path planning is crucial in environments like factories or warehouses, where the robot needs to navigate from one point to another efficiently and safely while avoiding dynamic and static obstacles.

Localization, the process of knowing the robot's position in the environment, is another critical aspect. Coupled with mapping, where the robot builds a map of the environment, these capabilities allow the AMR to understand its surroundings and navigate effectively.

Despite the advancement of technology, these problems remain complex due to the dynamic nature of industrial environments. Unexpected obstacles, changes in the environment layout, or even variations in lighting and acoustic conditions can challenge an AMR's capabilities.

This dissertation explores these fundamental challenges and presents novel solutions to improve AMR efficiency and effectiveness in industrial environments. By delving deep into the problems, we strive to bring forth practical and impactful solutions, further paving the way for a more comprehensive and effective adoption of AMRs across industries.

The Autonomous Mobile Robotics field, combining robotics, artificial intelligence, and sensor technology, has experienced remarkable growth over the past decade [17]. This trend, fueled by the digital economy and the demand for advanced automation solutions, shows no signs of slowing down in the upcoming years. Many of these robots are expected to serve in a variety of sectors, including agriculture, healthcare, wildlife protection, and exploration, along with logistics in warehouses, distribution centers, and factories [18–23] . Their tasks range from package delivery to the inspection and monitoring of high-value industrial machinery.

Key functionalities of these AMRs include autonomous navigation in partially or fully unknown environments while successfully avoiding static and dynamic obstacles. The performance of these AMRs relies heavily on their sensor technologies and underlying algorithms. Advancements in low-cost sensor technologies, including both intrinsic (wheel encoder, IMUs, etc.) and extrinsic perceptual sensors (cameras, radars, GPS, LIDARs, etc.), have broadened the practical applications of AMRs. This, in turn, has piqued the interest of researchers in developing advanced navigation algorithms for these autonomous machines.

A wealth of industries, from agriculture to healthcare, from wildlife protection to exploration, are recognizing and harnessing the potential of AMRs. For example, in agriculture, AMRs like the Harvest CROO Robotics system [24], autonomously pick strawberries, while in the medical field, robots such as the TUG by Aethon [25] deliver medications and supplies within hospitals. In wildlife conservation, there are drones monitoring wildlife populations and detecting threats like illegal hunting or habitat destruction. AMRs are also indispensable in warehouses, distribution centers, and factories, with robots like the MiR series from Mobile Industrial Robots [26] executing a range of tasks from package delivery to inspection and monitoring of expensive industrial machinery.

One of the core functionalities of an AMR is autonomous navigation, the ability to move independently in partially known or entirely unknown environments while avoiding obstacles. This capability is critical in a variety of real-world scenarios, like a factory robot navigating around workers and equipment, or a delivery drone avoiding trees and buildings in a city. Within autonomous navigation, a critical component is path planning, essentially the process by which a robot determines the optimal route from a start point to a goal point. It's the cognitive map-making of the robot's world, forming the plan that dictates its movements.

The effectiveness of an AMR's autonomous navigation hinges on two key elements: sensors and algorithms. On one hand, sensors act as the AMR's eyes and ears, providing the necessary data to make informed decisions. Thanks to technological advancements, a wide array of low-cost yet efficient sensors are now available. Intrinsic sensors, such as wheel encoders and

Inertial Measurement Units (IMUs), provide data about the robot's own state, like its speed or orientation.

On the other hand, extrinsic perceptual sensors, like cameras, radars, GPS, and Light Detection and Ranging (LIDAR) systems, sense the robot's environment. For instance, in the context of path planning, LIDAR sensors and cameras are used to detect obstacles and construct an environmental map that allows AMRs to compute safe and efficient routes. The Spot robot by Boston Dynamics [27] uses a 360-degree LIDAR system for navigation and obstacle avoidance, while the Skydio R1 drone [28] utilizes 13 cameras to construct a real-time 3D map of its surroundings, playing a critical role in dynamic path planning.

The availability of such sensors has sparked a surge in practical applications of AMRs. The Kiva robots in Amazon's warehouses, for instance, use barcode stickers on the ground and onboard cameras for localization and navigation. In turn, this surge in applications has stimulated researchers' interest in developing increasingly efficient AMR navigation algorithms, which are crucial for efficient path planning. These algorithms interpret sensor data, take into account the robot's constraints and task requirements, and generate the optimal paths for navigation.

The tremendous strides in the Autonomous Mobile Robotics field and the exciting promise it holds for the future are unequivocal. This dissertation delves into the challenges and potentials of this dynamic field, specifically in the context of sensor technology, path planning, and navigation algorithm development, aiming to contribute towards the further advancement of AMRs.

In Chapter 2, the focus begins with an extensive survey of existing literature on Autonomous Mobile Robots (AMRs) and their diverse applications. This is followed by an in-depth examination of the methods employed in localization, mapping, and path planning. The chapter then shifts to an analysis of wheeled AMR models, offering insights into their design and operational mechanisms. Additionally, the capabilities of AMRs in perception and autonomous navigation are thoroughly explored. The chapter also sheds light on the driving factors of this dissertation and clearly outlines the unique contributions of this study.

Chapter 3 is dedicated to the design, construction, and testing of affordable high-performance mobile robots, using a real-life example of a self-balancing robot. we provide also a full workflow of mobile robot building passing by all the important stages of this procedure.

Chapter 4 is devoted to path planning for AMRs. We first introduce the problem and then dive deep into discrete and sampling-based path planning methodologies. We discuss in detail several important algorithms such as Rapidly-exploring Random Trees (RRT), Probabilistic Roadmaps (PRM), and Fast Marching Trees with Differential Constraints (FDA*). We conclude this chapter with an evaluation of the discussed strategies.

Chapter 5 provides a general conclusion, synthesizing the insights from the previous chapters. It revisits the key contributions presented in this dissertation, reflecting on the advancements made in the field of AMRs. This chapter also discusses perspectives on future developments and potential areas for further exploration in the realm of autonomous mobile robotics.

1.2 Motivation

In 2003, a former Webvans Logistics employee named Mick Mountz founded Kiva System, a robotics company that focused on improving efficiency and flexibility within the logistics and packaging world using swarms of mobile robots to transport packages within the warehouse. At that time kiva's AGV(Autonomous Guided Vehicle) used a guidance system that depended on 2D floor-mounted tags to localize themselves within the warehouse.
Due to this New solution for logistics, Kiva saw explosive growth. In 2009, Kiva Systems was the sixth fastest-growing company in the United States, with its rate of growth far outpacing any other of its competitors in the industry. In 2012, the American multinational e-commerce company Amazon, made a huge investment to enhance their productivity in the supply chain by acquiring Kiva Systems for 775 million dollars. By 2015, 15_000 robots based on the original Kiva design were working in Amazon's warehouses now they have more than 520_000 robotic drive units creating an unprecedented increase in efficiency, productivity, and safety in the e-commerce business.
In January 2022 amazon announced Proteus, its first 'fully autonomous' warehouse robot. Proteus does not have to be confined for safety within a caged area like its predecessors; it is fully autonomous and is able to avoid dynamic and static obstacles such as humans or other robots, using advanced perception and navigation technology. Motivated by the remarkable advancement in logistics management using AMRs, this dissertation focuses on developing and enhancing autonomous navigation techniques used in AMRs, by proposing an efficient solution for Path planning in industrial environments using fewer computational resources. in addition to this, we will provide a full mobile robot design and construction workflow using a real-life example of a two-wheeled self-balancing robot This will improve the navigation process, at the same time reduce the cost of AMRs, and thus increasing their application range to include small retail businesses and even households.

1.3 Contributions of the Dissertation

This dissertation focuses on developing Autonomous Navigation for mobile robots, primarily enhancing its Path Planning approach while also contributing to the design and construction workflow. The central aim is to devise algorithms suitable for lower-end hardware, broadening the application range of Autonomous Mobile Robots (AMRs).

The following are the dissertation's contributions, categorized in order of their significance:

1. The major contribution is the development of a hybrid path-planning method. Traditional square grid-based path planning, though optimal, is resource-intensive in large-scale or high-dimensional scenarios. This work introduces a hybrid approach that combines stochastic path-planning algorithms with grid-based methods. The stochastic component incrementally discretizes the state space, significantly reducing the computational load and memory requirements for grid-based path planning. This innovation represents a major leap in computational efficiency, especially in complex navigation tasks.

2. A secondary contribution, is the advancement in the design and construction workflow for AMRs. The focus here is on improving the simplicity and cost-effectiveness of building self-balancing robots. Starting with an enhanced CAD design and the introduction of a Robot Controller Board (RCB), the workflow streamlines efficiency and simplifies troubleshooting. A realistic simulation environment using ROS+Gazebo facilitates the development of the control system before real-world implementation. The manufacturing process employs CNC machining to create individual parts that assemble via friction, eliminating the need for adhesives or fasteners. By using affordable and widely available components, this approach not only reduces costs but also expands the accessibility and application spectrum of AMRs.

These contributions collectively reduce the time and effort needed to build mobile robots and minimize the resources required for autonomous navigation, thus expanding the use cases of AMRs.

Chapter 2

Background

Contents

2.1 introduction

In this comprehensive chapter, we begin by presenting a thorough literature review that delves into the dynamic field of Autonomous Mobile Robots (AMRs). This review serves as a foundational backdrop, offering insights into the significant advancements and evolution in the realm of path planning and control algorithms within mobile robotics. We trace the

journey from the initial graph-based algorithms like Dijkstra and A*, through their transformative impact, to the advent of more sophisticated methodologies like the Potential Field Method, PRM, and RRT. This retrospective analysis not only charts the historical progress in the field but also highlights the computational and practical challenges encountered, setting the stage for understanding the subsequent innovations and adaptations.

Building on this contextual framework, the chapter then transitions into exploring the generalities of AMRs. The narrative takes the reader on an exploratory journey through the intricate and rapidly evolving domain of these robots, linking historical fascination with 'artificial creatures' evident in ancient Greek mythology and the inventions of Ismael al-Jazari to modern advancements. The discussion extends to the classification of robots based on locomotion, environment, application, and size, illustrating the vast diversity of AMRs designed for various sectors such as healthcare, military, and logistics.

Central to this section is an examination of the crucial role of control algorithms in executing navigation plans. These algorithms are pivotal in orchestrating the efficient and safe movement of AMRs, adapting to the complexities of various environments. The chapter emphasizes the importance of safety in both industrial and residential applications, highlighting the challenges posed by different terrains and the integration of sensors and machine learning for adaptive navigation strategies.

Moreover, the chapter delves into the theoretical underpinnings of autonomous navigation, addressing both indoor and outdoor challenges, and the role of advanced sensors like Lidars, IMUs, cameras, and GPS in the navigation process. Special attention is given to wheeled AMRs, particularly dominant in industrial and indoor settings, discussing their energy efficiency, speed, stability, and the types of wheels employed.

In essence, this chapter blends a rich literature review with an insightful exploration of AMRs' generalities, weaving together historical context, technological progress, and the future trajectory of robotics. It presents a nuanced understanding of the complexities and challenges in the field, as well as the innovative strategies and solutions that are driving the future of autonomous mobile robotics.

2.2 Literature Review

Early work on autonomous navigation included reactive robotics with no complex processing steps between the sensors and the actuators, such as the Grey Walter's tortoise elsie [29], which could successfully follow a light source using an onboard rotating photocell, or the Brightenberg vehicles [30] that mimics animal behavior of moving towards or escaping from a stimulus. Bug algorithms [31] were proposed as a slightly more advanced approach for indoor navigation where the robot moves along the boundaries of obstacles and walls, knowing only the relative position of its goal.
Achieving a more complex behavior requires some sort of depiction of the robot's environment [30], a map presentation that could be used for localizing the robot, and also for planning collision-free paths between its start and goal positions. However, building such a map from scratch requires position estimation, which makes these two problems(localization and mapping) highly interdepended [32]. And usually treated as a single problem called SLAM (Simultaneous localization and mapping).

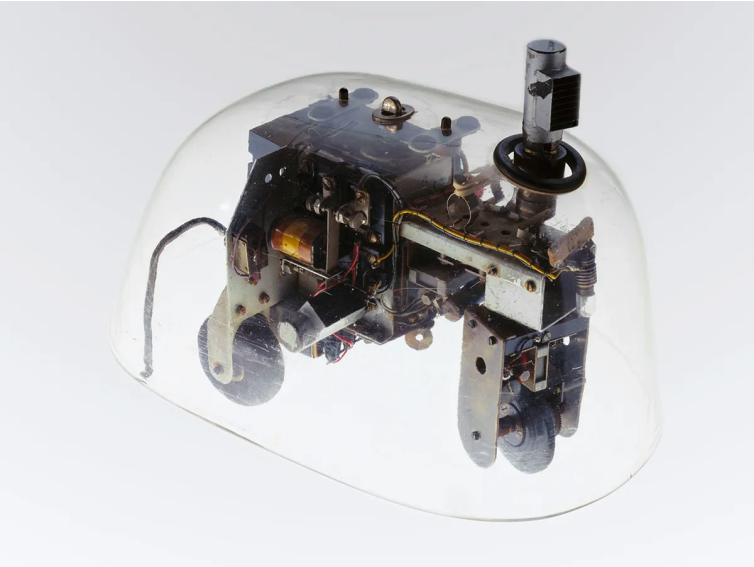

Figure 2.1: The Grey Walter's tortoise Elsie [1], an early example of autonomous navigation. Elsie could successfully follow a light source using an onboard rotating photocell.

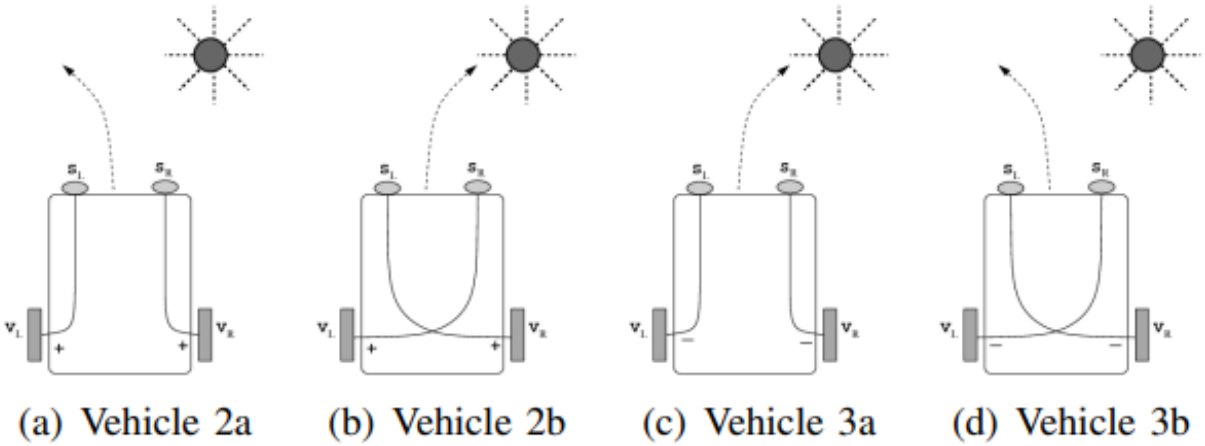

Figure 2.2: Schematics of Braitenberg Vehicles 2 and 3 [2].

2.2.1 Autonomous mobile robot design

With significant advancements in design, functionality, and applicability, the development of autonomous mobile robots (AMRs) has drawn significant interest in the field of robotics. In order to clarify the transformative journey from their inception to current accomplishments, this discourse undertakes a thorough retrospective analysis of the developmental trajectory of AMRs.

- *Early Years (1950s–1970s)*: The 1950s saw the advent of primitive robotic endeavors that were firmly rooted in computer technology. During these formative years, we saw early attempts at autonomous behavior as well as crude remote-control systems. The "Shakey" robot [33], created at Stanford Research Institute in the 1960s and showing basic navigation and problem-solving abilities, was a key development of this time period. These early robots, with their simple wheel-based mobility and simple obstacle avoidance sensors, embodied the first steps in the development of AMR.

- *Enhancements to Navigation and Industrial Uptake (1980s–1990s):* The development of sophisticated navigational and localization methods in the 1980s and 1990s led to fundamental design changes in a variety of robotic platforms. The Nomad and Pioneer series of wheeled AMRs, in particular, used sophisticated sensor arrays and computational power to navigate autonomously in changing environments, with uses in warehousing, logistics, and manufacturing. Legged robots like the "Genghis" series made advancements toward increased mobility and adaptability concurrently.

- *Flexible Design Configurations for Various Applications (2000-2010):* AMR design underwent a paradigm shift in the early 21st century as it worked to meet the requirements of various applications while incorporating a variety of robotic modalities. Drones and other aerial robots became popular, offering cutting-edge design paradigms with their quadcopters and hexacopters. Applications in agriculture, environmental monitoring, and aerial reconnaissance were made easier by these designs. Additionally, underwater robots adopted designs suited for aquatic exploration and data collection, as demonstrated by autonomous underwater vehicles (AUVs) like the "OceanInfinity" robots [34].

The design configurations' modularity further broadened the possibilities for AMR deployment. Compact drones were used for tasks like aerial photography and search and rescue missions, while smaller wheeled robots like the iRobot Roomba [35] found their place in domestic cleaning. Similar to the "ANYmal" series [36], modular legged robots demonstrated adaptability in a variety of terrains, including harsh and unorganized environments. Modern AMR designs embrace the integration of cutting-edge cognitive capabilities driven by AI across all modalities. These robots are now capable of sensory perception, object recognition, and interactive engagement in addition to autonomous navigation. The da Vinci Surgical System [37] is a prime example of medical robotics, which highlights the profound synergy between cutting-edge design and healthcare applications, where exact surgical maneuvers and human-robot cooperation are crucial. The design principles for energy efficiency, adaptability, and human-robot collaboration are combined in modern AMRs across all modalities. Design efficiency is clearly prioritized, in line with how industries are changing and what they need. The paradigm shift brought about by swarm robotics and multi-agent systems portends a time when heterogeneous robots will work together to solve problems in fields as diverse as environmental preservation and disaster response.

Overall, the development of AMRs in various robotic modalities emphasizes the crucial role that design plays in determining their utility and applicability. AMRs' ability to navigate, interact, and complete tasks across a variety of application domains has been greatly aided by design decisions, demonstrating the dynamic symbiosis between design innovation and the development of applications.

2.2.2 Navigation

The field of robotics and artificial intelligence faces a fundamental challenge known as simultaneous localization and mapping (SLAM). Two fundamental components of autonomous systems are involved in this issue: localization, which refers to a robot's capacity to locate itself within the environment, and mapping, which is the process of developing or updating a map of uncharted space. Each of these two tasks informs and improves the other in a continuous loop, which is where the complexity of SLAM comes from. A wide range of applications, including autonomous vehicles, drone navigation, augmented reality, and virtual reality systems, are significantly impacted by SLAM. A robot or autonomous system can navigate unknown environments with greater accuracy and dependability by successfully resolving the SLAM problem. Despite its significance, SLAM is a difficult problem because of a number of issues like sensor data noise, a lack of computational resources, and the requirement to handle real-time data. In this section of the literature review, we will first look at the individual SLAM components—localization and mapping—before delving into the combined SLAM problem and reviewing the different methods and algorithms that have been created to tackle it. The objective is to present a thorough understanding of the current state of SLAM research and to highlight areas where additional study may advance the field.

2.2.3 Localization

Localization in robotics, which is essential for path planning and navigation, involves identifying a robot's position within an environment. This task is more complex in dynamic, unstructured, or unknown environments.

- *A. Dead Reckoning Summary*
  Dead reckoning, a basic method of navigation, calculates a robot's current position based on its previous position and control inputs. However, sensor noise and inaccuracies in control inputs can lead to cumulative errors. The robot's new pose p_{t+1} is estimated based on its prior pose p_t and control input u_t, with consideration for the introduced noise ϵ_t.

- *B. Landmark-based Triangulation Summary*
  Landmark-based triangulation uses identifiable landmarks in the environment to correct the accumulated errors from dead reckoning. The robot measures the distance and bearing to these landmarks, calculating its position using triangulation. However, its effectiveness is reliant on the availability and accurate detection of landmarks.

- *C. Probabilistic localization and Bayesian filters Summary*
  Probabilistic localization methods such as Kalman filters and particle filters apply Bayesian inference principles to account for sensor data uncertainty, thereby offering a probability distribution of possible robot positions.

  Kalman filters use state estimates and covariance to make predictions and updates for the state of a linear dynamic system with noisy measurements. Particle filters, conversely, can handle nonlinear, non-Gaussian estimation problems. They update

particle weights based on likelihood, prior, and proposal distribution, and then perform resampling.

- *D. Global Positioning System (GPS)*
  The Global Positioning System (GPS), a satellite-based navigation system, calculates the distances to at least four satellites to determine the receiver's exact position using the principle of trilateration. While GPS can offer relatively accurate localization under ideal conditions, performance deteriorates in environments that block the line-of-sight to satellites or cause multi-path distortion of GPS signals.

To overcome these limitations, other global localization methods have been investigated. These include methods based on cellular networks, where the time delay of arrival (TDOA) or angle of arrival (AOA) of signals from multiple cell towers can be used to estimate the device's location. Another approach is using global navigation satellite systems (GNSS) other than GPS, such as the European Union's Galileo system, Russia's GLONASS, or China's BeiDou system. However, all these methods have their own limitations and are typically used in conjunction with other localization methods in a multi-modal approach to achieve robust and accurate localization across a wide range of conditions.

2.2.4 Mapping

Map representations usually are divided into two major paradigms, metric and topological maps [38]. Metric maps are relatively high-resolution discretizations of the environment usually depicted using occupancy grids [39–41].

Mapping is a critical task in autonomous robotics. It refers to the process of building a model or map of the environment based on sensor data. This map serves as a spatial representation that a robot can use to navigate, plan paths, and perform tasks efficiently. The importance of mapping lies in its ability to provide the robot with an understanding of its environment, enabling it to anticipate and respond to changes effectively.

There are several mapping techniques, each with its strengths and weaknesses and suited to different types of environments and tasks:

- *Occupancy Grid Mapping*

  Occupancy grid mapping is a technique where the environment is divided into a grid of cells, each of which is classified as occupied, free, or unknown. This binary representation is simple and efficient, making it particularly suitable for 2D environments or 3D environments with mostly flat surfaces. However, it may not adequately capture the complexity of more intricate 3D structures.

- *Landmark-Based Mapping*

  In landmark-based mapping, the environment is represented by a set of distinctive features or landmarks, which could be corners, edges, or other recognizable elements. The robot identifies and locates these landmarks in its sensor data, using them to build a map and to estimate its location relative to these landmarks. This technique

can provide a more compact and efficient representation than grid mapping, but the success of this approach depends heavily on the ability to reliably identify and localize landmarks.

- *Topological Mapping*

  Topological mapping represents the environment as a graph, where nodes correspond to locations of interest and edges correspond to feasible paths between these locations. This abstract representation can capture the connectivity and layout of the environment more effectively than grid maps, making it suitable for tasks such as path planning and navigation. However, it lacks the detailed geometric information provided by grid maps.

- *Semantic Mapping*

  Semantic mapping extends the other mapping techniques by associating semantic labels with parts of the map. For instance, in addition to representing the locations of walls and obstacles, a semantic map might also indicate the locations of chairs, tables, and doors. This high-level information can enable a robot to interact more intelligently with its environment, although the task of reliably recognizing and categorizing objects is a challenging problem in its own right.

2.2.5 Simultaneous Localization and Mapping

Simultaneous Localization and Mapping (SLAM) is a fundamental problem in robotics, which involves estimating a robot's pose in an unknown environment while simultaneously constructing a map of that environment. The SLAM problem emerges as a natural extension of the separate localization and mapping tasks, as autonomous robots often need to operate in unfamiliar environments without a priori maps.

Solving the SLAM problem is challenging due to several reasons:

- *Inherent uncertainty:* Sensor data is inherently noisy and subject to measurement errors. Moreover, the robot's motion is also subject to uncertainties, such as wheel slippage or uneven terrain. These uncertainties make both localization and mapping tasks more difficult.

- *Interdependence:* Localization and mapping are inherently coupled problems. Accurate localization depends on having a good map, while creating a good map relies on precise localization. This chicken-and-egg problem makes it challenging to solve both tasks simultaneously.

- *Computational complexity:* The SLAM problem requires the processing of vast amounts of sensor data in real-time, and the size of the map and the state space can grow rapidly as the robot explores larger environments. Managing this computational burden is a major challenge, especially for online SLAM solutions that must operate in real-time.

- *Data association:* Correctly associating new sensor measurements with previously observed landmarks or map elements is crucial for building a consistent map. However, this data association task is often complicated by perceptual aliasing, where different

parts of the environment appear similar, leading to ambiguities and potential errors in the map.

- *Loop closure:* When a robot revisits a previously explored area, it is essential to recognize the loop closure and correctly update the map and the robot's pose. Failure to do so can result in a distorted or inconsistent map. Detecting loop closures is a challenging problem due to the uncertainties in the map and the robot's pose, as well as potential perceptual aliasing.

Despite these challenges, various SLAM algorithms have been developed to tackle the problem, ranging from classical methods, such as Extended Kalman Filter SLAM and Graph SLAM, to more recent approaches that leverage machine learning and deep learning techniques. In the next section, we will explore some of these SLAM algorithms in more detail.

2.2.6 SLAM approaches

There exists a diverse set of methodologies aiming to solve the complex SLAM problem, with many offering unique advantages or tackling specific difficulties inherent in this task. While it would be impossible to cover the full gamut of approaches, we can discuss some of the most influential and widely adopted techniques.

Early efforts predominantly utilized filter-based algorithms. Notably, EKF-SLAM (Extended Kalman Filter) stands out as a prominent example [42, 43]. The EKF-SLAM approach estimates both the robot's position and the map simultaneously using the principles of Bayesian probability. By maintaining a belief over the robot's pose and map configuration, it effectively deals with the inherent uncertainty in these estimates. However, EKF-SLAM does have its limitations. Primarily, it suffers from systematic error due to the linearization process, which can propagate and exacerbate inaccuracies over time.

PF-SLAM (Particle Filter SLAM) represents a different take on filter-based SLAM [44,45]. Like EKF-SLAM, PF-SLAM maintains a belief over the robot's pose and map. However, instead of representing this belief as a Gaussian distribution, it utilizes a set of particles, each representing a potential configuration of the robot's pose and map. This approach allows PF-SLAM to represent more complex belief distributions and avoid the linearization errors of EKF-SLAM. Despite these advantages, PF-SLAM tends to be computationally expensive due to the large number of particles required for accurate representations.

In recent years, Graph-based SLAM algorithms have gained significant popularity, especially for mapping large-scale environments [46, 47]. These algorithms represent the SLAM problem as a graph where nodes represent robot poses or landmarks, and edges represent spatial constraints between these poses or landmarks. By optimizing this graph to minimize the discrepancies between the constraints, these algorithms can build globally consistent maps even in the presence of significant noise and uncertainty in the sensor data.

The proliferation and advancement of sensor technologies such as LiDARs, cameras, and Inertial Measurement Units (IMUs) have significantly accelerated the development and capabilities of SLAM systems. As these sensors become more affordable and reliable, they offer increasingly detailed and robust environmental data, which in turn leads to more accurate and resilient SLAM solutions.

FastSLAM is one notable example of a SLAM system that utilizes such advancements [48]. By using ranging sensors, often LiDARs, FastSLAM can perform accurate observations of the environment. This sensor data is then utilized to estimate the robot's pose and construct a map of the environment, based on the principles of recursive Monte Carlo sampling and particle filtering.

Other approaches synergize different sensor technologies to further enhance SLAM performance. The system proposed by Kohlbrecher et al. [49] combines a robust scan-matching approach using a LiDAR system with a 3D attitude estimation system based on inertial sensing. This combination aims to leverage the strengths of both sensor modalities, providing a more robust and accurate SLAM solution.

LiDAR technology has been a particularly potent driving force in the evolution of SLAM systems. LeGO-LOAM [50], another LiDAR-based method, is primarily used for real-time pose estimation and mapping. By integrating LeGO-LOAM with other SLAM systems, it is possible to further reduce pose estimation errors, leading to more accurate and consistent maps.

Moreover, various LiDAR-based feature extraction methods have been successfully implemented in a variety of SLAM systems, contributing to their performance in map construction and pose estimation. Techniques like split&merge [51], iterative point fitting [52], and RANSAC (Random Sample Consensus) [53] are just a few examples of these powerful methods.

The evolution of sensor technology continues to transform the SLAM landscape, and the incorporation of new sensor modalities and data processing techniques promises even greater advancements in the field. As sensors continue to improve in terms of cost-effectiveness, reliability, and measurement fidelity, we can expect to see a new generation of SLAM solutions that are more accurate, robust, and capable than ever before.

Visual information plays an integral role in many SLAM systems, and advancements in computer vision have unlocked new capabilities in this domain. Visual SLAM systems leverage cameras as the primary sensor, which provide rich information about the environment and often come with significant advantages in terms of cost, size, and power consumption compared to other sensor modalities.

One standout example in the realm of visual SLAM is ORB-SLAM [54]. ORB-SLAM is a feature-based monocular SLAM system that exploits ORB (Oriented FAST and Rotated BRIEF) features for all tasks, including tracking, mapping, localization, and loop closing. By operating in real-time and using pose-graph optimization for loop closing, ORB-SLAM presents a highly efficient and effective solution to the SLAM problem.

Contrasting the feature-based approach of ORB-SLAM, LSD-SLAM [55] is a direct monocular SLAM system. LSD-SLAM estimates geometric information directly from the intensity values in the image, eliminating the need for explicit feature detection and matching. This direct approach provides several benefits, including the ability to handle texture-less and low-contrast environments where feature-based methods may struggle.

The use of AprilTag fiducial markers has also been explored in the context of SLAM. TagSLAM [56] provides a simple, adaptable, and reliable method for SLAM. By providing a front end to the GTSAM factor graph optimizer, TagSLAM enables the rapid creation of a variety of experiments based on fiducial tags.

Despite the successes of these traditional visual SLAM systems, the presence of imper-

fections such as noisy measurements and flawed system models has inspired researchers to explore the incorporation of data-driven solutions, particularly deep learning, to enhance SLAM capabilities.

A number of deep learning-enhanced SLAM solutions have replaced handcrafted modules in traditional SLAM systems. For instance, CNN-SLAM [57] incorporates CNN-based depth-map predictions with monocular SLAM. By fusing these depth maps, it can generate a more robust, dense reconstruction of the scene, subsequently used for loop closing and graph optimization.

Dynamic-SLAM [58] represents another example of deep learning integration, achieving remarkable results by incorporating deep learning techniques for object detection and semantic segmentation in traditional SLAM, particularly suitable for dynamic environments.

LS-NET [59] takes a different approach by replacing traditional local optimization methods with a learned solver in visual SLAM. This solver uses neural networks to learn robust data-driven priors, and a classic optimization-based method is then employed to refine the solution. This approach eliminates the need for hand-crafted regularizers or priors, as these are implicitly learned from the data.

The integration of deep learning in SLAM systems represents a promising frontier for the field, holding the potential to enhance system robustness, improve accuracy, and handle increasingly complex environments.

LiDAR sensors, with their ability to capture precise, high-resolution distance measurements, have become a staple technology in many SLAM systems. As with visual SLAM, the advent of machine learning techniques, especially deep learning, has substantially evolved how LiDAR data is processed and understood, enabling new approaches to feature extraction that significantly impact the performance of LiDAR-based SLAM systems.

Feature extraction plays an essential role in mapping and localization. Given the diverse ways LiDAR data can be represented - including voxels, 2D images, meshes, and point clouds - a wide variety of feature extraction methods have been developed to segment and categorize this data.

One notable method is PointNet [60], a deep learning architecture designed specifically for point cloud data. PointNet performs segmentation directly on raw point clouds, accounting for the permutation invariance of the input data (the fact that point clouds are unstructured and the order of points is arbitrary) and the potential corruption of the points. This design allows it to handle raw point cloud data directly, without the need for complex pre-processing or transformation steps.

Building upon the concepts introduced by PointNet, PointNet++ [61] enhances the original architecture by learning local features from nearby points. It uses recursive operations on a nested partitioning of the input point set to learn hierarchical features, improving the network's ability to capture local structures and details, which are critical for many mapping and localization tasks.

Other learning-based architectures are designed to leverage the unique properties of specific LiDAR data. For example, 2DLaserNet [62] is an architecture designed for 2D laser scans. By taking advantage of the ordered relationship between successive points in a 2D LiDAR scan, 2DLaserNet can perform semantic classification of mobile robot locations, aiding in both mapping and localization tasks.

The advancements in learning-based feature extraction methods for LiDAR data continue

to reshape the landscape of LiDAR-SLAM systems. By better understanding and categorizing LiDAR data, these methods enhance the ability of SLAM systems to build accurate maps and reliably localize within them, pushing the boundaries of what autonomous systems can perceive and understand about their environments.

2.2.7 Path planning and control algorithms

In this section of our literature review, we primarily explore the intricate dynamics of path planning, with additional insights into command control, both indispensable elements for the autonomous navigation of mobile robots.

Path planning, an essential facet of autonomous navigation, requires meticulous decision-making and strategy formulation. It involves the generation of efficient, feasible paths for a mobile robot to traverse from its initial position to a predetermined destination. This complex task must account for the robot's physical characteristics, the environment's static layout, and unpredictable variables such as dynamic obstacles. The primary goal of an effective path-planning algorithm is to delineate a route that not only circumvents obstacles but also optimizes factors such as travel time, energy consumption, or distance. Control algorithms play a critical role in implementing the navigation strategies devised through path planning. They are responsible for the real-time control of the robot's movements and actions based on the calculated path. These algorithms allow the robot to react to its environment, adjusting its trajectory in response to real-world constraints or changes in the environment that might not have been accounted for during path planning.

This section aims to shed light on the importance of path planning in autonomous mobile robot navigation, with due recognition of the role of command control. The understanding and application of these concepts is crucial in the ongoing development and optimization of autonomous mobile robotic systems. Command and control mechanisms are integral aspects of mobile robotics, governing how the robots interact with their environment and how they execute their tasks. These mechanisms can be explored in terms of the methods of command, and the systems of control employed.

2.2.8 Path Planning Strategies and Algorithms

The development of path-planning strategies and algorithms for autonomous mobile robots has seen numerous iterations and innovations over the years. After the introduction of graph-based path planning algorithms such as Dijkstra's [63] and A* [64] the field saw the advent of potential field methods [65] in the 1980s. Under the potential field method, the robot was viewed as a particle moving within a field of forces. The target location served as an attracting force, while obstacles were considered repelling forces. Despite the simplicity and elegance of the potential field methods, they faced limitations in complex scenarios, primarily due to their susceptibility to local minima.

The graph-based algorithms, although pioneering, had their drawbacks. Dijkstra's algorithm, renowned for identifying the shortest path in a graph, was computationally intensive, particularly for larger graphs. This was largely attributed to its exhaustive search methodology. Furthermore, Dijkstra's and A* algorithms were found wanting in their capacity to

adequately handle dynamic and high-dimensional environments, which ushered in the need for more advanced algorithms and strategies.

the classical A* algorithm, despite its improvements, grappled with difficulties in handling high-dimensional and complex environments. Its exhaustive nature, though providing optimal paths, proved challenging when applied to larger and more intricate landscapes.

Researchers developed various A*-based algorithms to address these limitations, each adding new strengths while working to overcome existing shortcomings. The Iterative-Deepening A* (IDA*) algorithm [66] proved advantageous in memory conservation, employing a depth-first search strategy while maintaining A*'s optimal characteristic. However, IDA* can face issues with time complexity, particularly for higher-cost paths.

Simplified Memory-bounded A* (SMA*) [67] offered another solution to problems with memory constraints, discarding less promising nodes when memory is limited and revisiting them if necessary. Though SMA* eased the memory consumption problem, it could potentially lead to longer run times.

Jump Point Search (JPS) [68] introduced a significant enhancement in pathfinding speed for uniform-cost grid maps, pruning substantial portions of the search space while preserving optimality. Despite its speed, JPS is specific to grid maps, limiting its broader applicability.

Dynamic A* (D*) [69] and its subsequent improvements like D* Lite [70] and Focused D* [71] extended A*'s capabilities to dynamic environments. Here, the cost of edges can change during the search, making these algorithms ideal for robotic path planning in evolving environments. However, their performance can be affected by the frequency and magnitude of these changes.

Despite these advancements, A*-based algorithms can struggle in high-dimensional spaces due to their exhaustive nature. Consequently, the dawn of the 21st century witnessed the rise of sampling-based methods like PRM and RRT, designed specifically to tackle high-dimensional problems more efficiently. Sampling-based path planning methods, including PRM [72] and RRT [73], marked a significant shift in the field. However, they were not without limitations. These algorithms, for instance, often failed to provide optimal solutions due to their inherent randomness. Moreover, the standard versions of PRM and RRT could struggle in environments with narrow passages or closely spaced obstacles. These challenges spurred the development of several advanced stochastic algorithms that aimed to improve upon the existing methods.

One of the first advancements came in the form of Fast Marching Trees (FMT*) [74], a deterministic sampling-based algorithm that produced high-quality paths rapidly and deterministically. While FMT* offered a balance between path quality and computational cost, its performance could be hindered in environments with complex geometric constraints or narrow passages.

MAPRM (Medial Axis Probabilistic Roadmap) [75] is a probabilistic roadmap algorithm that can find paths in high-dimensional configuration spaces. It works by sampling configurations from the medial axis of the free space, which is a lower-dimensional representation of the free space. The sampled configurations are then connected together to form a roadmap, and a path can be found from the start configuration to the goal configuration using a graph search algorithm.

The Stochastic Trajectory Optimization for Motion Planning (STOMP) algorithm [76] took a different approach, aiming to minimize the cost of a path considering not only the

path length but also the smoothness and obstacle clearance. While STOMP was capable of producing high-quality, smooth paths, it often required fine-tuning of its parameters and a good initial trajectory to perform effectively.

Furthermore, in the quest for optimality, the decade of the 2010s brought about the advent of optimized versions of PRM and RRT [77], named PRM* and RRT*, respectively. These algorithms incorporated a rewiring step to refine the paths continually towards optimality. While these algorithms guaranteed asymptotic optimality, the computational cost was higher, particularly for RRT*, due to the rewiring step.

The development of these stochastic algorithms has made substantial strides towards addressing the limitations of traditional sampling-based methods. Nonetheless, there remains a complex interplay between efficiency, optimality, and adaptability that continues to drive research in the field of path planning.

Despite their advantages, both PRM and RRT often failed to deliver optimal paths. The dawn of the 2010s witnessed the introduction of their optimized variants, PRM* and RRT*, which incorporated a rewiring step to continuously refine the generated paths towards optimality.

In 2014, Gammell, Srinivasa, and Barfoot made a significant contribution to the field of path planning with the introduction of a focused variant of RRT* termed Informed RRT* [78]. To enhance efficiency, the algorithm would first use RRT* to find an initial path between the start pose and the goal region. This initial path was then utilized to compute an ellipsoidal subset from which new samples would be drawn, as opposed to the full state space. This method narrowed the search to only include the subset, which contained states that could potentially improve the initial path. Through this strategic focusing, Informed RRT* demonstrated superior performance to RRT* in terms of convergence rate, final cost, and the ability to handle narrow passages.

However, Informed RRT* still had room for improvement, especially in terms of handling more complex environments and improving the quality of the solution. This motivated the introduction of the Batch Informed Trees (BIT*) algorithm [79]. Unlike its predecessors, BIT* incorporates a novel batch-based heuristic search that continuously refines a homotopy-class representative tree. It also uses a series of batches to refine its paths, enabling the algorithm to improve the solution's quality over time. As a result, BIT* is able to handle complex environments and provides solutions of higher quality than Informed RRT*.

The Voronoi diagram-based path planning methods [80] also deserve mention. they find the path that maximizes the clearance from the obstacles, thus reducing the chances of collision. These methods can be particularly beneficial in environments with narrow passages or closely spaced obstacles.

Hybrid methods have also been explored, combining different strategies to exploit their respective strengths and mitigate their weaknesses. For instance, combining graph-based and sampling-based methods can result in a planner that is both computationally efficient and capable of finding high-quality paths.

Through the evolution of these path-planning strategies, researchers have continued to strive for algorithms that can handle a range of environments, from static to dynamic and from low-dimensional to high-dimensional, while meeting the demands for efficiency, optimality, and safety. The advancements in path-planning strategies have been crucial to the increasingly sophisticated capabilities of autonomous mobile robots.

2.2.9 Methods of Robot Command

Mobile robots can be commanded through a variety of structures, depending on factors such as the complexity of the task, safety requirements, and the need for precision.

- *Manual Command*: Here, a human operator directly controls the robot's actions. This is often used in situations where precise movements are required, such as bomb disposal or surgical robots. However, it may be inefficient or impractical for routine or large-scale operations.
- *Supervised Command*: In this scenario, the robot operates semi-autonomously. The human operator provides high-level commands, while the robot makes its own decisions on how to implement them. This method is used when there's a need to balance between human judgment and the efficiency of automation.
- *Autonomous Command*: In this mode, the robot is self-governed and makes decisions based on its programming and inputs from its sensors. Autonomous robots are ideal for complex or dangerous tasks such as space exploration or search and rescue missions where human control may not be feasible.

Control systems dictate how the robot interprets and acts on commands. Various paradigms have been used in mobile robotics:

- *Classical Control Methods*: These methods, including PID controllers, rely on mathematical models of the system and the environment. They are computationally efficient but may not perform well in complex or dynamic environments due to their reliance on a fixed model.
- *Fuzzy Logic Controllers*: Fuzzy logic allows for a more intuitive and flexible control scheme, which can handle imprecise or noisy sensor data. It provides a mechanism to model human-like decision-making processes but can be challenging to design and optimize.
- *Neural Networks*: Neural networks can learn to control a robot based on input-output data, which can be ideal for complex, non-linear systems. However, they require a large amount of data for training and their operation can be opaque, which is a challenge for safety-critical applications.
- *Genetic Algorithms*: These algorithms use principles of evolution to optimize the control parameters. They can deal with a large number of parameters and non-linear systems, but the optimization process can be computationally intensive.

Each of these methods has its strengths and weaknesses, and the choice of command method and control system depends largely on the specifics of the task, the environment, and the requirements for the mobile robot's operation.

Classical Control Methods

various approaches within classical control methods have been explored to address diverse applications. These approaches leverage mathematical models of the system and the environment to interpret commands and enable precise robot control. In this section, we review a range of classical control methods and highlight their specific applications in mobile robotics research.

- *Proportional-Integral-Derivative (PID) Control*

  PID control [81, 82] is a widely employed classical control method that has been extensively studied in mobile robotics [83, 84]. Its simplicity and efficiency make it suitable for a range of applications. Researchers have applied PID control to tasks such as autonomous navigation [85, 86], and path tracking [87].

- *Adaptive Control*

  Adaptive control [88] is another approach within classical control methods that have gained attention in mobile robotics research. This method aims to address uncertainties and changes in the system and environment by adapting control parameters in real time. Adaptive control has found applications in mobile robot trajectory planning, where it enables robots to autonomously adapt to dynamic environments.
  for instance [89] proposes an adaptive extension to the integrated kinematic and torque controller for nonholonomic mobile robots. Through an adaptive backstepping approach, it shows that an adaptive tracking controller can be designed for both kinematic and dynamic models with unknown parameters. A design example for a two-wheel actuated mobile robot is provided, featuring a novel kinematic adaptive controller and a derived torque adaptive controller. however it should be noted that Adaptive controllers for mobile robots may struggle with real-time adaptation, stability assurance, and dealing with non-linear uncertainties.

- *Sliding-Mode Control*

  Sliding mode control is a robust control technique within classical control methods that has been explored in mobile robotics for its ability to handle disturbances and uncertainties. It operates by driving the system state onto a predefined sliding surface, ensuring robust performance even in the presence of modeling errors or external disturbances. Sliding mode control has been applied to mobile robot path tracking, where it enhances tracking accuracy and robustness. for example [90] proposes a robust tracking control for nonholonomic wheeled mobile robots using sliding mode. The robot's posture is represented by polar coordinates, and its dynamic equation is feedback-linearized using the computed-torque method. The newly proposed sliding mode control law aspires to stabilize the robot to a desired trajectory, showing robustness to bounded external disturbances. Experimental results validate the effectiveness of the approach.

- *Optimal control*

  Optimal control methods aim to find control inputs that optimize a given performance criterion. Among the various optimal control techniques, model predictive control

(MPC) has been extensively studied in mobile robotics. MPC utilizes a predictive model of the system dynamics to compute control inputs that minimize a cost function over a finite prediction horizon. It has been applied to mobile robot motion planning and control, offering the capability to handle constraints and optimize trajectories.
in [91] The author develops an optimal controller for car-like robot trajectory tracking. After obtaining a tracking error model based on the robot's kinematic equations and linearizing this model, they use an optimal Linear Quadratic (LQ) design approach to create a controller. Simulations validate its effectiveness for tracking both straight and curved trajectories. Model Predictive Control (MPC) is a type of optimal control that has received significant attention within the robotics community due to its versatility and ability to handle constraints [92]. At its core, MPC involves repeatedly solving a finite-horizon optimal control problem, using a model of the system dynamics to predict future behavior. The solution to this optimization provides a sequence of control actions, of which only the first is implemented before the process is repeated at the next timestep. This gives MPC the ability to handle multi-variable systems and constraints while providing a robust and adaptable framework for control.

In conclusion, the literature review has highlighted significant advancements and evolution in the field of path planning and control algorithms in mobile robotics. The path planning problem's intrinsic complexity, which involves traversing from an initial position to a predetermined destination efficiently and safely, has been a prime focus in the field. Initial solutions involved graph-based path planning algorithms like Dijkstra and A*, which paved the way for future advancements. However, these methods faced challenges such as computational intensity and limitations in handling dynamic and high-dimensional environments.

The shortcomings of these traditional algorithms spurred research in new directions. Notably, the Potential Field Method proposed a metaphorical approach but fell short in complex scenarios due to issues with local minima. Sampling-based methods such as PRM and RRT emerged as more efficient solutions for high-dimensional environments, but they often lacked optimality and struggled in spaces with narrow passages or closely spaced obstacles.

Subsequent advancements saw several variants of these techniques, each striving to address specific limitations while adding unique strengths. From memory-conserving IDA*, SMA* to speed-enhancing JPS and the dynamic-environment suited D* family, these variations broadened the applicability of A* based methods. Sampling-based methods also saw enhancements with the advent of FMT*, STOMP, PRM*, RRT*, Informed RRT*, and BIT* among others. These methods collectively brought improvements in path quality, computational cost, adaptability, and handling complex environments. The Voronoi diagram-based methods and hybrid strategies further diversified the path-planning landscape.

Control algorithms, complementing the path-planning strategies, are pivotal in implementing the navigation plans. They manage the real-time control of the robot's actions, adjusting the trajectory as per environment dynamics and ensuring the efficient execution of planned paths.

While significant strides have been made, challenges persist in developing path-planning algorithms that can optimally balance efficiency, optimality, safety, and adaptability in varying environments. The ongoing evolution of these strategies and algorithms remains crucial for the future of autonomous mobile robotics, with potential applications extending from

service robots, autonomous vehicles, and beyond. Future research endeavors will undoubtedly continue to push the boundaries of these elements, driving the broader field of mobile robotics forward.

In addition to these aspects, the importance of creating new algorithms that consume less computational power and memory cannot be overstated. With increasingly complex tasks and environments being tackled by autonomous robots, there's a pressing need for more efficient algorithms.

Firstly, computational efficiency is paramount for real-time or near real-time decision-making in autonomous systems. High computational demand can lead to latency in decision-making, potentially causing delays in response times and reducing the overall efficiency of the robot. More efficient algorithms allow for faster processing of data, leading to quicker decision making and actions, thus enhancing the overall performance of the robot.

Secondly, memory efficiency is equally important. As the environments in which robots operate become more complex and dynamic, the amount of data that needs to be processed and stored increases. Algorithms that are memory efficient can handle larger data sets, operate in more complex environments, and do so without needing a proportional increase in hardware capacity. This not only makes the system more capable but also more cost-effective to operate.

Further, efficient algorithms can pave the way for miniaturization and power savings. Reducing the computation and memory needs of an algorithm allows the same task to be completed with smaller, less power-hungry components. This not only reduces the size and weight of the robot but also decreases its energy consumption, enabling longer operational times or increased functionality.

Finally, algorithms that use less computation and memory resources can be integrated more readily into systems with other concurrent tasks. This is crucial in multi-task robots where computational and memory resources are shared among different tasks. Efficient algorithms ensure that resources are distributed adequately, enabling the simultaneous execution of several complex tasks without compromise.

In conclusion, the development of new, efficient algorithms is of paramount importance to the future of robotics. It is a challenge that the research community must rise to, in order to continue driving forward the capabilities of autonomous systems. By striving for lower computation and memory demands, researchers can pave the way for more capable, efficient, and versatile mobile robots.

2.3 AMRs Generalities

For thousands of years, human beings have been fascinated by 'artificial creatures'. Ancient Greeks believed that Hephaestus, "God of fire" had mechanical assistants that he made from gold to help him in his workshop. The Arabic inventor and Islamic scholar Ismael al-jazari (1136-1206) describe in his book, "The book of knowledge of ingenious mechanical devices," several automata, including a humanoid girl serving drinks and a hand-washing/ ablution automaton.

The first programmable robot was a robotic arm invented by George Devol in 1954. General Motors was the first company to use this robot for industrial purposes (stacking hot

Figure 2.3: illustration of aljazari's washing automaton.

Table 2.1: Classification of robots by Locomotion, environment, application and size

Locomotion	Environment	application	size
Wheeled robot	Ground robot	Industrial robots	Micro robots
flying robot (artificial wings, propellers .. etc)	Underwater robot	Healthcare robots	nanobots
Legged robot	Drones	Agriculture	
Crawler robot		Military robots	
Swiming robot (using palmes)		Rovers	
Tracked robot			

metal). Since then, robotics has revolutionized many industries, but it was the development of autonomous mobile robots (AMRs) that changed our relationship with these machines forever. With contrast to AGVs (Automated Guided Vehicles) that rely on a predefined path to move in the environment, AMRs, are robots that are capable of free navigation through un-constrained environments with no human intervention. And since there is a wide range of possible environments/conditions in which they can serve, a wide selection of AMRs have been developed to satisfy the rising demand for autonomous solutions in all sorts of industries (healthcare, military, agriculture, logistics . . . etc.). These robots can be divided into different classes, depending on the classification criteria.

This chapter provides a comprehensive exploration of autonomous mobile robots, encompassing their design and the theoretical foundations of autonomous navigation. Beginning with an overview of the generalities surrounding autonomous mobile robots, we delve into the intricate web of design considerations that dictate their form and function. This includes a meticulous analysis of various design aspects, such as chassis construction, power sources,

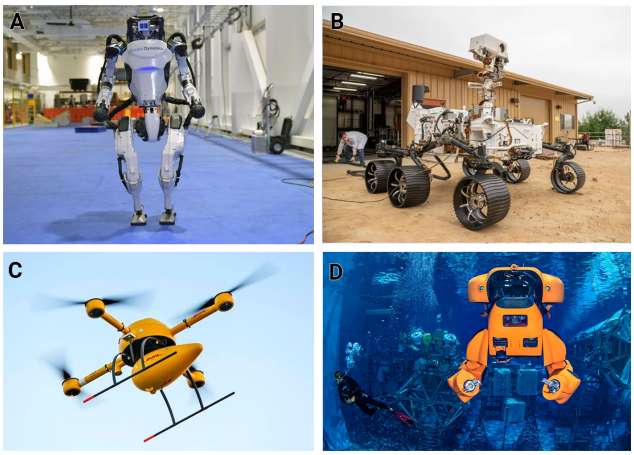

Figure 2.4: diffrent types of mobile robots.

and the selection of sensors and actuators. Following this, the discussion transitions into the theoretical underpinnings of autonomous navigation, a critical element in the realm of robotics. We explore the challenges and nuances associated with navigating indoors and outdoors, as well as the distinctions between static and dynamic environments. Additionally, the focus shifts to the pivotal role of sensors, including lidars, IMUs, cameras, and GPS, in the navigation process of autonomous mobile robots. These sensors serve as the sensory instruments, facilitating data collection and interpretation, ultimately enabling AMRs to navigate their surroundings with precision and adaptability.

2.3.1 Wheeled AMR models

In this dissertation, we are going to develop novel approaches for AMRs navigation bearing in mind a particular class of mobile robots that is considered the most used type in industrial and indoor environments in general, due to their low energy consumption, relatively higher speed and fewer stability problems which required less control effort. At the same time wheeled mobile robots are suitable for indoor applications because they don't have to deal with uneven terrain conditions that exist in outdoor environments [springer handbook of robotics].
In manufacturing and logistics, wheeled robots are used to automate the production process. They can be programmed to move materials from one location to another, which helps to reduce labor costs and increase efficiency. They can also be used to explore hazardous environments or to collect data. The are also being used in a variety of other applications, such as security, education, and research. They can be programmed to patrol an area, detect intruders, and even provide assistance to people in need. In education and research, wheeled robots can be used to learn about robotics concepts an develop new approaches.

2.3.2 Wheels types

Wheeled autonomous mobile robots are defined by their manner of locomotion. They use different types of wheels to achieve stable ground-based mobility. The type and formation of wheels they have are determined by their specific application which reflects directly on the control approaches researchers develop and implement. B.sciciliano and O.khatib in their book titles “2008 springer handbook for robotics”, divide the wheels into two main types : Standard wheels and Special wheels. This devision was based on the wheels structure, where the standard wheel is explained as a simple tire, while special wheels contains other unique mechanical structures such as spheres and rollers.

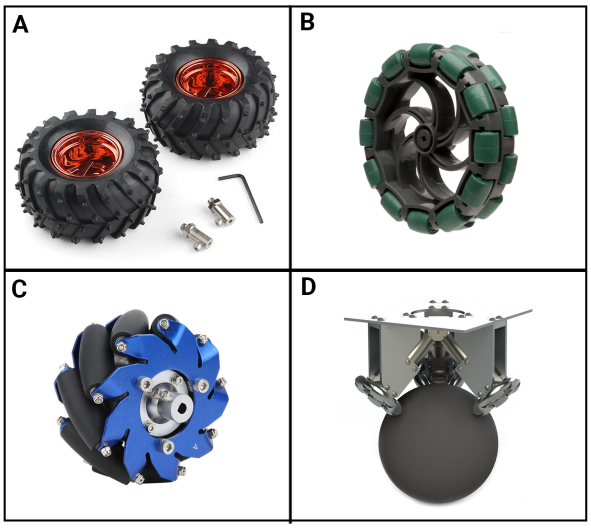

Figure 2.5: diffrent types of mobile robots wheels.

Standard wheels

A wheel is a circular mechanical component that rotates on an axle which allows the robot to move in its environment similar to traditional wheeled vehicles. Standard wheels can be designed to serve multiple purposes related to robotics locomotion. For example, Active-drive, passive-steering wheels allow for efficient control of the robot’s velocity and orientation (differential drive), while passive-drive passive-steering wheels or caster wheels are useful for providing balance to the locomotion module and also simplifying motion planning and control of the robot. It should also be noted that changing the castor offset and castor angle parameters can affect the stability, maneuverability, and swivel resistance of the caster wheels.
The steering and driving motions can be made passive or active based on the particular application. The six wheels that makeup NASA’s Mars rover Perseverance are each driven by a separate motor.Additionally, the two front wheels and the two rear wheels both have active steering, allowing the vehicle to swerve and curve, performing arcing maneuvers [93].

Standard wheels are common and have a relatively simple structure which can notably reduce the cost of building mobile robots and thus widen the range of possible applications. This is why till today standard wheels are the most commonly used type of wheels in robotics and transport vehicles in general (cars, bicycles, trolleys, carts, landing gears ... etc)

Special wheels

Special wheel designs are developed to overcome the non-holonomic velocity constraints created by standard wheels . Omnidirectional motion, in the sense of moving in any arbitrary direction, without the need for reorientation can be obtained by altering the wheel design to include passive spheres or rollers or in different formations to eliminate the non-holonomic constraints on the velocity.

Mecanum wheels are designed to create lateral motion using passive rollers around their rim. The angle between the rolling axis of these rollers and the wheel axle is generally either +45 degrees or -45 degrees, This design permits Omnidirectional robot movement, which increases maneuverability and drastically reduces the path planning complexity. In literature, Mecanum wheels with a roller angle of 0 degrees are mostly refered to as sweedish wheels, in relation to their sweedish inventor Mr.Bengt ilon, that invented these wheel design while working for Mecanum AB. Spherical wheels or Omni-Balls as refered to sometimes, are another way for concurring non-holonomic velocity constraints for mobile robots. The most most common design of such a wheel work by using rollers to drive a spherical Wheel in direct contact with the ground. This concept is the inverse of a traditional computer mouse design, where the is generated when the moving ball drives the mouse rollers [94]. This type of wheels is not approporiate for outdoor applications at it can be polluted which affects its

2.3.3 Differential Drive Robot

Differential drive is a type of robotic locomotion system that uses two independently driven wheels mounted on the same axis to move a robot. This system is commonly used in mobile robotics due to its simplicity and high maneuverability.

In differential drive, each wheel is driven independently, which allows the robot to turn in place or move in various directions. This is achieved by independently controlling the velocity of each wheel. For instance, driving the left wheel forward and the right wheel backward at the same speed will make the robot turn in place. Driving the left wheel faster than the right wheel causes the robot to turn left.

The kinematics of a differential drive robot is described by the following equations:

$$v(t) = \frac{r}{2}\left(v_r(t) + v_l(t)\right) \tag{2.1}$$

$$w(t) = \frac{r}{L}\left(v_r(t) - v_l(t)\right) \tag{2.2}$$

Here, $v(t)$ and $w(t)$ represent the linear and angular velocities of the robot, respectively. These velocities are functions of time, indicating that they can change as the robot moves. The wheel radius is denoted by r, while $v_r(t)$ and $v_l(t)$ are the instantaneous velocities of the right and left wheels. The term L represents the distance between the two wheels, also known as the wheelbase.

The linear velocity $v(t)$ is the average of the velocities of the two wheels, and it dictates how fast the robot moves forward or backward. The angular velocity $w(t)$, derived from the difference in wheel velocities, determines the rate at which the robot turns. By adjusting $v_r(t)$ and $v_l(t)$, the robot can perform a range of maneuvers, from straight-line motion to tight in-place turns.

Understanding these kinematic equations is crucial for designing control algorithms for differential drive robots, as they provide a direct relationship between wheel speeds and the robot's motion in the environment.

2.3.4 Ackermann Steering Robot

Overview of Ackermann steering

Ackermann steering is a geometry-based steering system that assures that during turns, all wheels of a vehicle follow the correct arc. When cornering at high speeds, the system aids in maintaining stability and control.

The basic idea behind Ackermann steering is that all wheels must turn on circles with a shared center point (called the instantaneous center of rotation, or ICR). The following equations describe the relationship between the inner and outer wheel turning angles:

$$\tan(\alpha_i) = \frac{L}{R - d/2} \tag{2.3}$$

$$\tan(\alpha_o) = \frac{L}{R + d/2} \tag{2.4}$$

where α_i and α_o are the turning angles of the inner and outer wheels, L is the wheelbase, R is the turning radius, and d is the distance between the two front wheels.

Key components

- Steering linkage:

  The steering linkage is a collection of rods, levers, and joints that convey motion from the steering mechanism to the wheels. It is intended to keep the wheels in the right Ackermann steering geometry during turns.

- Powered wheels:

  Ackermann steering robots typically have two motorized rear wheels and two steerable front wheels. The driving force is provided by the rear wheels, while the direction is determined by the front wheels.

- Motors:

  Motors are needed to power the rear wheels as well as control the steering system. Two motors are often used: one to drive the rear wheels and another to operate the steering mechanism.

Advantages and disadvantages of Ackermann steering

Advantages:

1. *Stability at high speeds*

   Ackermann steering geometry guarantees that during turns, all wheels follow the correct arc, avoiding the chance of sliding and preserving stability at high speeds. and since Proper wheel alignment reduces lateral forces acting on the wheels, the probability of losing traction is lower and the robot's ability to keep its intended course is boosted. This capability is especially beneficial for high-speed robots and robots operating in areas where keeping control during turns is critical.

2. *Predictable steering behavior*

   Ackermann steering provides predictable and consistent steering behavior due to its geometry-based design,which makes the robot motion easier to manage. also the steering design ensures that the inner and outer wheels revolve at distinct angles, allowing for seamless transitions between straight and curved tracks. This function is considered to be highly beneficial for robots that must traverse complicated areas with a variety of curves and obstacles, as well as autonomous vehicles that require precise steering control to follow their intended pathways safely.

3. *Good cornering performance*

   The Ackermann steering robots usually outperform differential drive robots in terms of cornering performance, since they maintain superior traction and control throughout turns. The improved steering geometry maintains the wheels aligned with the direction of travel, which reduces their tire scuff and guarantees maximum ground contact. This results in a smoother turning motion and more efficient power consumption, which might be advantageous for robots that need to navigate tight turns often or work in situations with limited space.

4. *Reduced tire wear*

   The Ackermann steering geometry often also contributes to lower tire wear by reducing tire scuff during turns. This usually results in cheaper maintenance costs and longer tire life, which is beneficial for robots operating in conditions where tire wear is a concern or for applications where low maintenance is a requirement.

Disadvantages

1. *Limited maneuverability*

   Ackermann steering robots have restricted agility compared to differential drive and holonomic robots since they require a bigger turning radius. In tight or constricted environments, more nimble robots, such as those with differential or holonomic drive systems, can traverse more effectively. Furthermore, Ackermann steering robots often cannot do in-place rotations, which can be disadvantageous in certain applications requiring tight turns or rapid changes in direction.

2. *Complex design*

   The steering connection system for differential drive robots is more complex, which may result in a more expensive robot that requires more maintenance. Steering arms, tie rods, and steering mechanisms are examples of additional mechanical components that might increase the overall weight and complexity of the robot. As more components are prone to wear and tear or probable failure, this might be the cause of higher manufacturing costs and increased maintenance requirements.

3. *Increased power consumption*

   Ackermann steering robots typically require additional motors to control the steering mechanism, which can increase the overall power consumption of the robot. This can be a concern in applications where power efficiency is critical or where extended battery life is desired. Differential drive and holonomic robots, which use fewer motors to achieve steering control, may be more energy-efficient alternatives.

4. *Slower response time*

   Ackermann steering robots may have slower response times for beginning turns or changing direction as compared to differential drive and holonomic robots. This is due to the steering mechanism's need to physically rotate the wheels in order to achieve the desired turning angle, which can take longer than the near instantaneous response times of differential drive and holonomic robots, which can change direction by adjusting the relative speeds of their wheels. This longer response time may be an issue in applications that demand quick direction adjustments or maneuvering.

Applications of Ackermann steering robots

1. Outdoor robots

   Ackermann steering robots are suitable for outdoor environments, where they can navigate through turns with better stability and control.

2. Agricultural robots

   In agricultural settings, Ackermann steering robots can be used for tasks such as crop monitoring, spraying, and harvesting, where precise and stable movement is required.

3. Autonomous vehicles

   Ackermann steering geometry is widely used in autonomous vehicles to ensure stable and predictable steering behavior, especially during high-speed turns.

2.3.5 Self-balancing Robots

A self-balancing robot is a type of mobile robots that maintains upright balance through a combination of mechanical design and control algorithms. Its mechanical structure often resembles an inverted pendulum, typically with one or more wheels, motors, and a power source.

SBRs find diverse applications across various domains. Á. Odry et al [95]. discuss the use of SBRs in educational environments, particularly for teaching feedback control concepts and fuzzy control in a hands-on manner. Beyond the classroom, self-balancing robots include iconic examples such as the Segway [96], used for personal transportation, and Boston Dynamics' Handle [97], employed in material handling within warehouses. Additionally, the MiP robot by WowWee [98] offers an entertaining introduction to robotics.

The design and concept of a self-balancing robot offer several advantages. Firstly, it provides stability and mobility in a small footprint. The robot's ability to maintain balance on a single point or axis allows it to navigate through narrow spaces and crowded environments. This compactness is particularly beneficial in applications such as warehousing and healthcare. The autonomous capabilities come from sensors like gyroscopes and accelerometers, which detect the tilt angle and rate of change. This data is fed into a control system, such as a Proportional-Integral-Derivative (PID) controller, which regulates motor adjustments to maintain balance. This rapid, continuous adjustment mechanism creates a highly responsive system, enabling the robot to quickly adapt to changes and disturbances in its environment. Advanced versions may incorporate additional sensors and machine learning algorithms for navigation, obstacle avoidance, and object recognition. The inclusion of machine learning algorithms enhances the robot's ability to learn from its environment, improving its functionality and performance over time. Applications of these robots range from warehousing and healthcare to surveillance and education, as well as in personal transportation devices. Their versatility and adaptability make them suitable for a wide array of tasks and environments.

Despite the challenges in achieving a stable control system and ensuring reliable sensor operation, the self-balancing robot design holds immense potential. Advancements in sensor technology, control systems, and machine learning are driving rapid development in the field, promising even more sophisticated and capable self-balancing robots in the future.

Working Principles

Self-balancing robots use a combination of mechanical engineering and control algorithms to keep their balance. The mechanical setup often appears resembling an upside-down pendulum on a cart. The challenge in maintaining control is moving the cart to keep the pendulum upright.

Newton's second law, which states that an object's acceleration is inversely proportional to its mass and directly proportional to the net force acting on it, can be used to explain the dynamics of an inverted pendulum.

A self-balancing robot relies on dynamic equations that capture the forces and moments acting on its system to maintain an upright position. The equilibrium of the robot is dynamically unstable, necessitating a continuous feedback control system for stabilization.

As illustrated in the provided figure, the robot is represented by a cart-pole system. The cart moves along the horizontal x-axis, while the pole, attached to the cart, is allowed to pivot and thus may deviate from the vertical y-axis by an angle $\theta(t)$. The forces acting on this system include the gravitational force (mg), frictional force (represented by a damping coefficient b times the angular velocity $\dot{\theta}(t)$), and a control input force $F(t)$. The moment of inertia of the pole about the pivot point is denoted by I.

The dynamics of a self-balancing robot are elegantly captured by a set of equations that

take into account the various forces acting upon the system. These equations are central to the design of a control system that enables the robot to balance dynamically. The robot's configuration, often modeled as a cart-pole system, is inherently unstable, which requires an active feedback control system for stabilization.

In the schematic provided in Figure 2.6, the cart-pole system is demonstrated along with the forces in play. The cart moves along the horizontal x-axis, and the pole, which is attached to the cart, can pivot and thus may deviate from the vertical y-axis by an angle $\theta(t)$. The forces acting on the system include gravity (mg), damping due to friction (represented by a damping coefficient b times the angular velocity $\dot{\theta}(t)$), and a control input force $F(t)$. The moment of inertia of the pole about the pivot point is denoted by I.

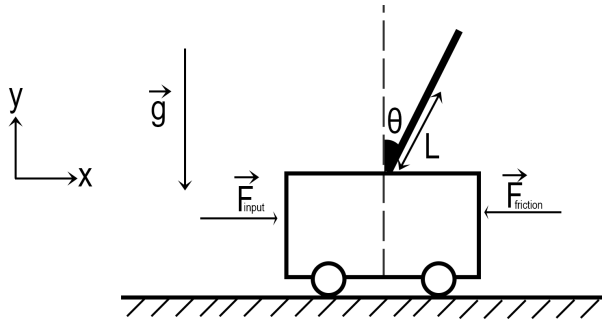

Figure 2.6: Schematic of the self-balancing robot, depicting the forces and variables at play.

Dynamic equations were derived from the forces in Figure 2.2, the complete derivation of the system's transfer function can be found in Appendix A - Calculation of the system's transfer function.

The dynamic model takes the following form [99]:

$$F_{\text{input}} = (m_{\text{cart}} + m_{\text{pend}})\ddot{x} + f\dot{x} + m_{\text{pend}}L\ddot{\theta}\cos(\theta) - m_{\text{pend}}L\dot{\theta}^2\sin(\theta) \tag{2.1}$$

$$(I_{\text{pend}} + m_{\text{pend}}L^2)\ddot{\theta} + m_{\text{pend}}gL\sin(\theta) = -m_{\text{pend}}L\dot{x}\cos(\theta) \tag{2.2}$$

Where m_{cart} and m_{pend} represent the cart's and pendulum's weight respectively. Parameter, L, is the distance to the pendulum's center of mass, θ the angle between the pendulum and the vertical axis as seen in Figure 2.2. The parameter, f is a friction parameter. Rewriting the equations as transfer functions renders:

$$\Psi(s) = \frac{\frac{m_{\text{pend}}L}{q}s}{s^3 + \frac{f}{q}(I_{\text{pend}} + m_{\text{pend}}L^2)s^2 - \frac{(m_{\text{cart}}+m_{\text{pend}})m_{\text{pend}}gL}{q}s - \frac{fm_{\text{pend}}gL}{q}}U_{\text{input}}(s) \tag{2.3}$$

$$X(s) = \frac{\frac{(I_{\text{pend}}+m_{\text{pend}}L^2)s^2 - gm_{\text{pend}}L}{q}}{s^4 + \frac{f}{q}(I_{\text{pend}} + m_{\text{pend}}L^2)s^3 - \frac{(m_{\text{cart}}+m_{\text{pend}})m_{\text{pend}}gL}{q}s^2 - \frac{fm_{\text{pend}}gL}{q}s}U_{\text{input}}(s) \tag{2.4}$$

In addition to this dynamic model, it's crucial to understand the static and kinematic models of the robot as functions of time. The static model considers the robot in a state of equilibrium, meaning it is perfectly upright with no motion. Under these conditions, the

gravitational force acting on the robot is in balance with the control forces, and the net torque around the pivot point is zero. This model is expressed as $mgL\sin(\theta(t)) = F(t)$ where $\theta(t) = 0$ for all t, indicating the robot is perfectly vertical at any time.

The kinematic model, focusing on the robot's motion, describes the robot's angular position ($\theta(t)$), angular velocity ($\dot{\theta}(t)$), and angular acceleration ($\ddot{\theta}(t)$) as functions of time. These variables, interconnected through time derivatives, are vital for understanding the motion of the robot over time.

Thus, a comprehensive understanding of self-balancing robots encompasses static, kinematic, and dynamic models, all considered as functions of time. This multi-tiered approach provides insight into the robot's operation, from its equilibrium state to its motion dynamics and the control algorithms needed for stabilization.

It should be noted that this is a simplified explanation, and the actual implementation may need to take into account additional elements like the motor dynamics, the nonlinearities of the system, and outside disturbances.

2.3.6 Autonomous navigation

Autonomous Mobile Robots (AMRs) have developed into an effective solution in a variety of industries, owing to their capacity to conduct real-time activities in complicated situations in a safe manner. AMRs have considerably enhanced productivity across many industries by lowering human exposure to dangers and reducing the time and energy necessary for certain operations. Autonomous navigation, which allows robots to move without external supervision or control, is one of the most important features of AMRs. This capability has enabled a wide range of applications, such as self-driving automobiles and search and rescue missions, resulting in a natural link between computer science and robotics.

Advanced algorithms and technologies such as machine learning, computer vision, sensor fusion, and localization techniques are widely used in this interdisciplinary field of research. For example, machine learning is critical in allowing robots to adapt to new situations, spot patterns, and make data-driven judgments. Its adaptability is especially useful in dynamic and unstructured contexts, where traditional rule-based programming falls short. computer vision may it be learning-based or traditional is an important component that enables robots to process and analyze visual data from cameras and other sensors. As a result, robots can recognize objects, barriers, and landmarks, as well as determine their position and orientation in relation to their surroundings. This ability to observe the environment adds to the adaptability afforded by machine learning.

Sensor fusion combines data from numerous sources, such as cameras, LIDAR, radar, and GPS, to build a full knowledge of a robot's environment. This combined knowledge is critical for making informed decisions about a robot's actions and motions, and it improves computer vision perception capabilities. Localization techniques, such as Simultaneous Localization and Mapping (SLAM), improve AMR efficacy by predicting a robot's position inside an area while simultaneously constructing a map of its surroundings. This feature is especially useful for robots exploring unknown or changing settings, and it contributes to the robustness of autonomous navigation.

The potential uses of AMRs are quickly expanding as research and development in autonomous navigation continue. Agriculture, healthcare, and manufacturing are among the industries that are rapidly incorporating AMRs to automate processes, enhancing efficiency and lowering reliance on human labor. AMR developments and integration will influence the future of work, revolutionizing how we approach productivity and safety in our daily lives, and eventually creating a seamless connection between each part of this new technology.

2.3.7 AMR Environmental Challenges

Autonomous Mobile Robots (AMRs) must travel through a range of situations, each with its own set of problems and peculiarities. We address the many types of robot surroundings and their distinctive properties in the context of AMR autonomous navigation in this subsection. We will also look at the effects of static and dynamic impediments on robot navigation, the importance of safety in industrial and residential applications, and the effects of topography and environment on robot navigation.

Robot environments can be broadly classified into two categories: indoor and outdoor. Indoor environments usually include homes, offices, warehouses, and factories, while the outdoor environments encompass urban streets, rural areas, off-road terrain. Each environment can present specific challenges that AMRs must overcome to successfully navigate and perform tasks :

- *Indoor Environments*:

  Indoor environments are characterized by confined spaces, walls, furniture, and other static obstacles. They often contain a mix of static and dynamic obstacles, such as people or other robots, which can create complex navigation scenarios. In these settings, safety is a paramount concern, especially in domestic applications where robots must share the environment with humans and pets.

- *Outdoor Environments*:

  Outdoor environments are typically more expansive and may include various terrains, such as asphalt, gravel, grass, or even rough off-road landscapes. The problem is that Navigating outdoors often introduces new challenges, such as varying lighting conditions, weather-related factors, and a greater variety of static and dynamic obstacles, including vehicles, pedestrians, and animals.

static and dynamic environments

AMRs must be able to manage static and dynamic obstacles, as well as adapt to a variety of terrains and settings, in order to successfully navigate these environments. This involves a grasp of the disparities between static and dynamic contexts, as well as the development of effective techniques for dealing with these variances.

Static environments are those in which obstacles and other features, such as walls, columns, and big immovable items, remain fixed in place. Because the layout remains same throughout time, these surroundings are often easier for AMRs to explore. As a result, the robot may rely on a pre-programmed map for consistent and predictable course planning. In static situations, AMRs use approaches like graph-based methods or grid-based algorithms to construct optimal pathways while avoiding obstacles.

Dynamic environments, on the other hand, contain features that are continually changing, such as moving impediments such as pedestrians, automobiles, or other robots. These surroundings are far more difficult for AMRs to navigate because they must constantly adjust to changes in the environment, updating their maps and pathways in real-time. AMRs require complex sensing, planning, and control systems that can understand and adapt to environmental changes fast in order to navigate dynamic settings effectively.

In both industrial and household AMR applications, safety is of the utmost importance. Robots in industrial contexts must be able to travel securely among machinery, workers, and other equipment without causing accidents or interruptions.

Various terrains and settings present their own set of problems for AMRs' autonomous navigation. Indoors, AMRs may confront obstacles like as stairs, ramps, or uneven flooring, whilst outside locations may provide difficulties such as rocky terrain, inclines, or even water dangers. To address these varied situations, AMRs must be outfitted with appropriate sensors, actuators, and algorithms. Wheeled robots, for example, may struggle with uneven terrain or stairs, but legged robots, such as Boston Dynamics' Spot, may be more suited for these settings due to their superior mobility and agility.

2.3.8 Sensors in AMR autonomous navigation

A range of sensors play an important part in perceiving and interpreting the world in the field of autonomous navigation for Autonomous Mobile Robots (AMRs). These sensors provide crucial information to the robots, allowing them to navigate through a variety of environments securely and efficiently. Cameras, LIDAR, RADAR, ultrasonic, and infrared sensors are among the most often utilized sensors in autonomous navigation.
Robots can acquire high-resolution photos of their surroundings using cameras as a form of sensor. This visual data gives important details about objects, textures, and colors, which can be used for tasks like object recognition, obstacle detection, and semantic segmentation. LIDAR (Light Detection and Ranging) sensors utilize laser beams to estimate distances to objects and produce three-dimensional maps of their surroundings, whilst RADAR (Radio Detection and Ranging) sensors use radio waves for the same purpose. LIDAR and RADAR sensors are both useful for creating accurate maps and locating the robot inside those maps. Ultrasonic sensors detect the existence of obstacles using sound waves, whereas infrared sensors measure the distance to objects using the reflection of infrared light.These sensors contribute to the overall understanding of the environment and facilitate the implementation of obstacle avoidance strategies.

LIDAR

LIDAR (Light Detection and Ranging) has become a crucial technique in the realm of autonomous navigation for developing accurate, high-resolution maps of the environment. LIDAR works by sending out laser beams and measuring how long it takes for the light to reflect off things and return to the sensor. The LIDAR system can compute distances to objects with high precision thanks to the time-of-flight (ToF) measurement.

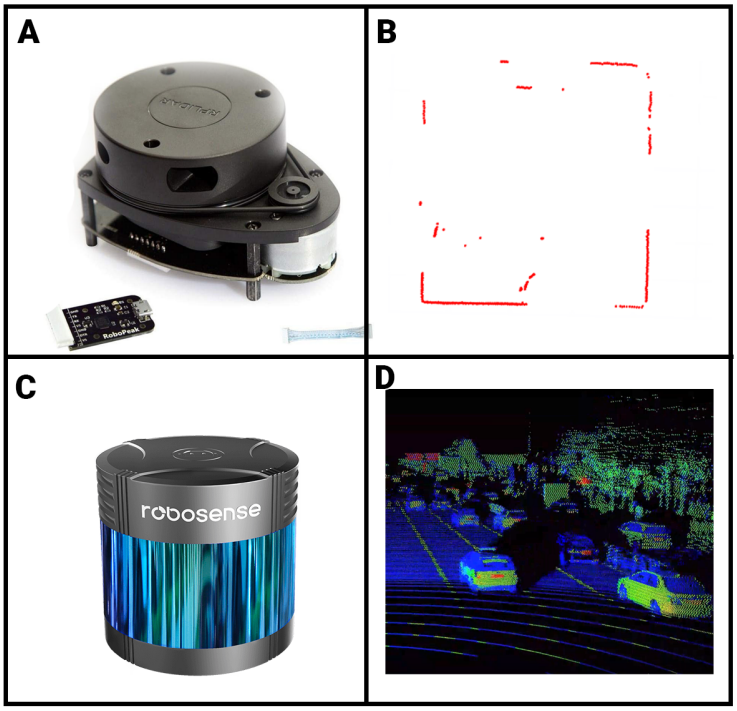

Figure 2.7: Diffrent Types of cameras used in autonomous navigation,(A) 2D RP LIDAR (B) 2D RP LIDAR PointCloud (C) 3D LIDAR (D) 3D LIDAR PointCloud

The basic equation for calculating the distance to an object using LIDAR is given by:

$$d = \frac{c \cdot t}{2} \tag{2.5}$$

Where d represents the distance to the object, c represents the speed of light, and t represents the time it takes for light to travel to the object and back to the sensor. The factor of two accounts for the fact that light traverses the distance twice: once to the object and again to the sensor.

LIDAR systems sometimes employ thousands of laser beams emitted in a scanning pattern to cover a wide field of view. Individual measurements can be merged to form a point cloud, which is a three-dimensional map of the environment.

To convert the raw LIDAR data into a point cloud, we can use the following equations:

$$x = d \cdot \cos(\theta) \cdot \cos(\phi) \tag{2.6}$$

$$y = d \cdot \cos(\theta) \cdot \sin(\phi) \tag{2.7}$$

$$z = d \cdot \sin(\theta) \tag{2.8}$$

The coordinates of the point in the point cloud are (x, y, z), the distance measured by the LIDAR system is d, the elevation angle is $theta$, and the azimuth angle is phi. These equations translate spherical coordinates $(d, theta, phi)$ into Cartesian coordinates (x, y, z).

LIDAR systems outperform conventional sensing technologies such as cameras and RADAR in various ways. They give high-resolution, three-dimensional data that can be utilized to precisely map the surroundings, detect obstructions, and locate the robot. Additionally, LIDAR systems are less impacted by lighting conditions and can function well both during the day and at night.

Both 2D and 3D LIDAR systems are important in autonomous navigation, but each has its own set of pros and disadvantages. In this section, we will look at how to use 2D LIDAR instead of 3D LIDAR for specific applications and settings.

Cost and Complexity: One of the key advantages of 2D LIDAR over 3D LIDAR is its lower cost and lower complexity. 2D LIDAR systems often have fewer components, which reduces overall cost while also making them easier to maintain and integrate into existing systems. This low-cost approach is especially appealing for applications that do not require a comprehensive 3D depiction of the environment or when money is tight.

Power Consumption: In general, 2D LIDAR systems use less power than 3D LIDAR systems. This reduced power usage is important for battery-powered robots or where energy efficiency is a priority. Reduced power consumption can also help the robot's operational time, which is especially useful in applications like logistics or warehouse automation.

Dimensions and weight: 2D LIDAR systems are often smaller and lighter than 3D systems. This feature is useful for tiny robots or when cargo capacity is limited. The reduced form factor allows for easier integration with the robot's design and reduces the platform's overall size and weight.

Data Processing: Because 2D LIDAR systems create less data than 3D systems, data processing can be simplified and the computational needs of the autonomous navigation system reduced. This reduction in data amount can result in faster processing times and potentially lower overall computational resources required by the robot, lowering costs and improving efficiency.

Terrain and Environment: The terrain and environment in which the robot operates can also influence the decision between 2D and 3D LIDAR. A 2D LIDAR system may be sufficient for navigation and obstacle identification in generally flat situations with low elevation changes, such as warehouses, factories, or other indoor settings. A 3D LIDAR system, on the other hand, would be better appropriate in more complicated situations with considerable elevation changes or where detailed 3D information is required for navigation.

While 2D LIDAR systems can give sufficient data for some applications, their inability to recognize tiny objects or navigate through complicated settings may limit their efficacy. A 3D LIDAR system would provide a more comprehensive and detailed picture of the environment in such instances, giving higher performance.

GPS

The Global Positioning System (GPS) has become a vital component for autonomous navigation in mobile robots, allowing them to detect their position and navigate through outdoor areas with relatively high accuracy in most cases. This technology makes use of a network of satellites orbiting the Earth to provide location data, which may then be combined with additional sensors and algorithms to allow for more exact navigation.

How GPS Works:

The United States Department of Defense is credited for creating the GPS, a satellite-based navigation system. It consists of a constellation of 24 to 32 medium Earth orbit satellites that provide global coverage. These satellites continuously transmit signals providing position and time information.

Ground-based GPS receivers, such as those used in mobile robots, pick up these signals and utilize the data to compute their position. A GPS receiver must receive signals from at least four satellites in order to establish its location. The distance to each satellite is calculated by the receiver using the following equation:

$$d = c \times \Delta t \tag{2.9}$$

Where d represents the distance between the satellite and the receiver, c represents the speed of light, and *Deltat* represents the time difference between when the signal was transmitted by the satellite and when it was received by the GPS receiver.

After calculating the distances to four or more satellites, the receiver employs a technique known as trilateration to estimate its position in three-dimensional space (latitude, longitude, and altitude).

GPS is important in autonomous navigation for mobile robots operating in outdoor situations. It allows robots to plan their paths, avoid obstacles, and arrive at their destination more efficiently by delivering accurate position information.

Nevertheless, due to considerations such as signal multipath, satellite geometry, atmospheric conditions, and signal blockage in urban canyons or deep forests, GPS alone may not be sufficient for reliable navigation. GPS is frequently integrated with other sensors, such as inertial measurement units (IMUs), Lidar, and vision-based systems, to improve localization and navigation accuracy. Typically, complex algorithms such as Kalman filters and particle filters are used to integrate many sensors.

Cameras

Cameras play a crucial role in autonomous navigation for mobile robots, providing visual information about the environment that enables robots to recognize objects, create maps, and plan and execute movements. This article discusses how cameras work in the context of autonomous navigation, incorporating mathematical equations to illustrate key concepts and principles.

To understand how cameras contribute to autonomous navigation, it is essential to comprehend the process of image formation. A common model used to describe this process is the pinhole camera model, which represents the relationship between 3D world coordinates and 2D image coordinates.

There are several types of cameras used in autonomous navigation, including monocular (single-lens) cameras, stereo cameras, and RGB-D cameras. Each type has its advantages and limitations, making them suitable for different applications and environments.

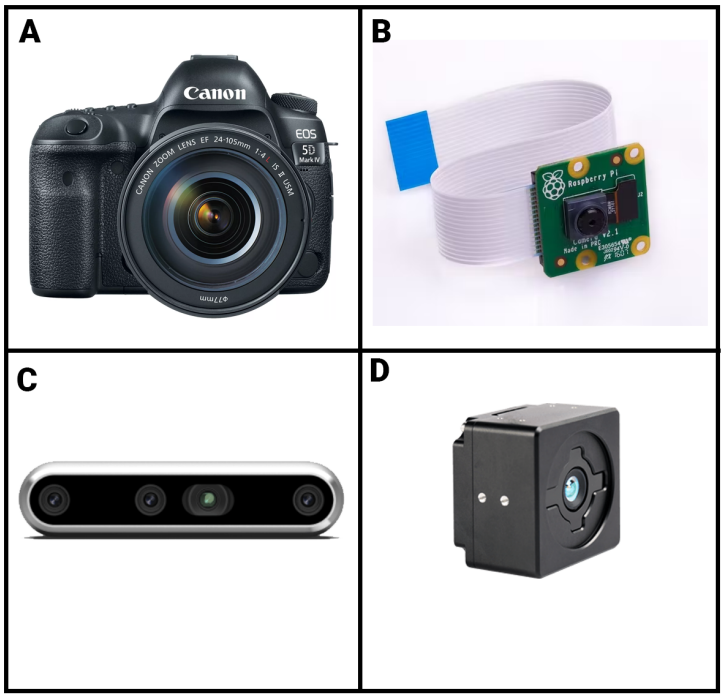

Figure 2.8: Diffrent Types of cameras used in autonomous navigation,(A) CANON EOS 5D Mark IV Monocular Camera [3] (B) Raspberry pi v4 monocular camera [4] (C) Intel D455 stereo camera [5] (D) RGB-D Camera [6]

Monocular Cameras

Monocular cameras capture 2D images and are widely used in vision-based navigation due to their simplicity, low cost, and ease of integration. They can be employed for tasks such as object recognition, scene understanding, and visual odometry. However, monocular cameras lack the ability to directly measure depth, which can be a limitation for certain applications.

Stereo Cameras

Stereo cameras use two lenses to capture images from slightly different viewpoints, simulating human binocular vision. This setup enables the calculation of depth information by comparing the disparities between corresponding points in the two images, resulting in a 3D representation of the scene. Stereo cameras are often used for tasks such as obstacle detection, mapping, and navigation in environments where depth information is crucial.

RGB-D Cameras

RGB-D cameras capture both color (RGB) and depth (D) information in a single image, typically using active sensors like time-of-flight or structured light. This type of camera provides rich data that can be used for tasks such as 3D mapping, object recognition, and pose estimation. RGB-D cameras are particularly useful in applications where accurate depth information is required, but they may have limitations in outdoor environments due to interference from sunlight.

IMU

An Inertial Measurement Unit (IMU) is an electronic device that uses a combination of accelerometers, gyroscopes, and magnetometers to detect linear acceleration, angular velocity, and, in certain cases, magnetic field strength. IMUs are considered to be a vital tool in the context of Autonomous Mobile Robots (AMRs) for calculating the robot's pose and tracking its mobility in real-time, which is required for autonomous navigation.

Accelerometers

The specific force measured by an accelerometer is the difference between the gravitational force and the inertial force acting on the sensor. An accelerometer's output can be expressed as a 3D vector:

$$\mathbf{a} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \tag{2.10}$$

To calculate the linear acceleration, subtract the gravitational acceleration from the measured acceleration:

$$\mathbf{a}_{lin} = \mathbf{a} - \mathbf{g} \tag{2.11}$$

Gyroscopes

A gyroscope measures the angular velocity of the sensor with respect to an inertial frame. The output of a gyroscope can also be represented as a 3D vector:

$$\boldsymbol{\omega} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{2.12}$$

Magnetometers

A magnetometer measures the strength and direction of the magnetic field. The output of a magnetometer can be represented as a 3D vector:

$$\mathbf{m} = \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix} \tag{2.13}$$

These measurements are combined to estimate the AMR's pose, which includes its position, orientation, and velocity. The pose can be represented as:

$$\mathbf{x} = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \\ \mathbf{v} \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ q_w \\ q_x \\ q_y \\ q_z \\ v_x \\ v_y \\ v_z \end{bmatrix} \tag{2.14}$$

where $\mathbf{p}$ is the position vector, $\mathbf{q}$ is the quaternion representing the orientation, and $\mathbf{v}$ is the linear velocity vector.

To estimate the pose from the IMU measurements, we need to integrate the measurements over time. The general equations for integration are as follows:

$$\mathbf{p}_{k+1} = \mathbf{p}_k + \mathbf{v}_k \Delta t + \frac{1}{2} \mathbf{a}_{lin,k} \Delta t^2 \tag{2.15}$$

Velocity Estimation

To estimate the linear velocity of the AMR, we need to integrate the linear acceleration measurements from the accelerometer over time. The discrete-time integration equation for velocity is:

$$\mathbf{v}_{k+1} = \mathbf{v}_k + \mathbf{a}_{lin,k}\Delta t \tag{2.16}$$

IMU in Sensor Fusion

To improve the accuracy of the pose estimation, sensor fusion techniques can be employed, combining data from the IMU with other sensors, such as wheel odometry, GPS, or cameras. The most common sensor fusion technique for AMRs is the Extended Kalman Filter (EKF) or Unscented Kalman Filter (UKF), which can be used to estimate the robot's state by fusing the IMU data with other sensor measurements.

The first step in autonomous navigation is most often, environment perception. This is typically done with a combination of sensors, such as cameras, LIDAR, and RADAR.
These sensors provide the robot with information about its surroundings, such as the location of obstacles, landmarks, and other objects. This information is then used to create a map of the environment and localize the robot within that map, allowing the planning of an obstacle-free path from a start to a goal location according to a general objective dictated by the task at hand. Any autonomous navigation system must take into consideration both static and dynamic obstacles in the environment which is an essential requirement in industrial and domestic applications. Achieving this crucial condition will ensure the safety of the robot itself as well as any humans, expensive equipment or other robots that may cross its path. Figure depicting (navigation components).
Finally, the robot must execute the plan. This is typically done using motion control algorithms, such as PID or model predictive control. These algorithms take into account the robot's current position and velocity, as well as the planned path, and generate commands to move the robot along the path. Autonomous navigation is a complex task and requires a combination of sensing, planning, and control algorithms. However, with advances in robotics technology, autonomous navigation is becoming increasingly feasible and is being used in a variety of applications. In the early days of autonomous navigation, state-of-the-art algorithms were based on simple rules and heuristics. These algorithms were used to guide robots through simple environments, As technology advanced, more sophisticated approaches were developed. These new algorithms used sensors such as cameras, LIDARs, and ultrasonic sensors to detect obstacles map the environment and localize the robot within that environment and then plan paths around them.
In the late 1990s, the development of artificial intelligence (AI) enabled the development of more advanced autonomous navigation systems. These algorithms used AI techniques such as Deep learning to enable robots to learn from their experience in the environment and plan paths accordingly. This allowed robots to navigate more complex environments, such as urban and industrial scenes.
In the last decade, the development of autonomous navigation algorithms has accelerated. Algorithms are now able to fuse data from multiple sensors to build 3D maps of their environment using fewer resources than before resulting in a wider range of AMR applications. This gave rise to interesting applications in multiple industries.
The Automobile industry was one of the biggest beneficiaries of the recent surge in Au-

tonomous driving advancement, companies like Tesla, Mercedez, Google, waymo, Toyota, Audi, and Nissan are actively developing their own electric vehicles with autonomous driving features, which increased the competition and thus reflected positively on the research scene. The latest NASA rover named Perseverance aka Percy is using a powerful auto-navigation system. Called AutoNav, which makes 3D maps of the Mars terrain, and plans a path around any obstacles without any human intervention. Another interesting application of autonomous navigation is Boston-dynamic's legged robot named spot, a mobile robot that can move around uneven terrain with unparalleled mobility. which is intended to automate typical inspection duties and data collection in a variety of environments. OTTO 1500 is one of the latest robots developed by Clear-path-robotics, designed to move heavy payloads up to 1900kg at the top speed of 2m/s through dynamic industrial environments.

2.4 Conclusion

As we reflect on the comprehensive literature review in this chapter, we revisit the significant strides made in the field of path planning and control algorithms in mobile robotics. The journey began with early graph-based path planning algorithms like Dijkstra and A*, which set a foundational base for future advancements. These methods, while revolutionary at their inception, faced challenges due to their computational intensity and limitations in dynamic, high-dimensional environments.

Recognizing these limitations, the field witnessed a shift towards more sophisticated techniques. The Potential Field Method, for instance, introduced a metaphorical approach to path planning, although it struggled in complex scenarios due to issues with local minima. This led to the emergence of sampling-based methods such as PRM and RRT, offering more efficient solutions for high-dimensional spaces but often compromising on optimality, especially in areas with narrow passages or densely placed obstacles.

The evolution continued with several variants of these techniques being developed, each addressing specific limitations of their predecessors. From memory-conserving IDA* and SMA* to speed-enhancing JPS and the dynamic-environment suited D* family, these variations greatly expanded the applicability and efficiency of A*-based methods. Similarly, sampling-based methods saw significant improvements with the introduction of FMT*, STOMP, PRM*, RRT*, Informed RRT*, and BIT*. These advancements not only enhanced path quality and computational cost but also improved adaptability in handling complex environments. Moreover, the development of Voronoi diagram-based methods and hybrid strategies added further diversity to the landscape of path-planning.

Transitioning from the theoretical advancements in path planning, the chapter delved into the practical world of Autonomous Mobile Robots (AMRs). Here, we explored the intricacies of AMR design and functionalities, shedding light on various models like Differential Drive, Ackermann Steering, and Self-balancing Robots. These models, each with their unique design considerations, play a crucial role in the practical application of the theoretical principles discussed earlier.

The aspect of autonomous navigation was a focal point, highlighting the environmental challenges AMRs face. Navigating through indoor spaces to adapting to outdoor terrains,

the role of advanced sensor technologies such as lidars, IMUs, cameras, and GPS became evidently crucial. These technologies not only facilitate precise and adaptable navigation but also signify the importance of integrating robust hardware with sophisticated software algorithms.

In synthesizing the insights from both the literature review and the exploration of AMRs, this chapter lays a robust foundation in the field of mobile robotics. It underscores the challenges that persist in developing path-planning algorithms and robotic systems capable of optimally balancing efficiency, optimality, safety, and adaptability in varied environments. The ongoing evolution in algorithm strategies, coupled with advancements in robotic design and sensor technology, is pivotal for the future of autonomous mobile robotics. Applications ranging from service robots to autonomous vehicles are just a glimpse of the potential that lies ahead.

As this chapter concludes, it is evident that the fusion of theoretical knowledge and practical applications sets a strong stage for future advancements in mobile robotics. The journey ahead promises to be filled with continued innovation and exploration, driving the capabilities of autonomous systems to unprecedented levels.

Chapter 3

An Efficient Workflow for Building Mobile Robots

Contents

3.1 Introduction

Mobile robotics represents a captivating realm where mechanics, electronics, and advanced control algorithms converge to create versatile machines capable of autonomous navigation and task execution.

In this chapter, we embark on an all-encompassing exploration of mobile robotics, focusing on the design, simulation, and construction of robotic systems. The development of such

robots entails intricate challenges, demanding a comprehensive understanding of mechanical principles, electronic components, and sophisticated control systems.

The journey begins with the design phase, where we delve into the conceptualization of robotic platforms, understanding their kinematics, and selecting suitable sensors and actuators. The integration of theoretical knowledge with practical considerations paves the way for the development of a coherent and robust robot design.

As the chapter progresses, we transition into the realm of simulation techniques, which have become invaluable tools in modern robotics research. Through simulation, we can create virtual environments that accurately emulate real-world scenarios, enabling researchers, engineers, and students to gain valuable insights into the robot's behavior and performance. In this section, we will explore the power and flexibility of simulation, leveraging the ROS-2 (Robot Operating System 2) framework and Gazebo simulation tool to validate control algorithms, optimize sensor integration, and refine the robot's responses.

Furthermore, we venture into the intricacies of the construction phase, where theoretical knowledge finds practical expression. The assembly of mechanical components, integration of electronic circuitry, and calibration of sensors culminate in the physical realization of the robot. Throughout this section, we will address the challenges faced during construction and the ingenious solutions employed to overcome them.

Throughout this chapter, we will explore the theoretical knowledge underpinning mobile robotics design, simulation, and construction, complemented by a practical example of a self-balancing robot. This hands-on illustration stands as a testament to the seamless integration of theoretical insights with practical implementation, showcasing how abstract concepts can be transformed into a tangible robotic system. As we progress, we will witness the culmination of design principles, simulation techniques, and physical construction, culminating in the realization of a fully functional self-balancing robot.

In conclusion, this chapter illuminates the multifaceted world of mobile robotics, delving into the intricacies of design, simulation, and construction. By intertwining theoretical understanding with practical demonstrations, we aspire to inspire researchers, engineers, and students to embark on their own journey of mobile robotics exploration. Through this comprehensive approach, we seek to bridge the gap between theoretical knowledge and real-world applications, fostering innovation and advancement in the captivating field of mobile robotics.

3.2 Design

The concept of robotics design transcends beyond mere engineering and programming; it is an art form in its own right. At its core, robotics design aims to create intelligent, versatile, and adaptive machines that emulate and, in some cases, surpass human capabilities. Whether it's the unyielding precision of industrial robots, the life-saving potential of medical robotics, or the promise of autonomous vehicles navigating our cities, each creation represents the amalgamation of visionary imagination, meticulous craftsmanship, and cutting-edge technology.

The design process is an intricate dance between the tangible and the intangible – an interplay of technical challenges, human needs, and futuristic aspirations. It involves striking a delicate balance between form and function, safety and efficiency, and versatility and spe-

cialization. Robotics designers navigate this terrain, shaping the future while being mindful of the ethical implications that accompany our increasingly automated world.

3.2.1 Mechanical Structure

The mechanical design of a mobile robot is a critical aspect that governs its mobility, stability, and adaptability to various terrains. Some of the key considerations in mobile robot mechanical design include:

Drive Mechanisms

Selecting appropriate propulsion systems such as wheels, tracks, or legs to suit the robot's intended environment and mobility requirements.

- *Wheel-Based Drive*: Wheeled robots are renowned for their agility and energy efficiency on smooth and flat surfaces. They can cover long distances quickly, making them ideal for applications that demand rapid movement. Furthermore, their stable base enables wheeled robots to carry relatively heavier payloads, making them well-suited for tasks involving transportation and logistics. its application includes Autonomous delivery robots in warehouses and e-commerce facilities, Automated guided vehicles (AGVs) for materials handling in factories, and Planetary rovers for space exploration missions.

- *Tracked Drive*: Tracked robots excel in traversing rough terrains where wheeled robots may encounter difficulties. Their continuous tracks provide exceptional traction, allowing them to navigate rocky surfaces or loose soil with ease. Tracked robots find extensive use in challenging outdoor environments and areas affected by disasters. and as for its application, it generally includes Search and rescue robots in disaster-stricken areas. Exploration robots in harsh terrains (forests or rocky landscapes), and Military robots for reconnaissance and transport in challenging terrains.

- *Legged Drive*: In mobile robots, legged drive mechanisms are critical for enabling versatile locomotion across a variety of terrains. These mechanisms mimic the biomechanics of animals with legs, allowing robots to navigate difficult environments that wheeled or tracked robots would be unable to navigate. To achieve stability, adaptability, and efficient motion, legged robots require intricate engineering and biomechanical considerations. Advanced sensors and algorithms are frequently used in these mechanisms to sense the ground, adjust gait patterns, and optimize foot placement. As a PhD student studying mobile robots, your dissertation could delve into the complexities of legged drive mechanisms, covering topics such as gait generation, control strategies, terrain adaptation, and the incorporation of AI and machine learning techniques to improve performance.

- *Multi-Rotor Propulsion*: Drones or UAVs (Unmanned Aerial Vehicles) are flying robots capable of aerial navigation and surveillance. They provide a birds-eye view of environments, making them useful for aerial photography, surveillance, crop monitoring,

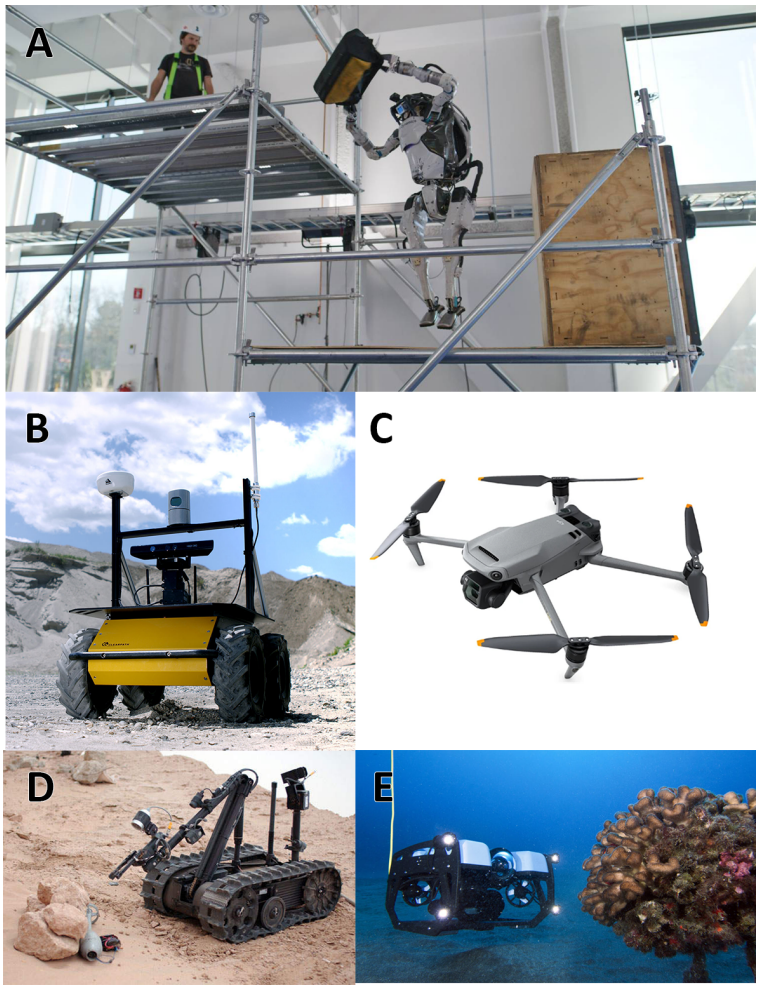

Figure 3.1: (A) Atlas by Boston Dynamics [7]. (B) Husky UGV by Clearpath Robotics [8]. (C) DJI Mavic 3 Drone [9]. (D) TALON Military Robot [10]. (E) BlueROV2 by Blue Robotics [11].

disaster assessment, and search and rescue operations. Improved drone autonomy, obstacle avoidance algorithms, energy-efficient flight patterns, and the integration of advanced sensors like LiDAR and computer vision for precise navigation could all be part of this research.

- *Swimming Drive*: Swimming robots can explore underwater areas that are difficult for other types of robots to access because they are built to operate in aquatic environments. These robots can have fins, propellers, or even biomimetic designs for effective underwater propulsion. They have uses in environmental monitoring, pipeline inspection, marine biology research, and underwater exploration. Your research might concentrate on improving swimming robots' autonomy and maneuverability, maximizing their energy effectiveness, and creating cutting-edge sensing methods for data collection and underwater navigation.

Chassis Design

Creating a sturdy and lightweight framework to house the robot's electronic components while ensuring stability during motion.

The chassis design of a robot serves as its foundational structure, holding together all the vital electronic components and mechanisms that bring the machine to life. Striking the perfect balance between sturdiness and lightweight construction is paramount to achieve optimal performance, efficiency, and agility in the robot's movements.

The choice of materials plays a pivotal role in crafting a reliable and durable chassis. Different materials offer unique properties that can influence the robot's overall performance, weight, strength, and cost. Below are some common materials used in robot chassis design and their reasons for usage or avoidance:

- *Aluminum:* Aluminum is a popular choice due to its excellent strength-to-weight ratio, making it lightweight yet sturdy enough to endure various stresses. Its high corrosion resistance ensures durability, especially in outdoor environments. However, aluminum can be relatively expensive compared to other materials.

- *Carbon Fiber:* Carbon fiber is an exceptional choice for lightweight chassis design. It is incredibly strong and boasts a low density, offering unparalleled weight reduction. This results in increased energy efficiency and better agility. However, carbon fiber can be quite expensive and may require specialized manufacturing processes.

- *Titanium:* Titanium combines high strength with relatively low weight, making it an ideal material for robot chassis that require exceptional durability. It offers excellent corrosion resistance and is well-suited for applications in harsh environments. However, titanium is one of the most expensive materials, which can significantly impact the overall cost of the robot.

- *Steel:* Steel is a traditional choice for chassis construction due to its affordability and high strength. It is readily available and can withstand considerable forces, making it suitable for heavy-duty robots. However, steel is relatively heavy compared to other materials, which can affect the robot's mobility and energy consumption.

- *High-Strength Polymers:* Advanced polymers, such as reinforced nylon or polycarbonate, offer a balance between strength and weight. They are cost-effective, easy to manufacture, and provide good resistance to impact and vibrations. However, their mechanical properties might not match those of metals like aluminum or titanium.

Suspension and Articulation:

Incorporating suspension systems or articulated segments is a crucial aspect of chassis design that can significantly enhance the robot's ability to traverse uneven surfaces and overcome obstacles. These features enable the robot to maintain better stability, traction, and maneuverability, particularly in challenging environments. Below are the key considerations for implementing suspension and articulation:

- *Suspension Systems:* Suspension systems consist of dampers, springs, or other mechanisms designed to absorb shocks and vibrations. By allowing the robot's wheels or tracks to move independently from the chassis, suspension systems can adapt to uneven terrain and maintain better contact with the ground. This translates to improved traction, reduced slippage, and increased stability, especially when navigating rough terrains, stairs, or rocky surfaces.

- *Active vs. Passive Suspension:* Suspension systems can be either active or passive. Active suspensions use sensors and actuators to adjust the suspension's response based on real-time feedback, optimizing performance for different conditions. Passive suspensions rely on mechanical design and are simpler, more robust, and cost-effective. The choice between active and passive suspension depends on the robot's complexity, budget constraints, and desired level of adaptability.

- *Articulated Segments:* In some robotic applications, incorporating articulated segments into the chassis design can greatly enhance the robot's mobility. Articulated segments allow the robot to bend or rotate at specific points, providing enhanced maneuverability in tight spaces and complex terrains. For instance, snake-like robots can slither through narrow passages, and quadruped robots can navigate uneven ground by flexing their legs independently.

- *Torsional and Linear Articulation:* Articulated segments can offer either torsional (rotational) or linear (translational) motion. Torsional articulation provides rotational flexibility, while linear articulation enables the extension and retraction of specific segments. A combination of both types of articulation can create versatile robots capable of adapting to a wide range of environments and tasks.

- *Control and Stability:* Implementing suspension and articulation introduces additional complexity to the robot's control system. Proper control algorithms and feedback mechanisms are essential to maintain stability and prevent erratic behavior. Advanced sensors, such as gyroscopes and accelerometers, can provide real-time data to help the robot adapt its suspension and articulation based on its surroundings and motion requirements.

Size and Form Factor

One of the fundamental considerations in developing mobile robots is determining the appropriate size and form factor for the robot. This crucial factor has a direct impact on the robot's capabilities, maneuverability, and overall performance in its intended environment. The size of a mobile robot is a delicate balance of multiple factors, including the application for which it is designed and the constraints imposed by the environment in which it will operate. In environments with confined spaces or narrow passages, for example, a compact robot with a streamlined form factor may be required to ensure effective navigation and operation. Tasks that require heavy lifting or extensive sensor arrays, on the other hand, may necessitate a larger form factor to accommodate the necessary equipment.

CAD design of a Self-Balancing Robot

In this part, we use Onshape, a potent and versatile CAD software, to demonstrate the critical design process of a self-balancing robot chassis [12]. The cloud-based nature of Onshape eliminates the necessity for resource-intensive installations, thus promoting efficient collaboration. It allows users to access and work on projects from any location. Its robust version control system makes it easy to track design alterations and provides a trace for decision-making and troubleshooting. Onshape, with its capability to streamline the design process and support diverse file formats, including STL meshes for simulations, is a flexible choice for various projects.

The robot chassis comprises three shelves (top, middle, and bottom) and two side plates that secure the shelves in place. The chassis design doesn't include sensors or other electronic components; instead, the motors and wheels were obtained from the Onshape public library.

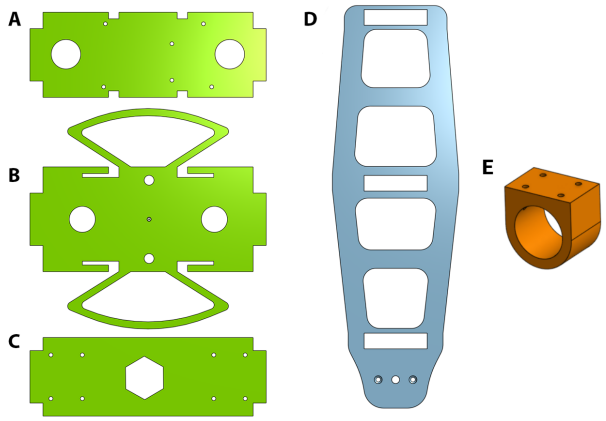

Figure 3.2: Chassis components of the self-balancing robot: (a) Top shelf designed for mounting the microcontroller board, (b) Middle shelf with bumpers for accommodating the batteries, (c) Bottom shelf designed for housing the motors, with motor mounting holes and wire pass-throughs, and (d) Side plate used for holding the three shelves in place.

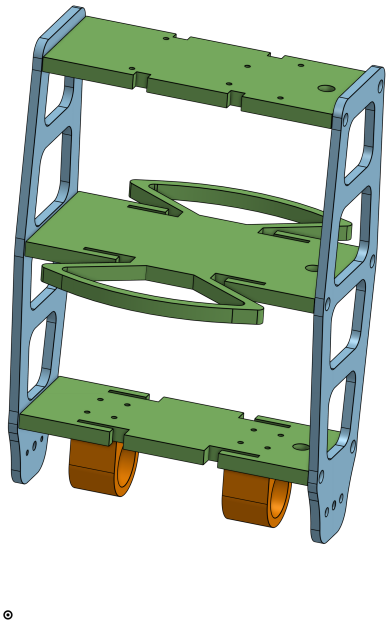

Figure 3.3: Assembled self-balancing robot in Onshape (no wheels)

We started the design process by creating a new Onshape document and importing the desired motors and wheels from the public library. Next, a new assembly was created to integrate the chassis components and the imported motors and wheels.

The chassis plates were designed in a new Part Studio, where sketches were drawn for the three shelves and the two side plates. Each shelf has a specific purpose and features tailored to its function: The top shelf is designed for mounting the microcontroller board (Figure 3.8a). The middle shelf accommodates the batteries and features bumpers to absorb shocks, enhancing the robot's durability (Figure 3.8b). The bottom shelf houses the motors and includes holes for motor mounting and wire pass-throughs between the chassis stages (Figure 3.8c). The side plates, shown in Figure 3.8d, hold the shelves in place.

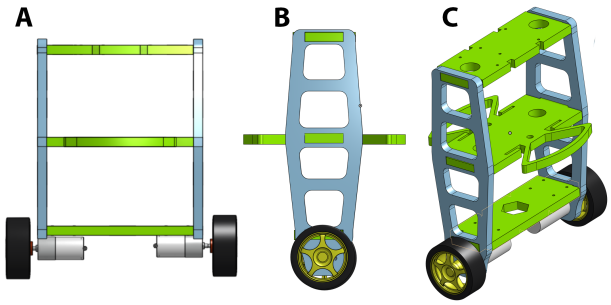

Figure 3.4: Assembled self-balancing robot in Onshape assembly: (a) Front view, (b) Side view, and (c) 3D view, illustrating the integration of the chassis components, motors, and wheels..

After completing the design of the chassis plates, we used the assembly to assemble the chassis and imported components (Figure 3.6). The 'Mate' tool was used to assemble the components, with the three shelves being mated to the side plates. The motors were mated to the bottom shelf, and the wheels were mated to the motor shafts.

Additional features such as mounting holes or other customizations could be added to the chassis plates by editing the sketches or creating new sketches and extrusions in the Part Studio, if necessary.

Finally, we exported the chassis parts as STL meshes for use in simulation. Each plate was exported from the Part Studio in STL format, a widely-used file format for 3D models in simulation and additive manufacturing.

3.2.2 Electrical Structure

Sensing Mechanisms

Sensors play a pivotal role in providing mobile robots with the perception needed to navigate and interact with their surroundings. Key sensing mechanisms in mobile robotics design include:

- *Lidar (Light Detection and Ranging):* Utilizing laser-based sensors to create 2D or 3D maps of the robot's surroundings, enabling accurate localization and obstacle detection.
- *Inertial Measurement Units (IMUs):* Combining accelerometers and gyroscopes to measure the robot's orientation and motion, aiding in motion planning and control.
- *Camera Systems:* Equipping the robot with cameras for visual perception, enabling tasks such as object recognition, path following, and environment understanding.

Actuation Methods

Actuators are essential for mobile robots to convert electrical signals into mechanical motion and enable their locomotion and manipulation capabilities. Common actuation methods in mobile robotics design include:

- *DC Motors:* Utilizing DC motors for driving wheels or tracks, providing efficient and precise control over the robot's movement.
- *Steering Mechanisms:* Incorporating mechanisms for differential steering or Ackermann steering to facilitate agile turning and maneuvering.
- *Leg Actuators:* Employing actuators with dynamic and adaptive capabilities in legged robots for versatile locomotion across challenging terrain.

Power and Energy Management

Mobile robots require efficient power systems and energy management to prolong their operational time and ensure sustained performance. Considerations in this aspect include:

- *Battery Systems:* Choosing appropriate battery types and capacities to meet the robot's power requirements while considering weight and runtime constraints.
- *Energy-efficient Electronics:* Opting for low-power components and optimizing algorithms to reduce energy consumption during operation.
- *Charging and Docking Systems:* Designing systems for autonomous charging or docking to enable the robot to recharge when needed.

Our self-balancing robot's creation relies heavily on a deep understanding and merging of two main areas: Electrical Design and Mechanical Design. The Electrical Design involves the choice and setup of several electronic elements such as microcontrollers, sensor modules, motor drivers, and potentiometers. These are all harmonized through a custom-made printed circuit board (PCB). Conversely, Mechanical Design is about the creation of a robust yet light chassis from PVC foam board. This design uses an innovative, screw-less approach and takes advantage of CNC machining for cost-effectiveness and precision. Both areas, while having different tasks and goals, collaborate towards the same end - to build a highly efficient, operational, and sturdy self-balancing robot.

Electrical Design of a self balancing robot

Our robot's electrical system [13] is a mix of intricate parts like an Arduino Mega microcontroller board, an MPU 6050 accelerometer-gyroscope module, an L298N motor driver module, and three potentiometers for tuning of the PID controller parameters. The MPU 6050 module serves as the primary sensor for detecting the robot's tilt angle, while the L298N motor driver module efficiently powers the two DC motors, actuating the robot's wheels as needed.

Connecting these components has historically been challenging, leading to a confusing web of wires. To overcome this, we intentionally designed and employed a small-footprint PCB to expedite assembly and maintenance.

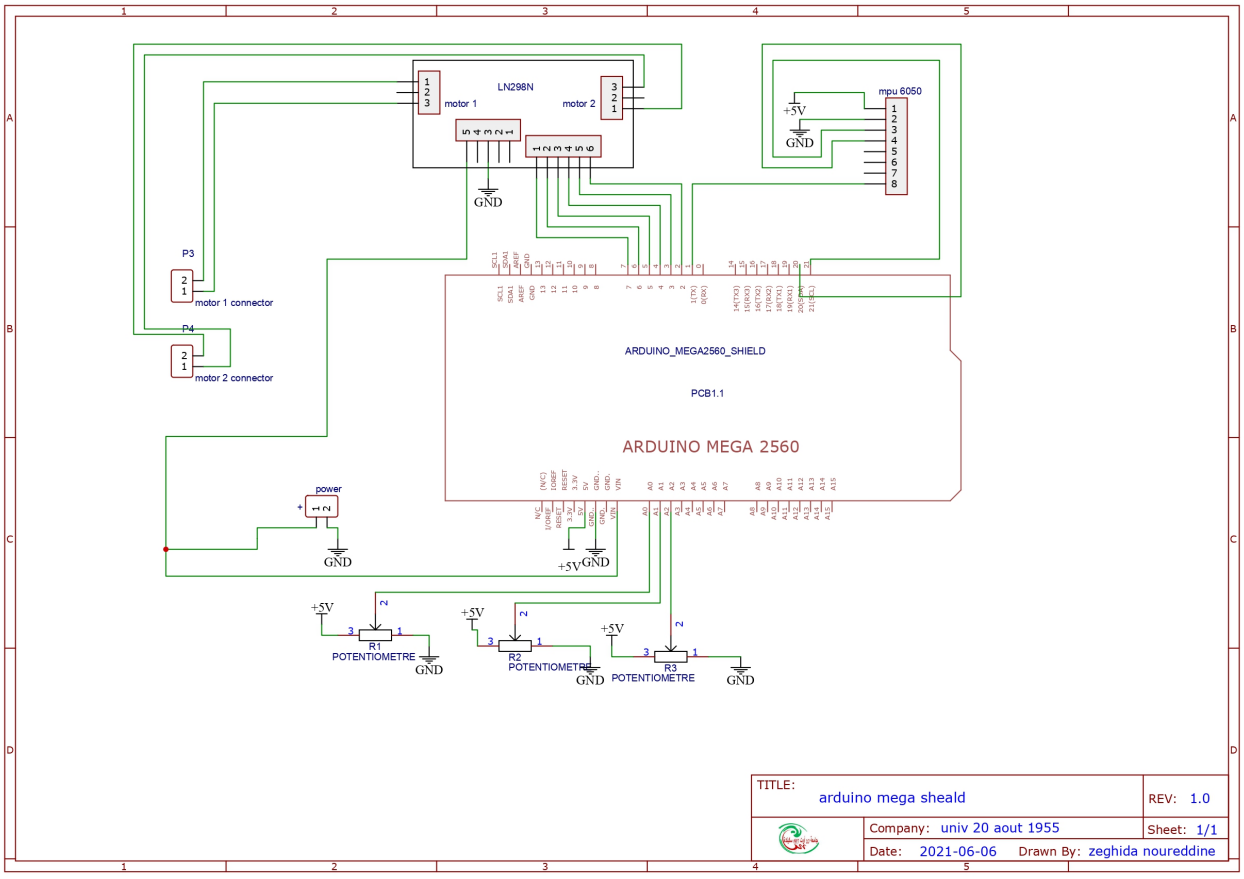

Figure 3.5: EasyEDA schematic of the Printed Circuit Board (PCB) designed for the self-balancing robot. The schematic illustrates the arrangement and interconnection of components, highlighting the compact and efficient design.

The PCB provides solid mechanical support and secure electrical connections for the electronic components through a network of conductive tracks and pads. It is typically layered with two copper layers laminated onto a non-conductive substrate.

3.3 Simulation

Simulation is a key component in the development and validation of mobile robot systems because it provides a dynamic and controlled environment for exploring and refining various aspects of mobile robot behavior. By creating lifelike virtual environments, Simulations allow us to comprehensively assess the performance of our design, rigorously test algorithms, and fine-tune navigation methods long before the robot even takes its first steps in the real world. This subsection provides a thorough examination of the simulation process, illuminating the fundamental components and methodologies that underpin the art of simulating mobile robots.

The importance of simulation in mobile robot development cannot be overstated. It

provides a risk-free environment for testing new ideas, optimizing existing algorithms, and confronting difficult scenarios without risk of physical harm or resource expenditure. By leveraging the power of simulations, we can avoid potential pitfalls, refine designs, and ensure that our robots are ready to face the complexities of the real world with a higher level of preparedness.

Simulating the behavior of sensors and actuators to provide a faithful representation of real-world interactions is a critical aspect of simulation. We can evaluate perception algorithms in a variety of scenarios by simulating sensor outputs such as camera images, LIDAR scans, and ultrasonic measurements. We can gauge how control strategies respond to different inputs and assess their efficacy in driving the robot's movement by simulating actuator actions such as motor movements and joint rotations.

Simulation is a fertile ground for evaluating and improving algorithms that govern various aspects of robot behavior. Path planning, obstacle avoidance, localization, and mapping algorithms can all be rigorously tested in simulated environments, allowing us to assess their performance across a wide range of scenarios. This step is critical in refining these algorithms, tailoring them to specific challenges, and increasing the mobile robot's overall efficiency.

The effectiveness of mobile robots in simulation can be measured using a variety of metrics and performance indicators. We can quantitatively measure the robot's behavior under various conditions by defining appropriate metrics such as navigation accuracy, completion time, energy consumption, and collision avoidance rate. These metrics provide actionable insights into the robot's design and algorithms' strengths and limitations, guiding iterative improvements.

Simulation is an indispensable tool at all stages of the development lifecycle. Simulation provides a versatile framework that accelerates the iterative process from the initial design phase to algorithm development, optimization, and testing. Furthermore, simulation can help train machine learning models for perception and control, allowing robots to learn from virtual experiences and apply what they've learned to real-world scenarios.

3.3.1 Self Balancing Robot simulation

Self-balancing robot design and construction can be challenging and complicated, requiring knowledge of mechanics, electronics, and control systems. Moreover, creating and testing a real robot prototype can be expensive and time-consuming. Therefore, simulating self-balancing robots before fabrication can speed up development and cut expenses while also saving time.

Recent advancements in the design and simulation tools have made it much easier and more accessible to students, engineers and researchers to design and simulate self-balancing robots. In particular, the use of the ROS-2 (Robot Operating System 2) framework and the Gazebo (a robot simulation tool), provided a powerful and highly flexible platform for designing and simulating complex robotic systems.

Self-balancing robots offer a wide range of practical uses. They can be utilized in the surveillance field to keep an eye on vital infrastructure or support search and rescue efforts. They can be utilized in the delivery sector to move items rapidly and effectively, particularly in congested urban locations. Another potential use for self-balancing robots is entertainment, such as in ride-on toys or live performances.

SDF

To create a Gazebo simulation, we first need to generate an SDF model of the robot based on the STL meshes exported from the Onshape design phase. This model comprises three main links: one for the chassis and one each for the left and right wheels and motors. These links form the robot's basic structure, enabling simulation in the Gazebo environment.

To simulate the motion of a self-balancing robot in Gazebo, we must define the joints between the chassis and the wheels/motors. This is generally done using revolute joints, which permit the wheels to rotate about an axis perpendicular to the chassis plane.

Adding joints to the SDF file provides several benefits, including accurate simulation of robot movement and components, testing different control algorithms, and tuning parameters such as PID gains. Defining joints in the SDF file also allows Gazebo's built-in physics engine to simulate the robot's dynamics, including collisions and interactions with other objects in the environment.

We incorporated a differential drive plugin within the SDF model to control the robot's motor movements. This plugin is a widely-used tool for controlling robots with two driven wheels, converting linear and angular velocity commands into individual wheel speed commands. By using this plugin, we can simulate the robot's movement and steering capabilities, offering valuable insights into its real-world performance.

ROS

Our self-balancing robot uses a PID controller to maintain balance. The controller continuously monitors the robot's orientation, compares it to the desired upright position (setpoint), and applies corrections to the robot's wheel speeds accordingly.

To improve the tuning process and provide a more visual and interactive experience, we developed a user-friendly interface using PyQt, a set of Python bindings for Qt libraries commonly used for creating graphical user interfaces (GUIs). This interface enables real-time adjustment of PID gains (Kp, Ki, Kd) and observation of how these changes impact the robot's balance and stability. This hands-on, interactive approach significantly simplifies the tuning process, making it less time-consuming and more intuitive.

Additionally, to ensure the repeatability of optimal performance and save future tuning time, we incorporated functionality to save and load PID gains from a file. This feature allows us to store optimal gain settings found during the tuning process and quickly restore them when needed. By enabling the file to be read and written directly through our PyQt interface, we ensure a seamless user experience for operating the self-balancing robot.

We created a launch file to streamline the process of launching the Gazebo simulation and configuring the necessary parameters. This launch file sets up the simulation environment, initializes the robot model, and loads the differential drive plugin and PID controller configurations.

Test and tunning

By running this launch file, we can swiftly start the simulation

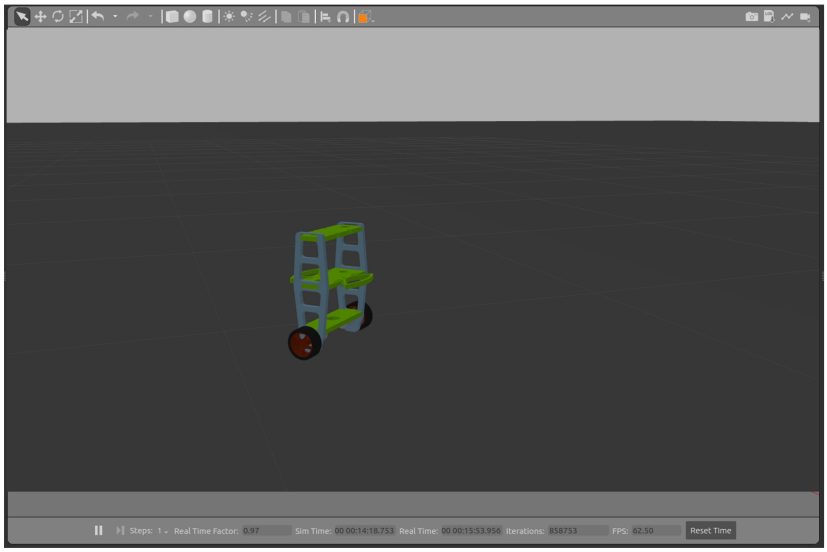

Figure 3.6: Our self balancing robot simulated in a Gazebo environment

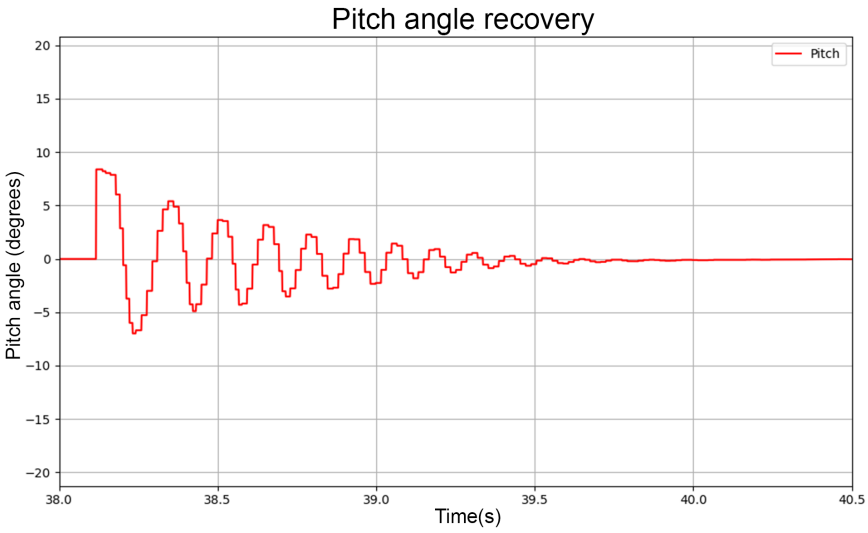

Figure 3.7: Pitch Angle Response of the Self-Balancing Robot. The figure shows the robot's ability to return to its desired angle after being displaced by $\approx 8.45°$ degrees. The swift and accurate response illustrates the effectiveness of the manually tuned PI controller with gains P=0.45 and I=0.55.

The optimization of the self-balancing robot's performance relied significantly on the tuning of the PID controller paramaters. This process was made easier with the PyQt interface, allowing us to make on-the-spot adjustments to the PID gains and providing immediate feedback on how these changes influenced the robot's balance and stability.

The PID controller works to maintain balance by continually monitoring the robot's orientation, comparing it with the desired upright position (the setpoint), and making necessary

corrections to the wheel speeds. The PID gains (P, I, D) dictate the intensity of these corrections.

The tuning procedure started with all gains at zero. The proportional gain (P) was first increased until the robot could sustain balance for short intervals but with noticeable oscillation. Subsequently, the integral gain (I) was gradually raised to decrease the oscillation and enhance the robot's stability. Finally, the derivative gain (D) was adjusted. However, for our system, it was determined that a derivative gain of zero was optimal.

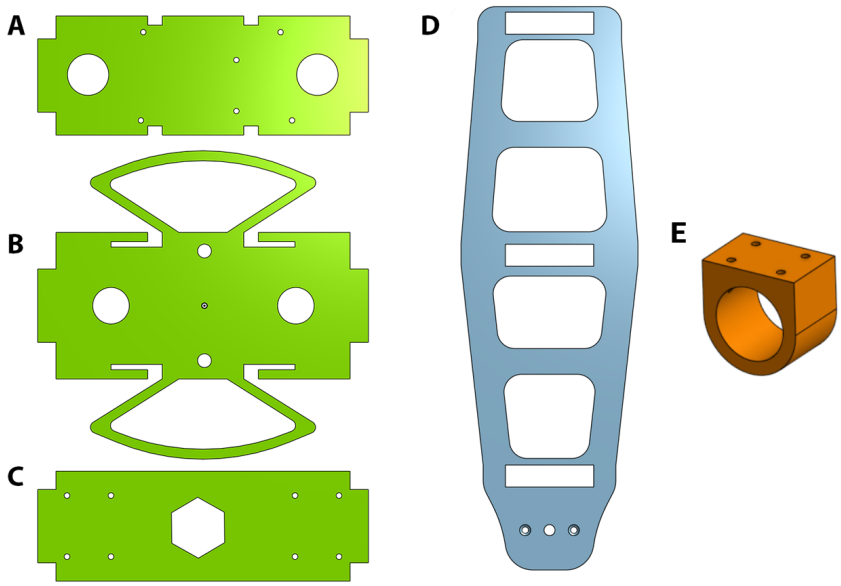

Figure 3.8: Chassis components of the self-balancing robot: (a) Top shelf designed for mounting the microcontroller board, (b) Middle shelf with bumpers for accommodating the batteries, (c) Bottom shelf designed for housing the motors, with motor mounting holes and wire pass-throughs, and (d) Side plate used for holding the three shelves in place. [12]

As a result, the optimal control system for our self-balancing robot is a PI controller, with gains P=0.45 and I=0.55. This set of values allowed the robot to maintain balance effectively while reducing overshoot and oscillation. Moreover, the robot exhibited a robust response to a displacement of approximately 10 degrees angle from the desired angle. The pitch angle response, illustrated in the included figure, clearly showcases the robot's capability to return to the desired angle swiftly and accurately.

The tuning process of the PID controller parameters played a pivotal role in enhancing the performance of our self-balancing robot. By tweaking the PID gains in real time using the PyQt interface, we identified the optimal PI control system for our robot. However, it should be noted that additional tuning may be required to fine-tune the robot's performance under different conditions or with changes in the robot's design.

3.4 Software development

Software for mobile robots frequently uses a control architecture that includes data collection from sensors, state estimation, decision-making, and actuation. The feedback control loop, which gives robots the ability to react in real time to their surroundings, is a key component of this design. This loop's main elements are:

- *Sensor Data Acquisition*: To learn about their environment, mobile robots use a variety of sensors, including cameras, LiDAR, inertial measurement units (IMUs), and ultrasonic sensors.
- *State estimation*: Estimating the robot's location, orientation, and velocity involves processing sensor data. For making wise selections, this assessment is essential.
- *Decision Making*: Control algorithms are used to decide the robot's behaviors based on the estimated state and stated objectives. These algorithms range from straightforward rule-based systems to sophisticated machine-learning models.
- *Actuation*: The robot's actuators, which might include wheels, motors, and manipulators, receive the control signals created by the decision-making process and carry out the intended actions.

Mobile robots frequently make use of sensor fusion and filtering techniques to improve the precision and dependability of sensor data. These techniques decrease noise and mistakes while combining data from many sensors. Kalman filtering and complementary filtering are typical methods for sensor fusion.

While the general software development concepts for mobile robots are still relevant, our attention is now focused on the particular software development elements of a self-balancing robot. A feedback control loop is used by the self-balancing robot to maintain balance. This loop consists of:

- *Sensor Data Acquisition*: The MPU6050 sensor gives real-time information on the pitch, roll, and angular velocities of the robot.
- *State estimation*: Information from the MPU6050 is analyzed to determine the tilt angle and angular velocity of the robot in order to determine its present state.
- *Error Calculation*: To calculate the error, the ideal pitch angle—which is typically set at 0 degrees—is contrasted with the predicted pitch angle.
- *Controller Algorithm*: The appropriate motor control signals are calculated using a proportional-integral-derivative (PID) controller based on the error, maintaining balance.

3.4.1 Aquiring IMU data using Arduino board

An Inertial Measurement Unit (IMU) sensor with a 3-axis gyroscope and 3-axis accelerometer on a single chip, the MPU-6050, is a popular and often-used IMU. It is a vital component for a variety of applications, including robots, drones, gaming controllers, and wearable technology since it offers crucial information regarding motion, direction, and acceleration. The following procedures must be followed in order to read the inclination angle from the MPU-6050 sensor using Arduino:

- Use the I2C interface to connect your Arduino board to the MPU-6050 sensor.
- Connect the SDA and SCL pins of the sensor to the relevant Arduino pins (often A4 and A5 on an Arduino Uno).
- Connect VCC and GND as well.
- Install the "Wire" library (for I2C connection) and the "MPU6050" library, which is a well-liked library for interacting with this sensor, to make communication with the MPU-6050 easier. Using the Library Manager in the Arduino IDE, you may install these libraries.

```
#include <Wire.h>
#include <MPU6050.h>

MPU6050 mpu;

void setup() {
  Wire.begin();
  Serial.begin(9600);

  mpu.initialize();
  // You can configure accelerometer sensitivity here if needed.
}

void loop() {
  int16_t ax, ay, az;
  mpu.getAcceleration(&ax, &ay, &az);

  // Print the raw accelerometer data
  Serial.print("Acceleration (mg): ");
  Serial.print("X="); Serial.print(ax);
  Serial.print(", Y="); Serial.print(ay);
  Serial.print(", Z="); Serial.println(az);

  delay(10); // Adjust the sampling rate as needed
}
```

3.4.2 Arduino control of DC motors

Self-balancing robots rely on motors as their essential structural support because they are in charge of converting control signals into motion. The two 12V DC motors in our self-balancing robot are crucial in preserving equilibrium since they regulate the speed and direction of the wheels. Self-balancing robots rely on dynamic balance, in which the robot's posture is continually adjusted by its motors to keep it from toppling over. By accurately adjusting the motors' rotational speed in response to sensor inputs, this dynamic equilibrium is attained. The robot can react quickly to changes in its orientation thanks to efficient motor control. Precision motor control is essential for minimizing disruptions and preserving a straight posture.

As a key method for regulating the rotational speed of DC motors in self-balancing robots, pulse width modulation (PWM) develops. The average voltage supplied to the motors is successfully regulated using PWM, which produces a pulsating signal with a varied duty cycle. We may manage the effective voltage delivered to the motors and, therefore, their speed by varying the duty cycle of the PWM signal. This exact control is crucial for optimizing the corrective steps performed to preserve balance in the self-balancing robot.

High-level programming is made possible by Arduino's user-friendly development environment, which makes it easier to build complicated motor control algorithms. This programming simplicity is very useful for developing and optimizing control systems. Furthermore, robots that can balance themselves must be able to react instantly to sensor input. The microcontroller design of the Arduino and its capacity for accurate timing make it an excellent choice for quick processing of sensor data and motor control modifications.

The following arduino sketch snippet shows PWM control of a DC motor using a L298N motor driver

```
// Defining motor control pins
const int enablePin = 9;  // PWM pin for speed control
const int in1Pin = 8;      // L298N IN1 pin
const int in2Pin = 7;      // L298N IN2 pin

void setup() {
  pinMode(enablePin, OUTPUT);
  pinMode(in1Pin, OUTPUT);
  pinMode(in2Pin, OUTPUT);
}

void loop() {
  // Setting the motor speed using PWM (0 to 255)
  int speed = 150;  // Adjust speed as needed
  analogWrite(enablePin, speed);

  // Setting the motor direction (clockwise or counterclockwise)
  digitalWrite(in1Pin, HIGH);
  digitalWrite(in2Pin, LOW);
```

```
20
21   // Applying a delay to maintain the motor speed
22   delay(1000);
23 }
```

3.4.3 Self balancing robot arduino sketch

The main control loop of the program is in charge of maintaining the robot's equilibrium in real time. It computes the motor control signal while continually monitoring sensor data. We utilize the PID controller to carry out these calculations within this loop.

The loop also manages the detection of available sensor data and makes sure the PID control runs well. It also includes extra activities, such as those that take place at particular intervals of time, such 1Hz and 5Hz. These activities might entail gathering information for logging, modifying PID tuning settings, or carrying out other duties necessary for the robot's operation.

In this chapter, our primary objective is to demonstrate an efficient workflow for developing a mobile robot rather than focusing on the fine-tuning and optimization of the software itself. To achieve this goal, we will leverage a codebase that has been primarily developed by Luka Gabrić and generously shared on his GitHub repository [100]. This existing codebase serves as a valuable foundation for our work and enables us to streamline the development process of the mobile robot.

For the robot to maintain its equilibrium, accelerometer and gyroscope measurements from the MPU6050 are essential. In this stage, we take the sensor signals and extract pertinent information, including the robot's pitch angle. This data is used by the PID controller to determine the proper motor control signal:

- *Initialization of the Sensor and PID*: This section of the code focuses on initializing crucial parts. The MPU6050 sensor is initialized with the mpu.initialize() method, which is essential for receiving crucial information regarding the direction of the robot. Additionally, we configured the PID controller to work in automatic control mode using pid.SetMode(AUTOMATIC), allowing it to continually alter the robot's balance.

```
1 mpu.initialize();  // Initialize MPU6050 sensor
2 pid.SetMode(AUTOMATIC);  // Initialize PID controller in
     AUTOMATIC mode
```

- *Primary Control Loop*: The code's beating heart is the main control loop. It computes the control signal for the robot's motors and continuously examines sensor data. The pid.Compute() method determines the required motor changes within this loop in order to reduce the discrepancy between the desired (setpoint) and actual (input) robot angles. the motors are functionally controlled by the motorController.move() method, which converts the PID output into motor actions.

```
1 while (!mpuInterrupt && fifoCount < packetSize) {
2     pid.Compute();  // Perform PID calculations
```

```
    motorController.move(output, MIN_ABS_SPEED);  // Control
       motors
    // Additional actions at specific time intervals
    // ...
}
```

- *Handling Sensor Data* :The processing of MPU6050 sensor data is the main topic of this section. The accelerometer and gyroscope measurements in the sensor data are essential for calculating the robot's pitch angle. The PID controller's feedback loop makes use of the determined pitch angle (input) to make sure the robot stays balanced.

```
mpu.dmpGetQuaternion(&q, fifoBuffer);
mpu.dmpGetGravity(&gravity, &q);
mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
input = ypr[1] * 180/M_PI + 180;  // Compute robot's pitch
   angle
```

- *Optional PID Tuning*: This section of the code enables manual adjustment of the PID settings if necessary. In order to change the proportional (kp), integral (ki), and derivative (kd) gains of the PID controller, it contains methods like setPIDTuningValues() and readPIDTuningValues() that read values from potentiometers. The balancing behavior of the robot may be improved by manual modification depending on testing and optimization needs.

```
void setPIDTuningValues() {
    readPIDTuningValues();
    if (kp != prevKp || ki != prevKi || kd != prevKd) {
        pid.SetTunings(kp, ki, kd);  // Update PID parameters
        // ...
    }
}
```

3.5 Construction and Testing

The construction and testing phase is critical in the development of mobile robots because it transforms theoretical designs into tangible machines. This section delves into the painstaking process of assembling and validating physical robots to ensure that they meet design specifications and perform reliably in real-world scenarios. The construction phase begins with the assembly of hardware components. This section describes how mechanical parts, electronic modules, sensors, actuators, and power sources are integrated in a systematic manner. It goes over the difficulties and considerations that must be made when connecting intricate components, ensuring proper alignment, and securing them together to form a cohesive robotic platform.

The process of translating design schematics into physical structures is known as mechanical fabrication. This section describes the techniques used to create robot chassis, frames, joints, and mechanisms. with an emphasis on precision and material selection to achieve the desired balance of weight, durability, and performance.

Sensor calibration is critical for achieving accurate perception. This section delves into the complexities of sensor calibration, addressing techniques for mitigating biases, accounting for noise, and ensuring consistent sensor readings. It discusses the incorporation of calibrated sensors into the framework of the robot, as well as the harmonization of sensor outputs to provide a reliable and coherent perception of the environment.

The driving force behind robot movement is provided by actuators. This section goes over how to configure and control actuators like motors, servos, and pneumatic systems. It discusses methods for fine-tuning control parameters, calibrating movement ranges, and synchronizing actuation with sensory inputs, resulting in precise and responsive robot motion.

During the software integration phase, various software modules are intertwined to form a functional robot. This section delves into programming the control algorithms, communication protocols, and perception systems of the robot. It goes over how to integrate ROS (Robot Operating System) nodes, API calls, and real-time communication strategies to allow for seamless interaction between hardware components.

The validation of the built robot is an important step in ensuring that its performance matches the design objectives. This section describes the thorough testing procedures used to assess navigation accuracy, obstacle avoidance, sensor fusion, and overall system behavior. It discusses controlled experiments, scenario-based tests, and stress tests used to identify potential limitations and refine algorithms.

The building and testing phases are iterative, driven by feedback and insights gained from test results. This section discusses how testing results guide iterative improvements, algorithm refinement, and hardware configuration changes. It emphasizes the cyclical nature of development, in which each testing cycle improves the performance and capabilities of the robot.

3.5.1 Electrical part

We [13] reached out to SNC Almitech PCB Company, an Algerian firm, to manufacture our carefully designed PCB. After about ten days, we received the PCB via Kazi Tour's quick delivery service.

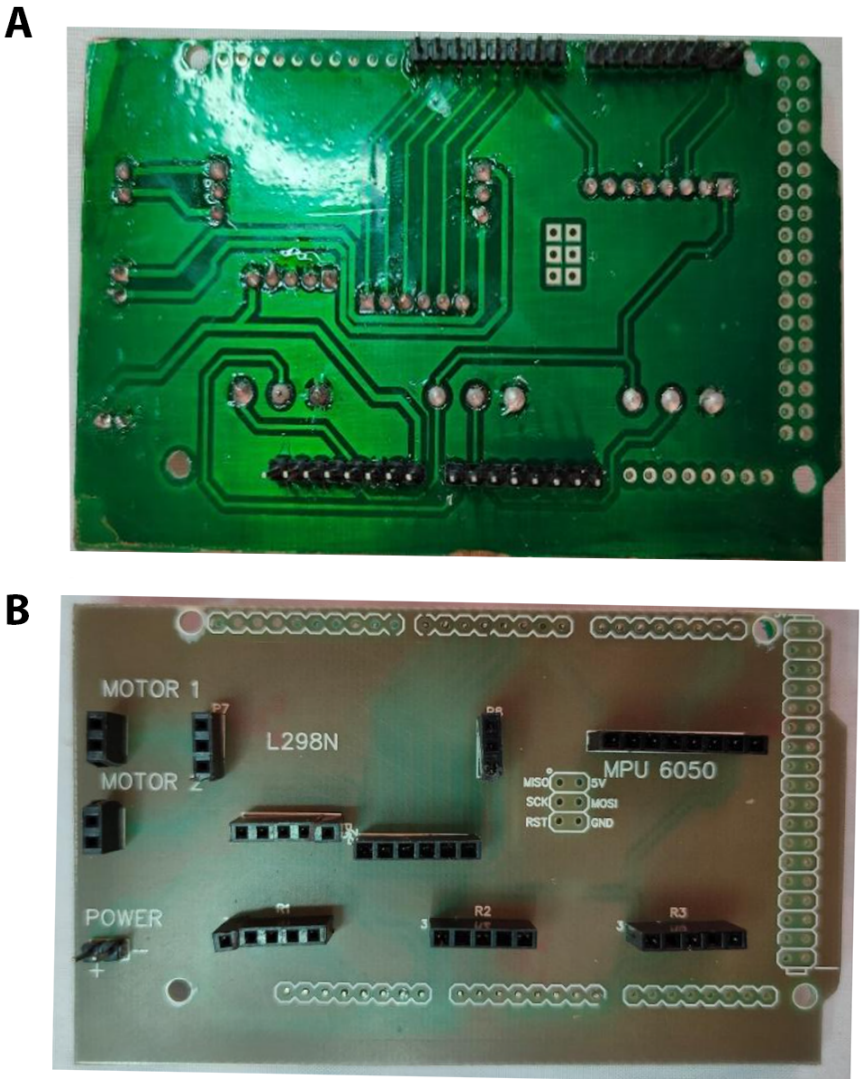

Figure 3.9: (a) Front face of the manufactured Printed Circuit Board (PCB) for the self-balancing robot, illustrating the arrangement of components and connectors. (b) Back face of the PCB, showcasing the conductive tracks providing the electrical connections.

Upon receipt, the PCB went through a detailed inspection to check for any potential issues that could affect its operation. After this quality control step, we mounted the electronic components onto the PCB using pre-soldered header pins, simplifying the process and enabling quick assembly.

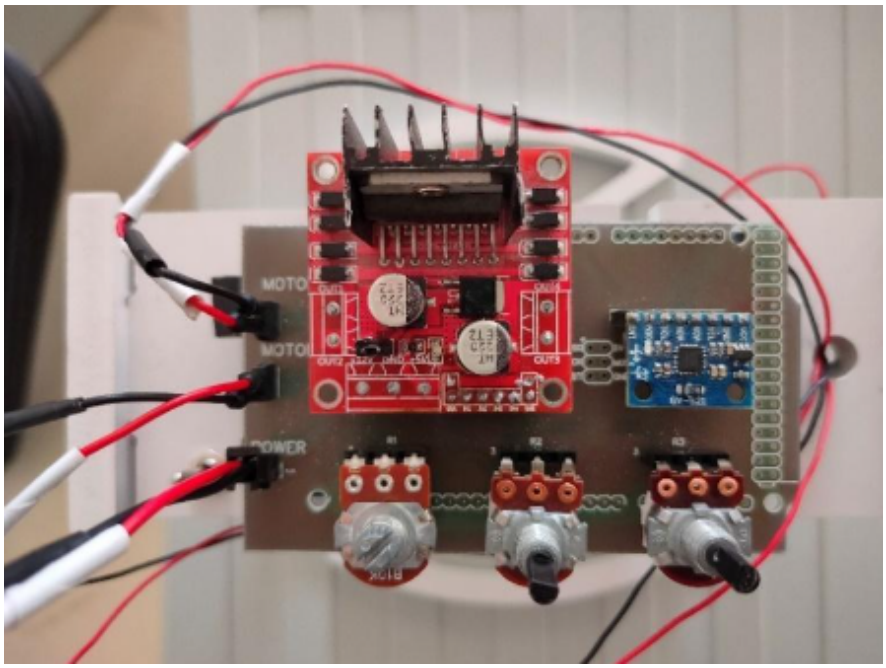

Figure 3.10: Illustration of the custom-designed Arduino Mega Shield PCB with various electronic components precisely and neatly mounted, demonstrating the effectiveness of the design in minimizing wire clutter.

Through this design approach, we successfully condensed a complex network of electronic components into a compact, organized, and manageable PCB, making our self-balancing robot functional and efficient.

3.5.2 mechanical part

We selected PVC foam board for the chassis [13] due to its lightweight nature and ease of machining. To create the parts, we turned to CNC machining, a method that ensures precision and consistency in the production of the chassis parts.

Once all parts were machined, we assembled them using an innovative screw-less approach. Instead, the parts interlock with each other, secured by friction. This method simplifies assembly, disassembly, and any necessary maintenance.

The final result of this mechanical design is a sturdy, lightweight, and efficient chassis that comfortably houses all the electrical components and the two DC motors. The robot's structure, combined with its intricate electrical design, results in a highly efficient self-balancing robot that can navigate its environment with ease.

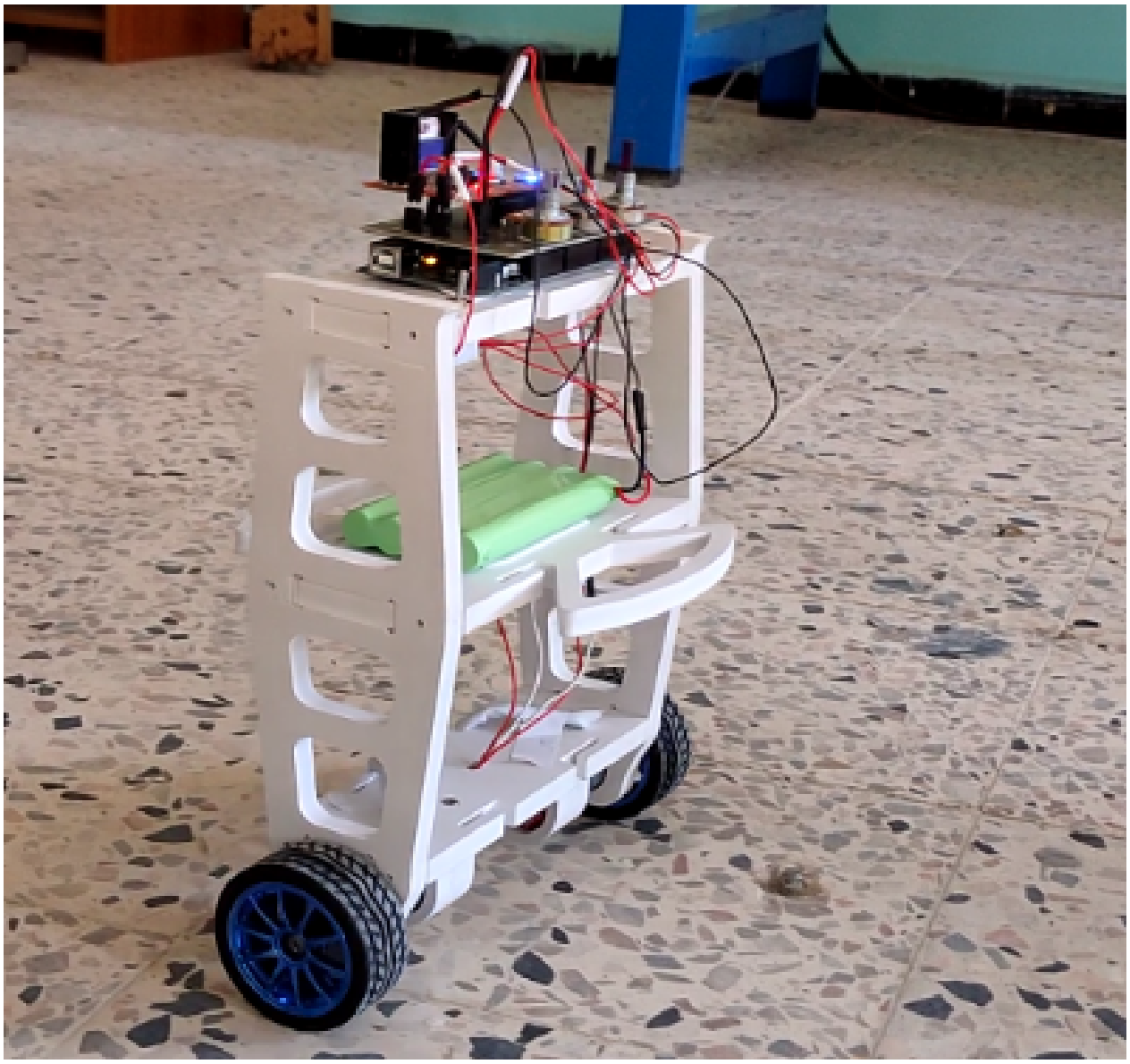

Figure 3.11: The experimental robot achieving a stable equilibrium position after meticulous PID tuning, demonstrating successful balance and control. [13]

3.6 An Updated Version

During this chapter, the intricacies of building a self-balancing robot as an application of the proposed design and construction workflow for mobile robots were explored. This subsection highlights a crucial upgrade to the robot, focusing on its modernization and enhanced usability. A significant aspect of this upgrade is the transition from an Arduino Uno to an Arduino Nano-based printed circuit board (PCB), now endowed with wireless capabilities. This improvement not only advances the design but also introduces a versatile wireless setup, enabling intuitive wireless control and marking a notable step in user-friendly interaction.

Additionally, the integrated Bluetooth module is used to wirelessly transmit the robot's Inertial Measurement Unit (IMU) data, facilitating a seamless and real-time performance analysis. This feature is especially critical as it eliminates the need for wired connections, which could negatively affect validity of the results.

Furthermore, the upgrade includes substantial structural refinements, particularly to the robot's chassis. The bumper plate has been redesigned with bridge-like patterns, enhancing its shock absorption capabilities, a key factor for a self-balancing robot prone to falls. Additionally, the batteries have been strategically repositioned higher in the robot, leading to a more centered and stable balance, which is essential for its optimal functionality. This subsection provides a detailed overview of these advancements, illustrating how each enhancement contributes to the robot's improved performance and reliability in practical scenarios.

3.6.1 PCB update

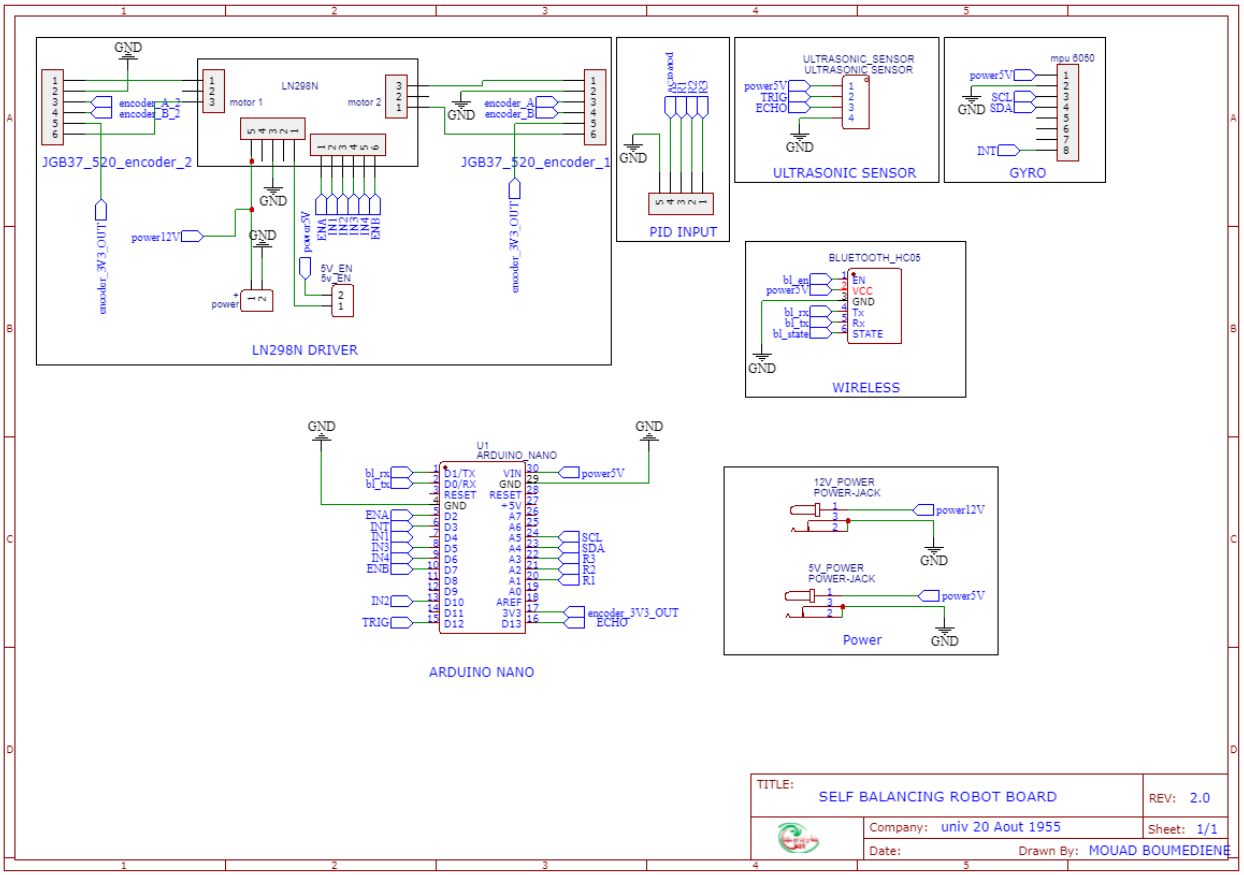

Figure 3.12: Schematic of the updated Self-Balancing Robot with Arduino nano

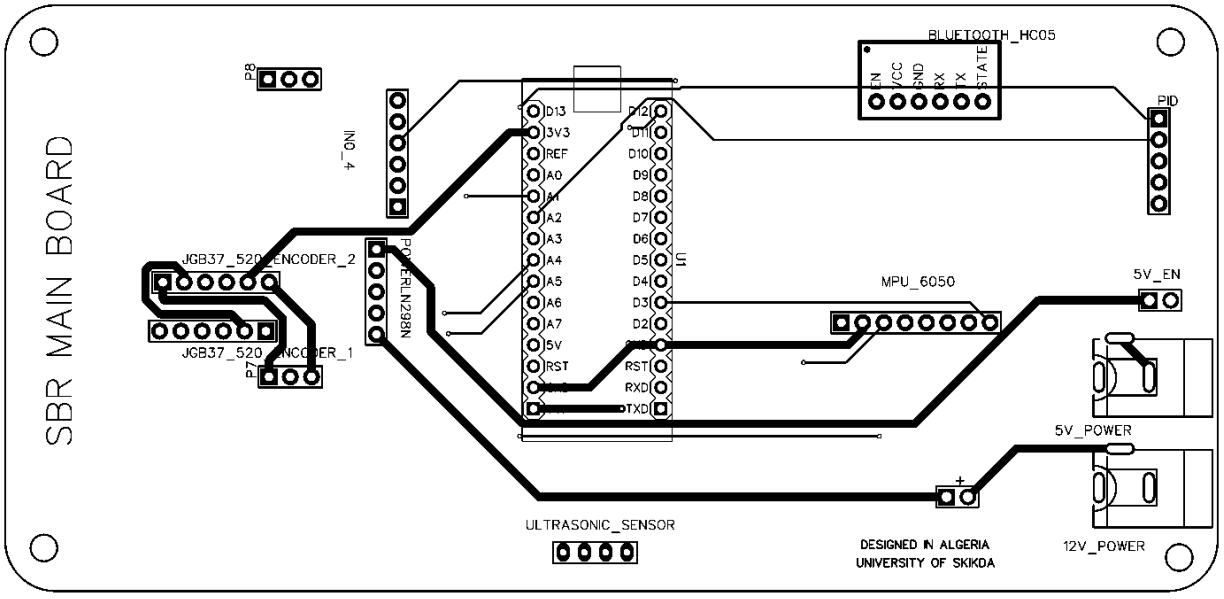

Figure 3.13: PCB Layout for an Arduino Nano-based Self-Balancing Robot, showcasing component placement and electrical trace routing for sensors

The transition to the Arduino Nano offers several technical advantages over the Uno. The Nano's significantly smaller size (18 mm x 45 mm) compared to the Uno (53 mm x 68 mm) is a critical factor in mobile robotics, where space efficiency and weight distribution are paramount. Despite its reduced form factor, the Nano maintains similar digital and analog pin configurations to the Uno. However, its compactness is achieved through a surface mount package, a space-saving alternative to the Uno's through-hole design.

Performance-wise, both the Nano and Uno are based on the ATmega328 microcontroller, ensuring that the shift to a smaller board does not compromise processing capabilities. A noteworthy advantage of the Nano is its lower power consumption, which is crucial for prolonging operational time in mobile robotic applications. Additionally, the Nano's smaller size offers enhanced integration flexibility within the PCB, allowing for a more complex and compact electronic layout, an essential aspect for the intricate design of the self-balancing robot.

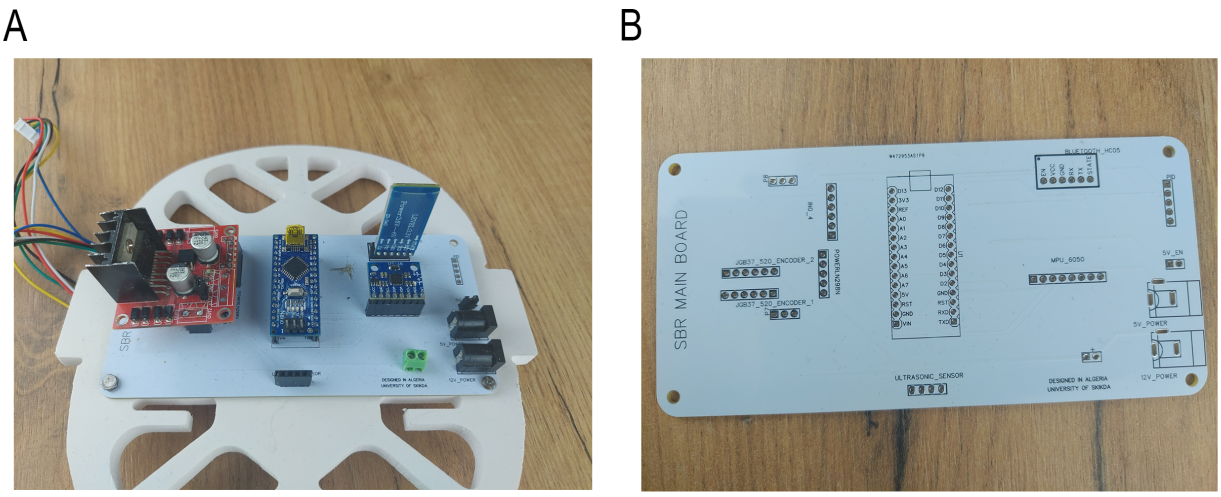

Figure 3.14: (A) Assembled self-balancing robot control board featuring Arduino Nano, motor driver, mpu6050 module and HC-05 bluetooth module mounted on the middle plate of chassis. (B) Corresponding unpopulated PCB highlighting the layout and circuit design for the control board.

The addition of the HC-05 Bluetooth module elevates the robot's communication capabilities. This versatile module supports both master and slave modes, enabling seamless connection with smartphones for remote control and data transmission. A key feature of the HC-05 is its capacity to wirelessly transmit the robot's IMU data, facilitating real-time monitoring and analysis without the encumbrance of wires, which could disrupt the robot's balance.

Moreover, the HC-05 module offers a high degree of customization through AT commands. This flexibility allows for the adjustment of settings such as baud rate and pairing information, ensuring precise and tailored control over the robot's wireless communication. The module's stable transmission range of approximately 10 meters is another advantage, providing reliable and uninterrupted connectivity, essential for effective control and data gathering in dynamic environments.

3.6.2 Chassis update

The recent updates to the chassis of our self-balancing robot represent a significant milestone in the evolving landscape of robotic engineering and design. In the field of robotics, continuous adaptation, and innovation are essential for keeping pace with technological advancements. The chassis, as the core structural element, is crucial for the robot's functionality and resilience. These updates are a crucial step in our ongoing journey of innovation, underscoring our commitment to incorporating new technologies and design approaches. Iterative design is fundamental in the world of robotics. It involves a continuous cycle of testing, feedback, and refinement. This process ensures that each iteration of our robot is an improvement over the last. By regularly updating and testing the chassis, we enhance the robot's adaptability, reliability, and efficiency. Each iteration is not just about meeting immediate functional requirements but also about gaining a deeper understanding of the robot's

operational dynamics, which is vital for developing more innovative and effective solutions in future designs.

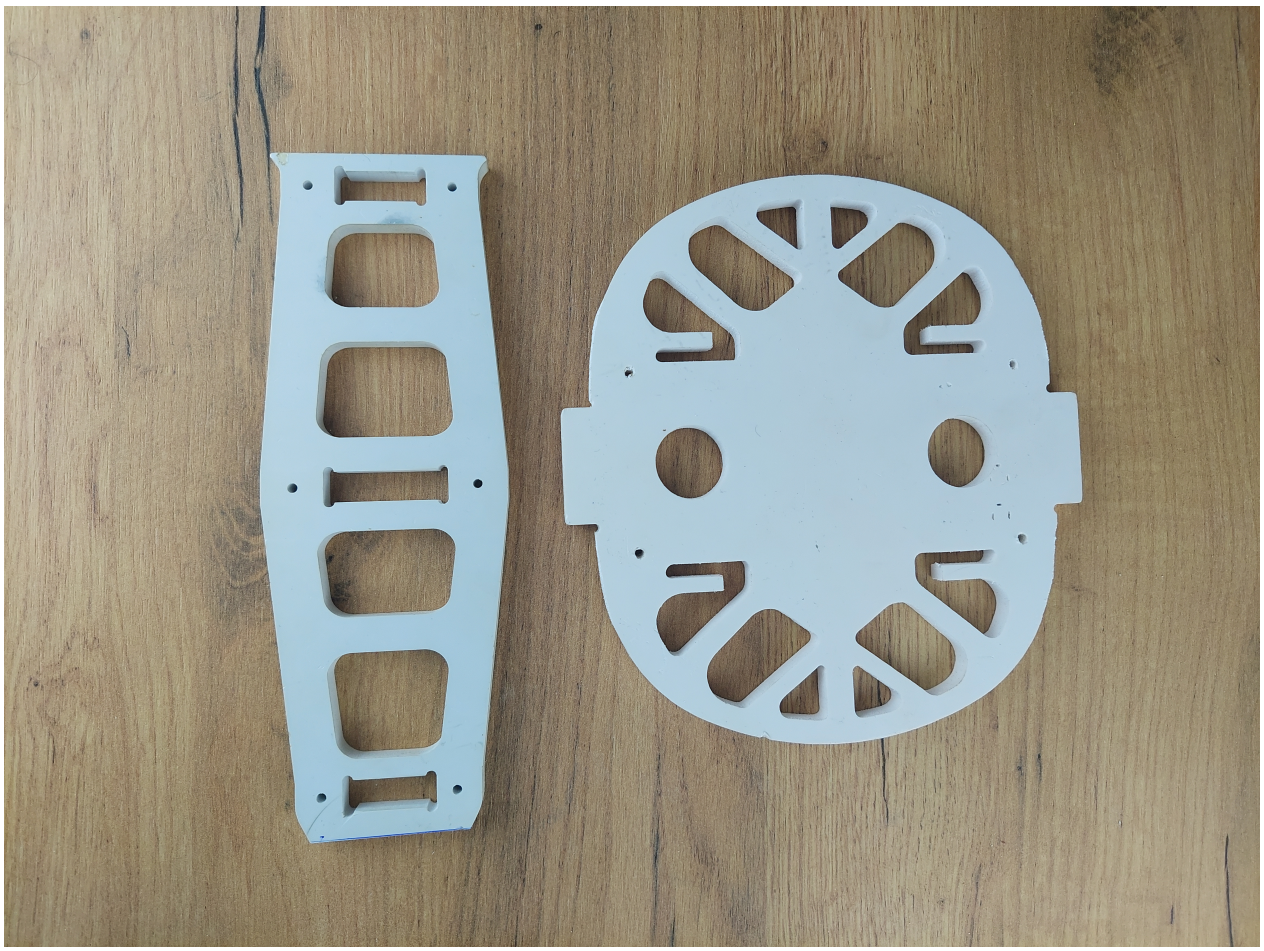

Figure 3.15: Updated parts of our self-balancing robot

The specific changes we have implemented (figure 3.15) - the removal of fixed motor holders, the introduction of adaptable motor mounting options, and the reinforcement of the middle plate - are pivotal. They reflect our dedication to improving the robot's performance and flexibility. These modifications allow for a wider range of motor compatibility and increase the robot's structural integrity, which is essential for its application in diverse environments. In summary, these updates mark a significant phase in our continuous process of development, ensuring that our robot remains at the forefront of self-balancing robotic technology.

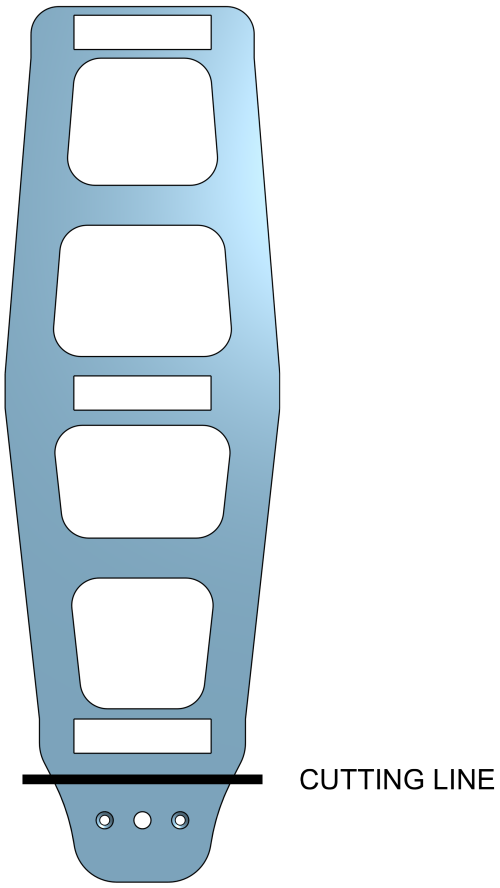

Figure 3.16: Illustration showing the cutting axis of the side plate, to remove the motor mounting holes

Initially, our robot's design included motor holders that were compatible with only a specific type of motor. This limited the range of motors that could be used with the robot. To address this limitation, we removed these dedicated motor holders. Instead, we cleverly modified the side plates of the robot by adding cuts 3.16. These cuts are designed to accommodate the motor holders that come directly from the motor manufacturers. This adjustment allows for a broader selection of motors to be used, as we can now order any motor along with its appropriate holder, ensuring a perfect fit and greater flexibility in motor choice.

Another significant change is the redesign of the robot's middle plate. Previously, the middle plate may have had limitations in terms of strength and shock absorption. To improve this, we've altered the design, potentially using different materials or structural adjustments, to enhance its durability and shock resistance. This change is crucial for a self-balancing robot, as it increases the robot's ability to withstand impacts and vibrations, a common challenge in mobile robotics. Notably, these design ideas weren't just theoretical but were

implemented and tested in a practical setting. They were part of a master's thesis that i supervised . This academic application provides a solid foundation for the effectiveness of these design changes. It indicates that the modifications have undergone rigorous testing and validation in a research environment, adding a layer of credibility and reliability to the design updates.

In conclusion, these updates to our self-balancing robot represent a thoughtful and strategic approach to design improvement. By increasing compatibility with various motor types and enhancing the robot's structural integrity, we've significantly broadened the potential applications and improved the overall performance of the robot.

3.6.3 Results

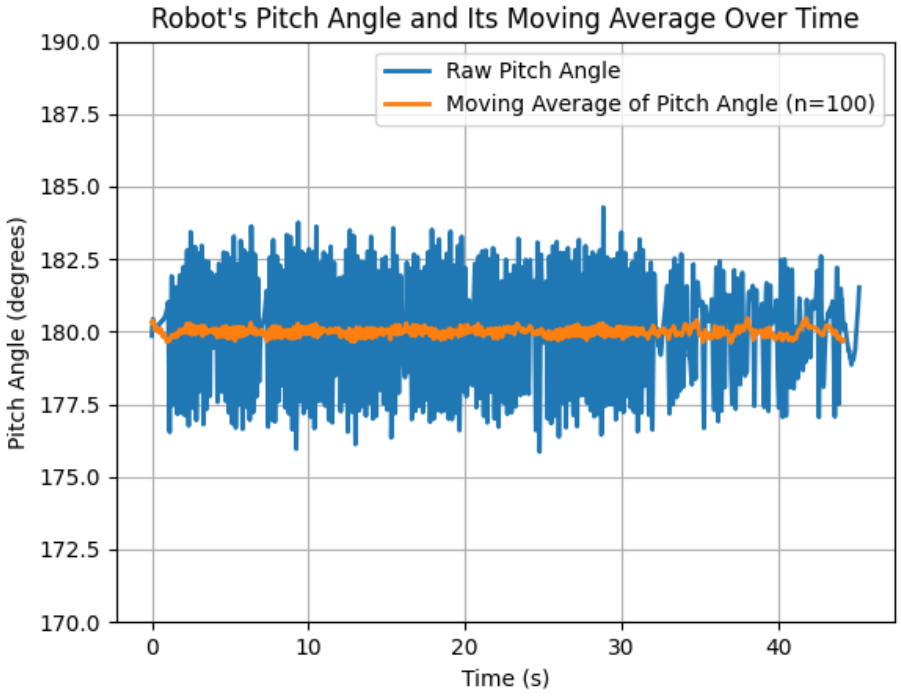

Figure 3.17: A plot showing the raw and averaged pitch angle of our updated version of self balancing robot while operating on a flat surface

In order to assess the performance of our updated self-balancing robot, we conducted an experiment aimed at evaluating its ability to maintain an upright orientation—a critical aspect indicative of its balancing efficacy. Central to this assessment was the robot's pitch angle, which had to remain consistently near a vertical baseline to ensure stability. To capture this metric, the robot was equipped with an inertial measurement unit (IMU) that provided real-time data on its orientation.

For a comprehensive analysis, it was imperative to gather high-fidelity data without impeding the robot's movement. To this end, we utilized a HC-05 Bluetooth module, which

facilitated the wireless transmission of the pitch angle data directly to a PC. This setup enabled us to monitor its performance in real-time, a crucial feedback for immediate analysis.

Once the experiment commenced, The motors worked in concert with the control algorithms to make precise adjustments in response to any detected tilt from the vertical axis. The IMU relayed the pitch angle data at high-frequency intervals, capturing the subtleties of the robot's adjustments. This stream of data, charting the course of the robot's attempts to balance itself, was continuously sent to the PC over the Bluetooth connection 3.17.

The experiment spanned over 40 seconds, a period strategically chosen to observe the robot's performance and its ability to sustain balance over time. The raw data, was subjected to further analysis by applying a moving average filter, which revealed the underlying stability trends by smoothing out the high-frequency fluctuations.

Inspection of the plotted data revealed the robot's active and dynamic self-correction mechanism in action. Despite the noise, the robot's pitch angle remained remarkably consistent with the expected equilibrium position, as indicated by the moving average line. This was a testament to the updated system's ability to adapt and maintain balance. However, occasional spikes in the pitch angle suggested the need for further tuning of its PID controller gains.

Overall, the wireless transmission of data was instrumental in capturing an accurate depiction of the robot's balancing performance, free from any tethering that could influence its behavior which was not the case for the first version of this robot. The analysis not only confirmed the effectiveness of the updated self-balancing mechanisms but also shed light on areas where further improvements could be made.

3.7 Conclusion

this chapter on mobile robotics has given a thorough understanding of the fascinating fusion of mechanics, electronics, and sophisticated control algorithms. I have explored the design, simulation, and construction domains, and we have discovered the complex problems and creative solutions that define the creation of autonomous robotic systems.

This chapter has highlighted the interdisciplinary nature of mobile robotics research, ranging from the early design phases, where the conceptualization of robotic platforms and the careful selection of sensors and actuators take center stage, to the remarkable power of simulation techniques, which enable us to scrutinize and optimize every aspect of robot behavior in virtual environments.

The construction stage is where the journey reaches its pinnacle because this is where theory is put into practice. Here, the complex integration of electronic circuitry, the calibration of sensors, and the assembly of mechanical parts result in the physical realization of a robot. This section has been filled with problems that have been solved creatively, demonstrating the resourcefulness and tenacity that characterize the field of mobile robotics.

I were led by a real-world example—a self-balancing robot—as I made my way through these complex facets of mobile robotics. This practical example demonstrated how theoretical understanding and actual application can coexist in harmony. It demonstrated the transformation of intangible ideas into concrete robotic systems, illustrative of the transformative potential of mobile robotics research.

In summary, this chapter explored the intricate details of design, simulation, and construction to shed light on the multifaceted world of mobile robotics. We wanted to encourage researchers, engineers, and students to set out on their own odyssey of mobile robotics exploration by fusing theoretical knowledge with real-world examples. With this all-encompassing strategy, we hope to close the gap between theoretical understanding and practical applications, fostering innovation and advancing the fascinating field of mobile robotics. For those who wish to venture further into the uncharted regions of robotic autonomy and its seemingly limitless potential, this chapter serves as a starting point.

Chapter 4

Path Planning for AMRs

Contents

4.1 Introduction

Path planning is a crucial module in most autonomous navigation systems. Once a sufficiently accurate map of the environment has been built, an obstacle-free path to a goal position within that map can be calculated. Due to the increasing integration of autonomous mobile robots over the past two decades, path planning has received a significant amount of attention not just from robotics developers but also from many other disciplines such as computational structural biology [101] [102], crowd simulation [103, 104], and video game development [105]. Discretizing the continuous state space using a deterministic grid with a predetermined resolution is one of the most popular approaches for solving the path-planning problem. This grid representation of the environment is then searched using graph search algorithms such as A* [106] and Dijkstra's [103], to obtain a resolution-optimal path between the start and goal positions. However, the biggest Inconvenience in these approaches is that building and maintaining the grid becomes highly expensive especially when a high resolution is required. Another method for solving this problem is using a stochastic approach

which totally avoids building a square grid representation of the environment, and instead follows an incremental procedure upon which, random obstacle-free samples from the environment are generated and then used to grow a search tree in the free space. This drastically reduces the planning cost in terms of computational resources, as well as memory consumption. However, the biggest drawback of random-sampling-based algorithms such as RRT[6] and RRT*[7] face is the fact that the resulting path is sub-optimal. Which negatively affects the AMR performance.

In This chapter, we will propose a method for exploiting the fast search advantage the probabilistic sampling-based methods have, in reducing the cost of optimal deterministic grid-based methods; which will result in better performance regarding computational efficiency and memory usage.

4.2 Problem Definition

Let's consider an open set X contained or equal to Rn that defines the state-space of the path planning problem where $n \geq 2$. Also, let $X_{obs} \subset X$ be a set of states that are in collision with obstacles. And let the set X_{free} be its complement where it contains only obstacle-free states.
Figure that has a path planning environment containing a start and goal and obstacles with X,x_{goal}, x_{start} defined.also define Euclidean distance $d_2(x_1, x_2)$ X_{goal}: the desired goal region
X_{start}: the initial state
$d_2(x_1, x_2)$: the euclidean distance between states $x_1 and x_2$ belong to X^2.
The general path planning problem, is to calculate a path $\pi[0, 1] \subset X_{free}$ and obstacle-free path where $\pi(0) = x_{start}$ and $\pi(1) \in X_{goal}$ if such a path exists, if not it must return failure.

4.3 Discrete Path Planning

In the case of discrete path planning, the state space X is in most cases a countable, finite, set. It also has to be large enough to include all the information relevant to the path-planning problem. It should also be noted that the inclusion of irrelevant information will only render the problem intractable [8]. In this section, we represent the environment by a discrete state space X that lies on an occupancy grid which is a fixed-resolution square grid, where obstacle states are represented by occupied cells and free-space states are represented by free cells. The occupancy grid representation is one of the most commonly used frameworks by Discrete path planning algorithms, and it will allow us to clearly describe the problem at hand, along with its proposed solution. An occupancy grid of size $(H \times W)$ is created by discretizing the continuous state space using a fixed resolution r representing the dimensions of its cells. The discretization resolution should be small enough to capture the details of the environment; however, it is appropriate to point out that the size of the memory space required for storing this structure and the computation resources for its manipulation does have a negative relation with the resolution. This means that keeping a reasonably high enough resolution will ensure that solving this path-planning problem will still be affordable in terms of computer resources. Once the size and resolution of the grid is determined, the next step is to determine the occupancy of each cell. This is typically done by using a sensor

such as a laser rangefinder or camera to measure the distance to obstacles in the environment. The distance measurements are then used to assign a value to each cell indicating whether it is occupied or not.

Finally, the occupancy grid can be used to plan a path for a robot or other autonomous systems. Graph searching algorithms such as A* or Dijkstra's can be used to find the shortest path between any two nodes in the implicit graph within the occupancy grid representation of two points in the environment, taking into account the occupancy of each node.

4.3.1 A*

A* is a famous grid-based discrete path planning algorithm used to find the shortest path between two points on a graph. It is a type of best-first search algorithm, meaning that it searches for the best path from the start to the goal by expanding the most promising nodes first.

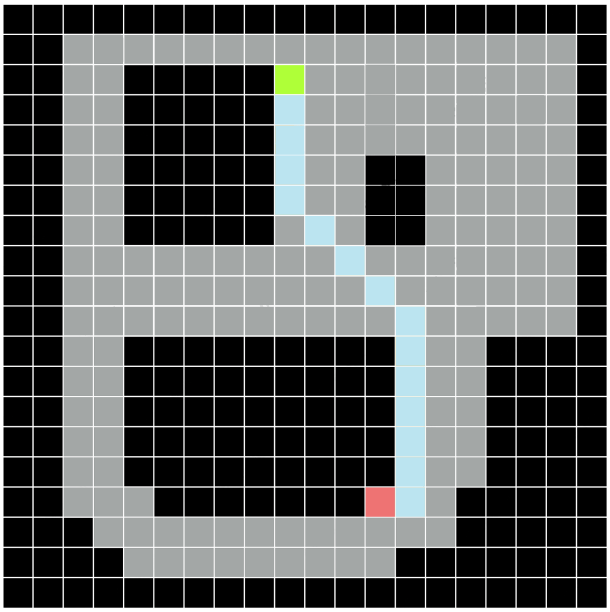

Figure 4.1: illustration of A* algorithm planning a path from a start to a goal position in a binary map.

The A* algorithm works by combining the cost of a path with the estimated cost of the remaining path to the goal. This is done by assigning a cost to each node in the graph, which is the sum of the cost of the path from the start to the node plus the estimated cost of the remaining path to the goal. The algorithm then expands the most promising node, which is the node with the lowest cost. This process is repeated until the goal is reached. A* is an example of a heuristic search algorithm, meaning that it uses an estimate of the remaining cost to the goal to guide its search. This estimate is called the heuristic, and it is usually based on the distance between the current node and the goal. The heuristic is used to determine which node to expand next, and it helps the algorithm to avoid expanding nodes that are unlikely to lead to the goal.

4.3.2 D*

The D* algorithm is an incremental variant of A*. It works by maintaining a priority queue of nodes that need to be evaluated. As the environment changes, the algorithm updates the priority queue and re-evaluates the nodes in order to find the shortest path. This allows the algorithm to quickly adapt to changes in the state space and find the best path in a dynamic environment.

The main difference between the two algorithms is that A* is a static algorithm, meaning that it can only be used in a static environment. On the other hand, D* can be used to update the path in response to changes in the environment. This makes D* more suitable for environments, where the state-space is constantly changing.

4.3.3 Life-long planning A*

Lifelong Planning A* (LPA*), is an incremental variation of A* that integrates concepts from the literature on algorithms and artificial intelligence. LPA* repeatedly determines the shortest pathways between a given start vertex and a particular goal vertex. When the edge costs of a graph change or new vertices are added or removed. Figure

At first LPA* performs a similar search to that of A*, with the sole exception that LPA* favors vertices with smaller g-values. For the following searches, the calculation time drops remarkably due to the fact that LPA*reuses the parts of the prior search tree that are comparable to the present one. Making it very beneficial in finding paths in dynamic environments, as the graph can be updated as the environment changes.

4.3.4 Genetic Algorithms

Genetic algorithms are a type of optimization algorithm that can be used for path planning. They are based on the principles of natural selection and evolution and are used to find the best solution to a given problem. They work by creating a population of chromosomes or potential solutions and then using a process of selection and mutation to find the best one. The selection process involves evaluating each chromosome and selecting the one that is most likely to be successful. The mutation process involves randomly changing some of the parameters or genes of the solution to see if it can be improved.

The process is repeated until the best solution is found. Genetic algorithms are particularly useful for path planning because they can take into account multiple factors and find the most efficient route. They are also able to adapt to changing conditions, such as obstacles or terrain, which makes them ideal for dynamic environments.

4.4 Sampling-based path planning

The basic concept behind most Sampling-based path planning algorithms is to avoid the creation of an expensive occupancy grid before beginning the search process. Instead, they randomly sample points from the environment to build a graph where its first vertex is the start point from where the planning begins. Next, they use a search algorithm from the previous section to find the shortest path between the two points. The search algorithm can

be either a heuristic search that uses a heuristic function to estimate the cost of a path or an exact search that uses an exact cost function. Sampling-based planning relies on drawing Random samples from a uniform probability distribution over the robot's planning domain. This reduces immensely the computation time and memory space needed for the creation, maintenance, and manipulation of a grid. This inspired the development of many algorithms that uses this concept RRT: [73], RRT* [107], PRM [72], BIT* [79]. However, the quality of said solutions is tightly tied to the denseness of the sampling sequence. This means that the cost of the calculated path reduces when the number of uniformly random samples increases. An algorithm is said to be complete if within its design constraints, it finds a solution or reports that no solution was found. Sampling-based path planning algorithms, are often probabilistically complete, meaning if a solution does exist the probability of finding this solution asymptotically approaches infinity as the number of randomly generated samples increases.

4.4.1 RRT

The Rapidly-exploring Random Tree path planning algorithm is a popular algorithm used in robotics and autonomous navigation. It is a probabilistic algorithm that incrementally samples random points in the environment and connects them to form a growing tree-like structure. Which is used to find a feasible path between two configurations.

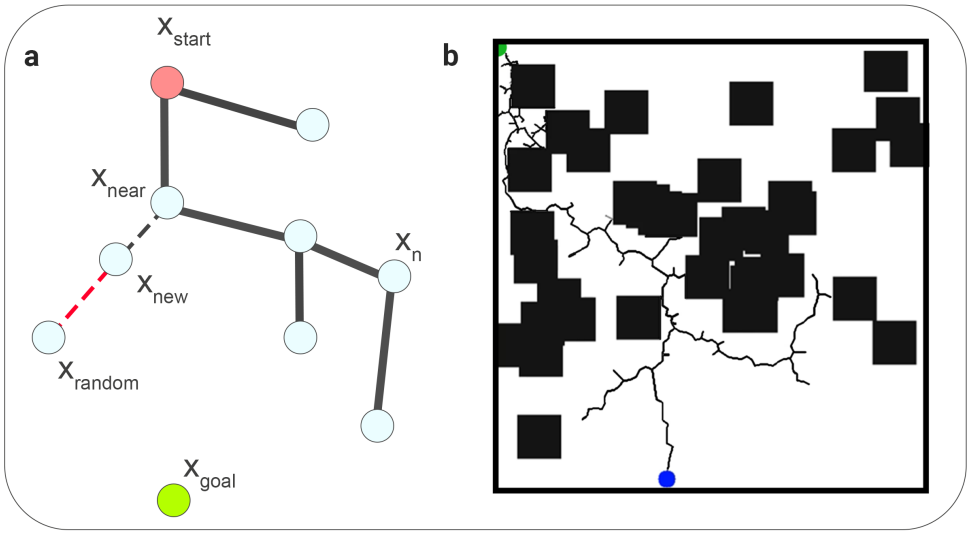

Figure 4.2: (A) illustration of RRT planning Process (B) Simulation results of RRT algorithm.

At the beginning, the RRT graph is composed of only one Vertex Qstart, in each iteration a random point Qrand is sampled from a uniform probability distribution over the configuration space. If Qrand is located in the free space, then RRT will use it to expand its graph. This is done by finding Qnear which is the closest point in the tree to that sample, if the direct line to Qrand does not cross any obstacle then RRT will take a step towards it using a fixed predetermined step size and then add that point Qnew to the graph if it's obstacle

free. This process repeats until either Qnew is close enough to goal the region or some other termination condition is met. The RRT algorithm is an efficient and effective way to find a feasible non-optimal path between two points in a given environment. It is also relatively simple to implement and can be used in a variety of applications. Furthermore it is a robust algorithm that can handle dynamic constraints.

4.4.2 PRM

In contrast to single query methods mentioned in this section, where only a single start-goal pair is handled at a time, the probabilistic road map (prm) generates uniformly random samples and builds a graph over the configuration space. After the graph is constructed the road map can be used to search for feasible paths given multiple start-goal pairs. In this case, it seems to be reasonable to allocate more resources for building the road map in this preprocessing phase, so future queries can be answered efficiently [lavall book]. PRM works by splitting the planning process into two phases, the preprocessing phase and search phase.

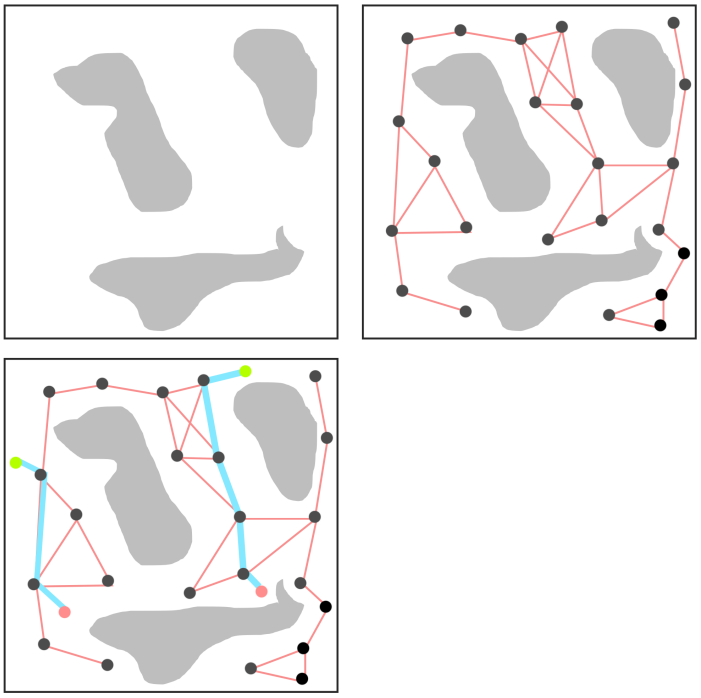

Figure 4.3: illustration of PRM algorithm building a Road Map and planning multiple queries within that map.

During the preprocessing phase, random samples are drawn from a uniform probability distribution. In every iteration the new sample Si must be checked for collision, if no collision is detected this sample is added to the graph by attempting to connecting the nearby vertices. The graph vertices to which Si attempts to connect with can be chosen in several ways. For example the k nearest neighbors, all point in a radius, or to connect to all visible vertices in the graph G.

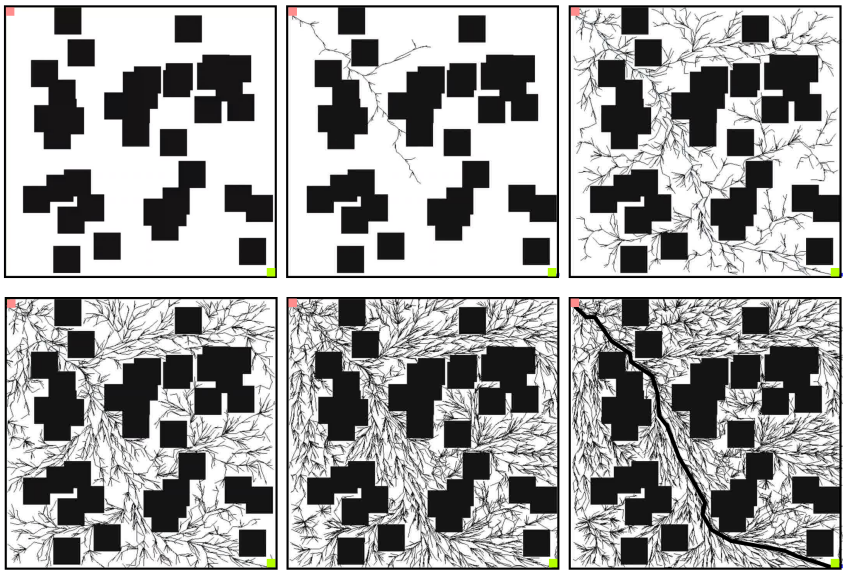

Figure 4.4: simulation results of RRT* algorithm.

In The query phase, multiple queries have to be answered; where each query is in the form of a qi and qg pair that represents the initial and goal configurations respectively. At first, each pair will have to be connected to the graph in a similar way to the preprocessing phase. If the connection step went successfully, PRM performs a search for the least cost path that connects qi to qg through the graph G. PRM offers a neat solution for multi-query path planning, however, when it comes to the notion of completeness, this approach have an unfortunate drawback. If PRM fails to find a solution to query it is not decisively known whether no solution exists or if further samples need to be introduced to the graph in order to enhance its resolution.

4.4.3 RRT*

The RRT* path planning algorithm is another efficient sampling-based planner. It is an improvement over the RRT algorithm, as it is able to find an asymptotically optimal solution to the path planning problem. This algorithm which was proposed in 2011 by karaman and frazzoli [7], uses new samples to rewire the paths extended by the search tree, thus reducing the cost of the produced path until an asymptotically optimal solution is returned.

In contrast to RRT, The cost of paths generated by RRT* are proven to eventually converge to the global optimum [107], meaning that with enough uniformly random samples, an optimal path between two input configurations can finally be obtained as the number of iteration approaches infinity. It should also be noted that RRT* is by inheritance, a probabilistically complete algorithm which means that if a solution exists RRT* will find it with a high probability.

the RRT* algorithm as shown in the pseudo-code works in a similar way to the RRT search by growing a tree in the obstacle-free space until an initial feasible route is found; thus using random samples to incrementally grow the tree until it reaches the goal configuration. But in

Algorithm 1 $RRT^*(X, x_{start}, x_{goal})$

$V \leftarrow \{x_{start}\}$,$E \leftarrow \emptyset$
$T \leftarrow (V, E)$
$c_{init} \leftarrow \infty$
for $k \in \{1, ..., K\}$ **do**
  $x_{rand} \leftarrow Sample(X)$
  $X_{nearest} \leftarrow Nearest(T, x_{rand})$
  $x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$
  if $ObstacleFree(X_{nearest}, X_{new})$ **then**
    $V \leftarrow V \cup \{x_{new}\}$
    $X_{near} \leftarrow Near(T, x_{new}, r_{RRT*})$
    $x_{min} \leftarrow x_{nearest}$
    $c_{min} \leftarrow cost(x_{min}) + d_2(x_{nearest}, x_{new})$
    for all $x_{near} \in X_{near}$ **do**
      $c_{new} \leftarrow cost(x_{near}) + d_2(x_{near}, x_{new})$
      if $c_{new} < c_{min}$ **then**
        if $CollisionFree(x_{near}, x_{new})$ **then**
          $x_{min} \leftarrow x_{near}$
          $c_{min} \leftarrow c_{new}$
        end if
      end if
    end for
    $E \leftarrow E \cup \{(x_{min}, x_{new})\}$
    for all $x_{near} \in X_{near}$ **do**
      $c_{near} \leftarrow cost(x_{near})$
      $c_{new} \leftarrow cost(x_{new}) + d_2(x_{new}, x_{near})$
      if $c_{new} < c_{near}$ **then**
        if $CollisionFree(x_{new}, x_{near})$ **then**
          $x_{parent} \leftarrow Parent(x_{near})$
          $E \leftarrow E\ \{(x_{parent}, x_{near})\}$
          $E \leftarrow E \cup \{(x_{new}, x_{near})\}$
        end if
      end if
    end for
    if $IsGoalState(x_{new})$ **then**
      return $C_{init} \leftarrow cost(x_{new})$
    end if
  end if
end for

return C_{init}

contrast to RRT, the new samples will trigger a rewiring process. The rewiring process used in RRT*, attempts to reduce the cost of the path connecting the start vertex to each vertex in the X_{near} set containing X_{new} neighboring vertices. If the newly added vertex is found to provide a better path to a neighboring vertex then the parent of that vertex is replaced by X_{new} and the old edge is deleted.

4.5 FDA*

In this subsection, We provide a Focused Discretization (FD) technique for reducing the cost of single-query grid-based path planning approaches by narrowing the search to a smaller subset of the environment which is guaranteed to contain the optimal path. We also present an implementation of this method named FDA^* (Focused Discretization A^*) usingA^* as an optimal graph search algorithm [108].

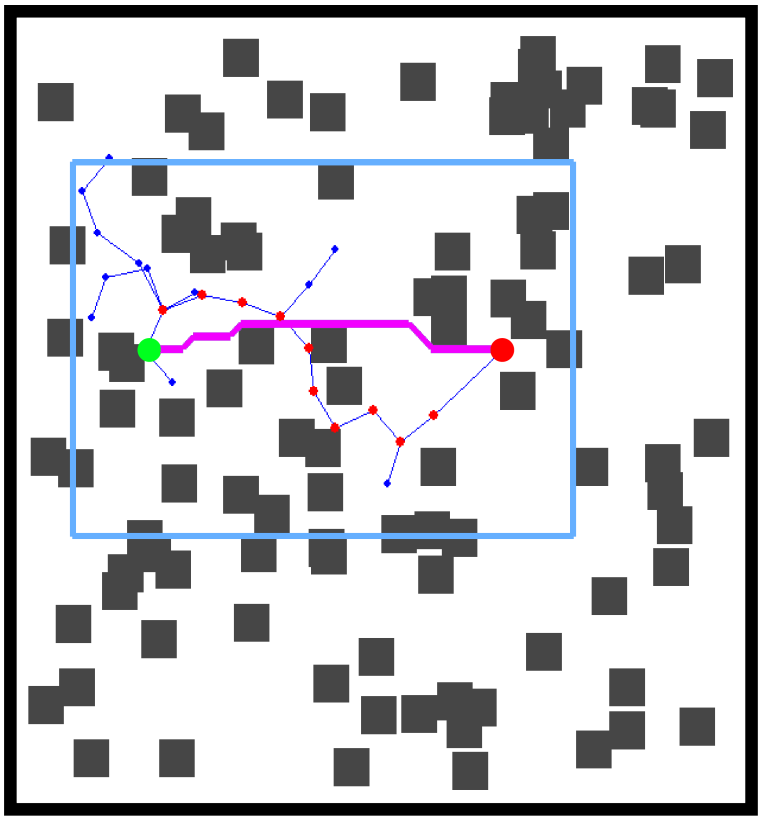

Figure 4.5: Example of FDA^* in a 2D Euclidean space with square obstacles, the new state-space represented by the blue borders is discretized and searched by the A^* algorithm and finally the resolution optimal path is generated between the start (green) and goal (red) states.

Focusing the path planning process to include only a subset of the full state space has been used before in stochastic algorithms in the form of rejection sampling. An initial path to the goal is calculated; after which a prolate hyperelipsoid is overlayed over the state space

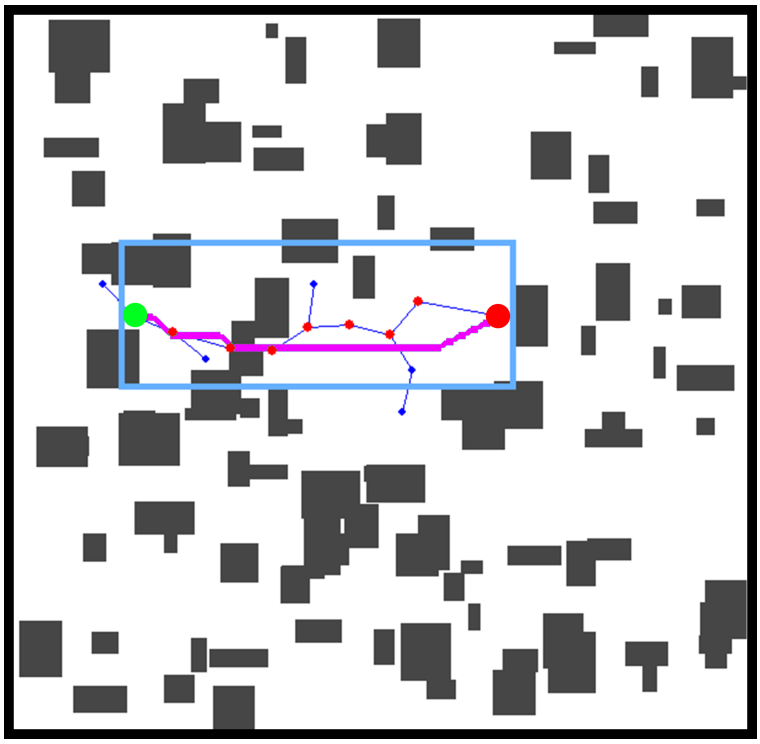

Figure 4.6: Example of FDA^* in a 2D Euclidean space with random size obstacles, the new state-space represented by the blue borders is discretized and searched by the A^* algorithm and finally the resolution optimal path is generated between the start (green) and goal (red) states.

and only states that fall within the boundaries of this hyper-ellipsoid are accepted to be used in further growing the search tree.

The Focused Discretization concept exploits this idea and adapts it to the realm of grid based path planning. It works by first calculating a feasible initial path using a fast search algorithm such as RRT*, Then calculates a subset that fulfills the following inequality: $C_x < C_{init}$. Where C_x is the cost of the unique optimal path from x_{start} to x_{goal} bounded to pass through the state x belong to X(state space) and C_{init} is the cost of the initial path.This subset is defined by a prolate-hyper-ellipsoid with x_{start} and x_{goal} as its focal points and D_1 and D_2 as its transverse and conjugate diameters. Realistically, discretizing the state space into a square grid requires us to use a geometrically compatible structure. A good candidate for this task can be the hyper-rectangle X_r that tightly bound X_s, as the new boundaries of our state space. By definition, X_r is less or equal than X which means that it will allow for avoiding the high costs correlated with building a grid over the full state space and also the costs of storing and maintaining this expensive structure as going to be shown later.

The pseudo-code shown in algorithm 2, presents an implementation of Focused Discretisation using A^* path planning algorithm named FDA*. The only difference between the A^* and FDA^* implementations is the introduction of lines 2 and 3. where RRT* is used for the few first iterations in order to calculate an initial path of sub-optimal cost C_{init} from (x_{start} to x_{goal}). The cost C_{init}, together with the x_{start} x_{goal} pair, is then sent to the function new bounds, which computes and provides the state space's new boundaries in the form of the

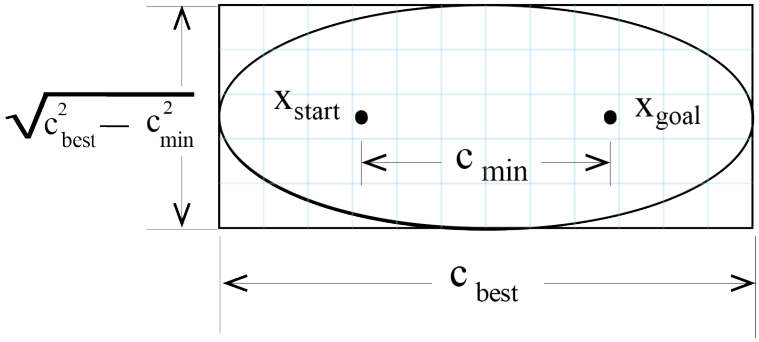

Figure 4.7: This figure shows the ellipsoidal subset X_s in the case of a 2D path planning problem calculated departing from the initial path between x_{start} and x_{goal} of cost C_{init} inside X_r which is the rectangular subset that tightly bounds X_s where its width and height are equal to X_s's diameters it is used to mark the new boundaries of the state-space to be discretized

Algorithm 2 $FDA^*(X, C_{init}, x_{start}, x_{goal})$

1: $L_r \leftarrow$ initialize to the original domain
2: $\boldsymbol{C_{init} \leftarrow RRT^*(X, x_{start}, x_{goal})}$
3: $\boldsymbol{L_r \leftarrow new\_bounds(X, C_{init}, x_{start}, x_{goal})}$
4: $G_r \leftarrow discretiz(X, L_r)$
5: $A^*(G_r)$

vertices of the rectangle X_r.

In order to demonstrate our method's performance regarding computational efficiency and memory usage, we designed two experiments, each conducted by performing a vast number of Monte Carlo simulations in environments populated with randomly generated (30x30) cell obstacles. Furthermore, the density of obstacles will be changed to create environments of varying complexity (examples are shown in figure 3). It should be noted that the density values employed in our experiments do not exceed 32% since most real-life applications operate within these limits.

For this initial test, we used a 1200x1200 pixel map to compare the performance of our

Table 4.1: comparison of the performance of FDA* with the classic A* algorithm regarding execution time in different scale environments.

env	dimensions (pixels)	start-goal distance (pixels)	obstacle density from which FDA* becomes out-performed
1	512 x 512	300	0 %
2	700 x 700	300	16 %
3	900 x 900	300	16.50 %
4	1200 x 1200	300	19.37 %
5	1500 x 1500	300	21 %

method (FDA*) and the traditional A* algorithm, while maintaining a constant distance between the start and target positions. The amount of memory used by each algorithm was noted each time a simulation was run, and the final relsults were averaged and finally shown in figure 4. This experiment demonstrates a significant reduction in memory usage for the FDA* algorithm when compared to A*, which is also displayed in Figure 5, where their ratio is illustrated, showing an average reduction in memory usage of 92% in relatively simple environments, like environment (a) in Figure 3. Moreover, while this percentage drops off in areas with more dense populations and small passageways, environments with greater complexity still achieve an average reduction of 83We also conducted an experimental comparison between the two algorithms regarding their computational efficiency. we used an environment with 600x1200 pixels and ran both the FDA* and the traditional A* algorithms through the same number of simulations as the first experiment. We then averaged the execution times for both of them. The results of this experiment are shown in Figure 6, where it is evident that our algorithm performs better in situations with an obstacle density of less than 15%, which is thought to be the region in which the majority of real-world applications operate. Beyond that, the conventional A* path planning outperforms our technique regarding execution-time. Several further trials were conducted using merely the map's size variation, similar to the second experiment. Table 4.1 shows the findings, and it is easy to see that as the size of the problem increases, the density at which our approach is outperformed, increases. Giving our algorithm the advantage in relatively large-scale environments.

We have shown that focusing the discretization process to include only the states that are actually capable of contributing to finding the optimal path, can immensely reduce memory requirements, especially in large-scale environments. Furthermore, this Focused Discretization approach can reduce the overall run time for grid-based path-planning algorithms. Especially through low and medium obstacle-density environments.

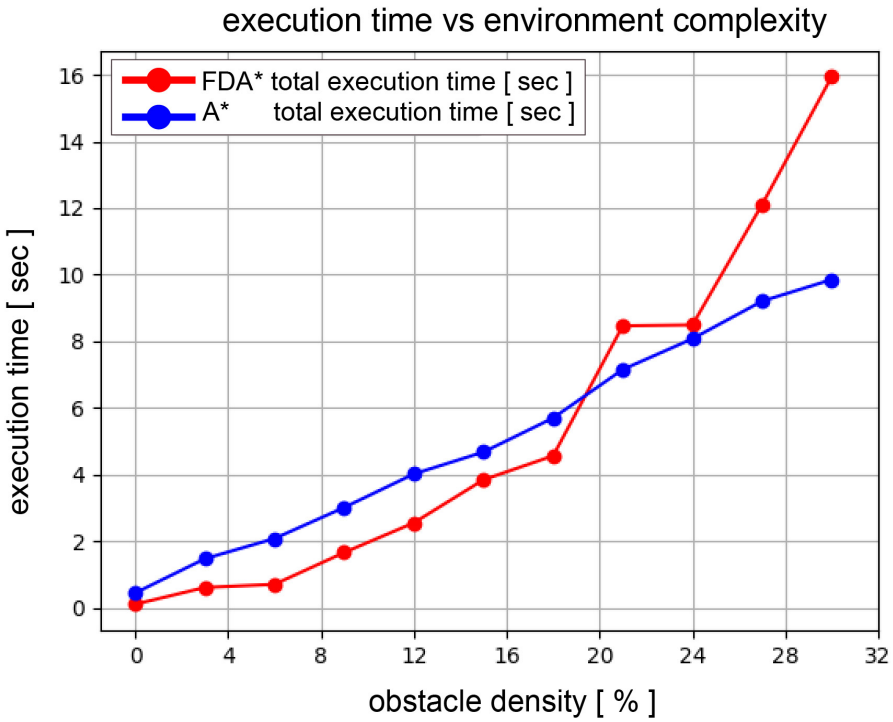

Figure 4.8: Average execution time needed by FDA* and A* for finding the optimal path in 2-D environments with varying obstacle densities and a constant distance between the start and goal poses.

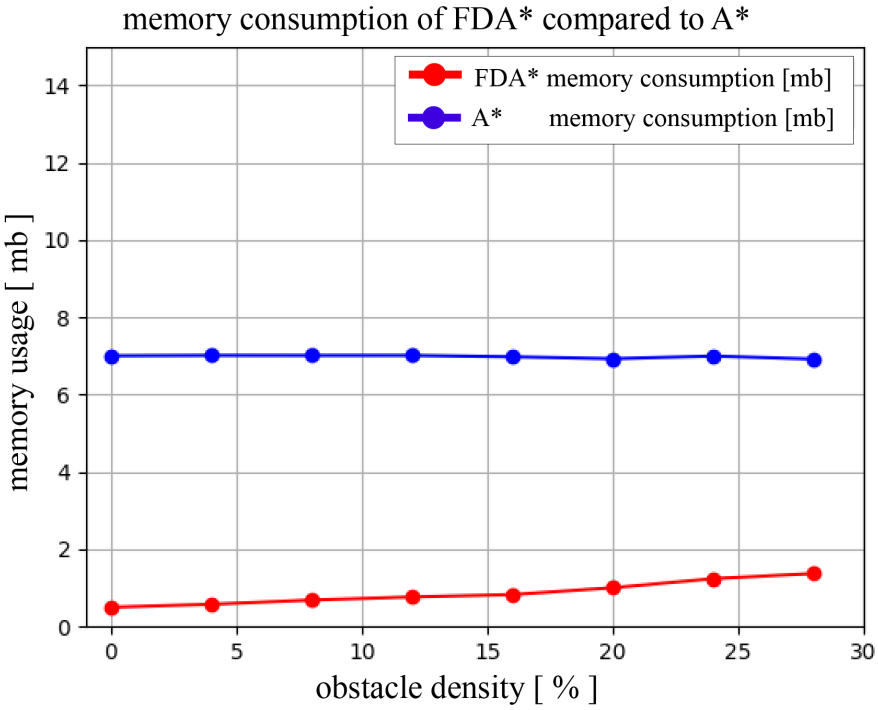

Figure 4.9: Comparison between the average memory space consumed by FDA* and A* algorithms

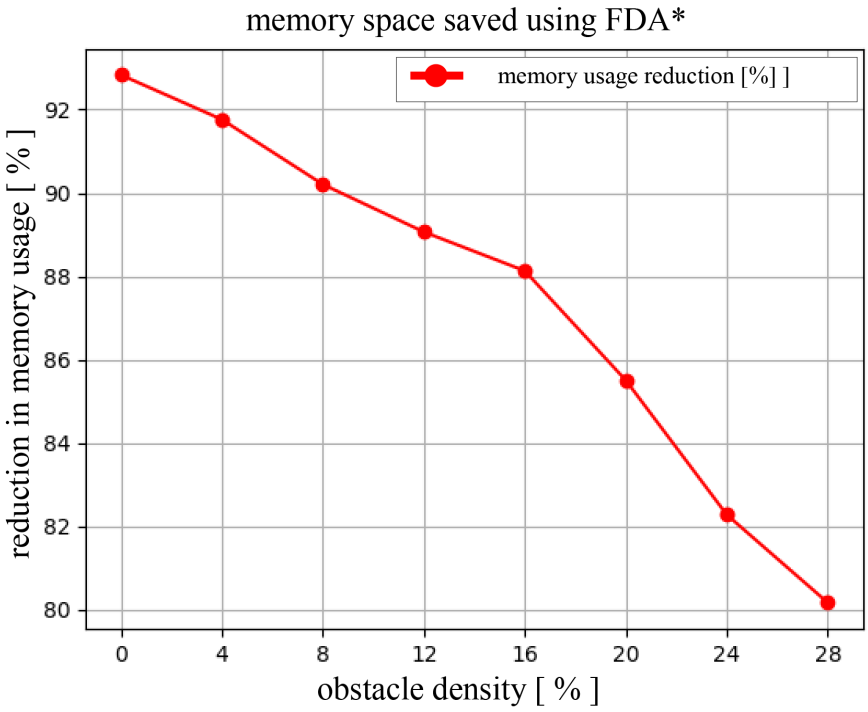

Figure 4.10: The average percentage of memory space saved using FDA* instead of the classic A* algorithm

4.6 Conclusion

This chapter has presented a detailed definition of the path-planning problem for autonomous mobile robots. The general problem has been divided into two separate instances which are grid-based path planning and sampling-based path planning. For each instance, several related algorithms were presented in detail providing illustrations and pseudo-code. At the end of this chapter, a novel approach for solving the grid-based path planning problem using stochastic-based algorithms was presented. Simulated experiments showed that Focusing the search process to include only a part of the state space which is guaranteed to contain the optimal solution, reduces not only the required memory space but also the needed computational resources.

Chapter 5

General Conclusions

The trajectory of this doctoral journey has been profoundly influenced by a fervent aspiration to drive innovation within the expansive realm of autonomous mobile robotics (AMR). Throughout the course of this dissertation, the focus has been steadfastly directed toward enhancing the foundational elements of AMR development. Chief among these elements are Path Planning and the actual workflow employed in crafting these remarkable machines. This pursuit, however, has been characterized by a delicate equilibrium between the incorporation of established practices and the infusion of cutting-edge innovations.

The overarching objective has remained clear and resolute: to curtail the computational processing demands associated with autonomous navigation. This overarching goal bears significance as it has the potential to drastically broaden the horizons of AMRs, rendering them adaptable to a multitude of contexts and sectors. This aspiration underscores the intrinsic value of making AMRs more accessible and versatile tools, poised to cater to the evolving needs of industries, from manufacturing and logistics to healthcare and even everyday life.

As we embark on the final chapter of this journey, it is an opportune moment to pause and reflect upon the substantial research conducted thus far. This reflection is pivotal not only to acknowledge the depth of insights gleaned but also to reiterate the profound importance of these findings within the broader landscape of AMR.

The significance of this research effort lies in its potential to revolutionize the way AMRs operate and are integrated into various applications. By streamlining path planning algorithms and optimizing the workflow for constructing AMRs, this work stands as a testament to the commitment to advancing the field of robotics. These contributions serve to lower the entry barriers for individuals and organizations interested in harnessing the power of AMRs, ultimately democratizing access to these transformative technologies.

The Autonomous Mobile Robotics (AMR) field has witnessed remarkable progress, holding immense promise for the future. This dissertation embarks on a comprehensive exploration of this dynamic domain, with a particular emphasis on sensor technology, path planning, and navigation algorithm development. The core motivation for this research lies in the transformative impact AMRs have had on various industries, particularly exemplified by Amazon's acquisition of Kiva Systems and the subsequent proliferation of AMRs in warehouse operations. These developments have illuminated the potential for enhancing efficiency, productivity, and safety in industrial environments.

The primary contribution of this dissertation is the introduction of an innovative path

planning approach for AMRs. Traditionally, grid-based state-space representations have been a fundamental tool for optimal path planning. However, the limitations of this method become evident in large-scale and high-dimensional contexts, where computational demands escalate. To address this challenge, a novel hybrid method is proposed, combining the strengths of stochastic path planning algorithms with grid-based methods. This approach named FDA* (Focused Discretization A*) incrementally discretizes the state space, significantly reducing the computational resources required for path planning while still achieving suboptimal obstacle-free paths. By enhancing the efficiency and reducing memory consumption in AMR path planning algorithms, this contribution opens doors to a broader range of practical applications.

A secondary contribution revolves around streamlining the design and construction of AMRs. Historically, building autonomous mobile robots has been a resource-intensive and time-consuming endeavor, limiting their adoption and accessibility. This dissertation introduces an efficient workflow that commences with the improvement of classic self-balancing robot designs using Computer-Aided Design (CAD) tools. Additionally, it introduces the concept of a Robot Controller Board (RCB), which simplifies control system development, enhances efficiency, and eases troubleshooting. The workflow extends to a simulation environment created with ROS+Gazebo, where the robot's control system is realistically developed and tested before actual deployment. Furthermore, the construction of the self-balancing robot involves fabricating 3D model parts using a CNC machine, which are assembled without the need for adhesives, bolts, or nuts. Cost-effective components, readily available online or in local electronics retail shops, ensure the affordability of the robot, expanding its potential applications beyond traditional industrial settings.

In essence, this dissertation's contributions align harmoniously with the evolving landscape of AMRs, addressing critical challenges while embracing the transformative potential of these robotic systems. The overarching motivation stems from the dual objectives of enhancing industrial efficiency and democratizing the adoption of AMRs. By reducing the barriers to entry through streamlined design and construction workflows and innovating in path planning, this research aims to propel AMRs into a wider spectrum of applications, encompassing small retail businesses and even household use. Furthermore, it envisions the broader advancement of the field of robotics, not only by tackling immediate practical challenges but also by laying the foundation for future innovations and breakthroughs. This dissertation, at its core, embodies the spirit of progress, accessibility, and innovation in the realm of Autonomous Mobile Robotics. The horizons for future exploration and application of the methods developed within the framework of this dissertation are both expansive and promising. The culmination of this research effort marks not an endpoint but a launching pad into a myriad of potential avenues where these innovations can be harnessed to drive transformative change.

One of the foremost directions for future exploration lies in the domain of warehouse management. The methods and technologies refined in this dissertation hold immense potential to revolutionize the way warehouses are operated and managed. By integrating the optimized path planning algorithms and efficient robot construction workflows, warehouses can elevate their efficiency, reduce operational costs, and enhance inventory management. This translates to faster order fulfillment, reduced errors, and ultimately, an improved bottom line. The integration of AMRs can pave the way for highly adaptable and agile warehouse

operations, capable of swiftly adapting to changing demands and maintaining a competitive edge in the dynamic world of logistics.

Another compelling arena for the application of these methodologies is in search and rescue operations. The ability to navigate complex, dynamic, and hazardous environments is a paramount requirement in such scenarios. The path planning techniques developed in this dissertation, which minimize computational resources while ensuring efficient navigation, can be instrumental in the deployment of AMRs for swift and effective search and rescue missions. These robots, equipped with advanced perception and navigation technology, can navigate disaster-stricken areas with precision, locate survivors, and relay crucial information to first responders, potentially saving lives in critical situations.

Furthermore, the realm of education and research stands as a fertile ground for the application of these innovations. The development of autonomous mobile robotics platforms optimized for educational and research purposes is a noble endeavor. By simplifying the design and construction process of AMRs, educational institutions and researchers can empower students and professionals to explore the vast potential of robotics. These platforms can serve as hands-on learning tools, allowing individuals to experiment with various components and algorithms, fostering a deeper understanding of robotics principles. Such platforms also have the potential to accelerate research in the field, enabling researchers to quickly prototype and test new concepts, ultimately advancing the state of the art in autonomous mobile robotics.

In conclusion, the future venues for the application of the methods developed in this dissertation are rich and diverse. From warehouse management to search and rescue, and from educational platforms to cutting-edge research, the impact of these innovations is poised to be profound. It is in these future endeavors that the true significance of this research will be fully realized, as it catalyzes progress and innovation across a wide spectrum of industries and domains, ultimately enriching our lives and the way we interact with autonomous mobile robots.

Bibliography

[1] I. Spectrum. Meet the roomba's ancestor: The cybernetic tortoise - ieee spectrum. https://spectrum.ieee.org/meet-roombas-ancestor-cybernetic-tortoise. Accessed 26 May 2023.

[2] I. Rani, "A model and formal analysis of braitenberg vehicles 2 and 3," in *IEEE International Conference on Robotics and Automation*, 2012.

[3] (2023) Eos 5d mark iv with canon log. https://www.usa.canon.com/shop/p/eos-5d-mark-iv-with-canon-log?color=Black&type=Kit. Accessed: Apr. 08, 2023.

[4] R. P. Ltd. (2023) Buy a raspberry pi camera module 2. https://www.raspberrypi.com/products/camera-module-v2/. Accessed: Apr. 08, 2023.

[5] Intel® RealSense™ Depth and Tracking Cameras. (2023) Stereo depth solutions from intel realsense. https://www.intelrealsense.com/stereo-depth/. Accessed: Apr. 08, 2023.

[6] e-con Systems. 3d tof mipi camera for nvidia jetson agx orin / agx xavier. https://www.e-consystems.com/3d-depth-cameras/tof-cameras/3d-mipi-tof-camera-nvidia-jetson-agx-orin-xavier.asp. Accessed on Aug. 18, 2023.

[7] B. Dynamics. Atlas. https://bostondynamics.com/atlas/. Accessed Aug. 16, 2023.

[8] C. Robotics. Husky ugv - outdoor field research robot by clearpath. https://clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/. Accessed Aug. 16, 2023.

[9] PB Tech. Buy the dji mavic 3 drone cine premium combo hasselblad camera (cp.ma.00000458.01) online. https://www.pbtech.co.nz/product/DRODJI0007/DJI-Mavic-3-Drone-Cine-Premium-Combo-Hasselblad-Ca. Accessed Aug. 16, 2023.

[10] A. Technology. Talon tracked military robot. https://www.army-technology.com/projects/talon-tracked-military-robot/. Accessed Aug. 16, 2023.

[11] Bluerov2. https://bluerobotics.com/store/rov/bluerov2/. Blue Robotics. Accessed Aug. 16, 2023.

[12] M. Boumediene, S. Laouar, D. L. Mehennaoui, and P. S. Ouchtati, "Design, simulation and control of a self-balancing robot in a gazebo environment and ros2 framework," in *International Conference on Technological Advances in Electrical Engineering (ICTAEE)*, May 23 2023.

[13] M. Boumediene, N. Zeghida, B. Manaa, and D. L. Mehennaoui, "Design, construction and control of a self-balancing robot including a new frame assembly approach and a custom pcb," in *International Conference on Technological Advances in Electrical Engineering*, May 24 2023.

[14] "Kiva systems: Three engineers, hundreds of robots, one warehouse," *IEEE Spectrum*, 2023, accessed: Dec. 25, 2023. [Online]. Available: https://spectrum.ieee.org/three-engineers-hundreds-of-robots-one-warehouse

[15] (2023) Innovative human-robot cooperation in bmw group production. https://www.press.bmwgroup.com/global/article/detail/T0209722EN/innovative-human-robot-cooperation-in-bmw-group-production?language=en. Accessed: Dec. 25, 2023.

[16] R. Moore-Colyer. (2023) General motors connects quarter of its robot workforce to the internet. https://www.silicon.co.uk/data-storage/bigdata/general-motors-robots-iot-208671. Accessed: Dec. 25, 2023.

[17] (2021, Nov) The robots are coming. https://www.bloomberg.com/news/newsletters/2021-11-08/what-s-happening-in-the-world-economy-the-robots-are-coming. Accessed: Dec. 25, 2023.

[18] (2023) Bringing underserved communities life-saving aid through aerial logistics. https://www.science.org/doi/10.1126/scirobotics.adm7020. Accessed: Dec. 25, 2023.

[19] (2023) Logistics robots market. https://www.futuremarketinsights.com/reports/logistics-robots-market. Accessed: Dec. 25, 2023.

[20] (2023) Using robotics to supercharge health care. https://news.mit.edu/2023/using-robotics-supercharge-health-care-0123. Accessed: Dec. 25, 2023.

[21] (2023) Integrating robotics into wildlife conservation: Enhancing predator deterrents through innovative movement strategies. https://phys.org/news/2023-06-robotics-wildlife-predator-deterrents-movement.html. Accessed: Dec. 25, 2023.

[22] S. W. Breck, J. T. Schultz, D. Prause, C. Krebs, A. J. Giordano, and B. Boots, "Integrating robotics into wildlife conservation: testing improvements to predator deterrents through movement," *PeerJ*, vol. 11, p. e15491, Jun 2023.

[23] (2023) Global agriculture robots research report 2023-2028: Explore the unmanned future of farming - driverless tractors, drones and precision robots redefining the burgeoning market. https://finance.yahoo.com/news/global-agriculture-robots-research-report-160000987.html. Accessed: Dec. 25, 2023.

[24] (2023) Harvest croo robotics. https://www.harvestcroorobotics.com. Accessed: Dec. 25, 2023.

[25] (2023) Aethon - autonomous mobile robots - healthcare and hospitality. https://aethon.com/. Accessed: Dec. 25, 2023.

[26] (2023) Mobile industrial robots | automate your internal transportation. https://www.mobile-industrial-robots.com/. Accessed: Dec. 25, 2023.

[27] (2023) Spot. https://bostondynamics.com/products/spot/. Accessed: Dec. 25, 2023.

[28] (2023) Skydio r1 review. https://www.pcmag.com/reviews/skydio-r1. Accessed: Dec. 25, 2023.

[29] C. Grey Walter, "Biography essay. encyclopedia of cognitive science," *Unknown*, vol. 4, pp. 537–539, 2003.

[30] I. Rano, "A model and formal analysis of braitenberg vehicles 2 and 3," in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 910–915.

[31] K. McGuire, G. de Croon, and K. Tuyls, "A comparative study of bug algorithms for robot navigation," *Robotics and Autonomous Systems*, vol. 121, p. 103261, Nov 2019.

[32] C. Stachniss, *Robotic Mapping and Exploration*. Springer Berlin Heidelberg, 2009, vol. 55.

[33] N. J. Nilsson, "Shakey the robot," Stanford University, Tech. Rep. Technical Report No. 323, 1984.

[34] Ocean infinity - ocean infinity. https://oceaninfinity.com/. Accessed: Sep. 29, 2023.

[35] Roomba® robot vacuum cleaners | irobot®. https://www.irobot.com/en_US/us-roomba.html. Accessed: Sep. 29, 2023.

[36] Anymal x. https://www.anybotics.com/robotics/anymal-x/. ANYbotics. Accessed: Sep. 29, 2023.

[37] What is da vinci robotic surgery? a complete overview. https://www.intuitive.com/en-us/patients/da-vinci-robotic-surgery. Accessed: Sep. 29, 2023.

[38] S. Thrun, "Robotic mapping: A survey," *Exploring artificial intelligence in the new millennium*, vol. 1, no. 1, pp. 1–35, 2002.

[39] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[40] P. Corke, *Robotics, Vision and Control*. Springer International Publishing, 2017, vol. 118.

[41] S. Thrun, "Learning occupancy grid maps with forward sensor models," in *Autonomous Robots*. Springer, 2001, pp. 25–32.

[42] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot, "Consistency of the ekf-slam algorithm," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 3562–3568.

[43] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," *Autonomous robot vehicles*, vol. 3, no. 1, pp. 167–193, 1987.

[44] K. Murphy and S. Russell, "Rao-blackwellised particle filtering for dynamic bayesian networks," in *Sequential Monte Carlo Methods in Practice*, A. Doucet, N. de Freitas, and N. Gordon, Eds. New York, NY: Springer New York, 2001, pp. 499–515.

[45] M. Montemerlo and S. Thrun, "Simultaneous localization and mapping with unknown data association using FastSLAM," in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, Taipei, Taiwan, 2003, pp. 1985–1991.

[46] J. Folkesson and H. Christensen, "Graphical SLAM - a self-correcting map," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04.*, New Orleans, LA, USA, 2004, pp. 383–390 vol.1.

[47] G. Dubbelman and B. Browning, "COP-SLAM: Closed-form online pose-chain optimization for visual SLAM," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1194–1213, 2015.

[48] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2007, p. 6.

[49] S. Kohlbrecher, O. von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, Nov. 2011, pp. 155–160.

[50] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 4758–4765.

[51] S. Pfister, S. Roumeliotis, and J. Burdick, "Weighted line fitting algorithms for mobile robot map building and efficient data representation," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 1, 2003, pp. 1304–1311.

[52] A. Siadat, A. Kaske, S. Klausmann, M. Dufaut, and R. Husson, "An optimized segmentation method for a 2d laser-scanner applied to mobile robot navigation," *IFAC Proceedings Volumes*, vol. 30, no. 7, pp. 149–154, 1997.

[53] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[54] R. Mur-Artal, J. Montiel, and J. D. Tardós, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.

[55] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *Computer Vision–ECCV 2014*, vol. 8690. Springer, 2014, pp. 834–849.

[56] B. Pfrommer and K. Daniilidis, "Tagslam: Robust slam with fiducial markers," *arXiv preprint arXiv:1910.00679*, 2019.

[57] K. Tateno, F. Tombari, I. Laina, and N. Navab, "CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction," *arXiv*, vol. abs/1704.03489, 2017, accessed: Dec. 15, 2022. [Online]. Available: http://arxiv.org/abs/1704.03489

[58] L. Xiao, J. Wang, X. Qiu, Z. Rong, and X. Zou, "Dynamic-SLAM: Semantic monocular visual localization and mapping based on deep learning in dynamic environment," *Robotics and Autonomous Systems*, vol. 117, pp. 1–16, 2019.

[59] R. Clark, M. Bloesch, J. Czarnowski, S. Leutenegger, and A. J. Davison, "Ls-net: Learning to solve nonlinear least squares for monocular stereo," *arXiv preprint arXiv:1809.02966*, 2018. [Online]. Available: http://arxiv.org/abs/1809.02966

[60] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *arXiv preprint arXiv:1612.00593*, 2017.

[61] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *arXiv preprint arXiv:1706.02413*, 2017. [Online]. Available: http://arxiv.org/abs/1706.02413

[62] B. Kaleci, K. Turgut, and H. Dutağacı, "2dlasernet: A deep learning architecture on 2d laser scans for semantic classification of mobile robot locations," *Engineering Science and Technology, an International Journal*, vol. 28, pp. 1034–1044, Jun. 2021.

[63] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, Dec 1959.

[64] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, pp. 100 – 107, 08 1968.

[65] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proceedings. 1985 IEEE international conference on robotics and automation*, vol. 2. IEEE, 1985, pp. 500–505.

[66] R. E. Korf, "Depth-first iterative-deepening: An optimal admissible tree search," *Artificial Intelligence*, vol. 27, no. 1, pp. 97–109, 1985. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0004370285900840

[67] S. Russell, "Efficient memory-bounded search methods," in *Proceedings of the 10th European Conference on Artificial Intelligence*. Vienna, Austria: John Wiley & Sons, New York, NY, 1992, pp. 1–5. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.105.7839

[68] D. Harabor and A. Grastien, "Online graph pruning for pathfinding on grid maps," *AAAI*, vol. 25, no. 1, pp. 1114–1119, Aug 2011.

[69] A. Stentz, "The d* algorithm for real-time planning of optimal traverses," Carnegie Mellon University, Tech. Rep., 1994.

[70] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 354–363, 2005.

[71] A. Stentz, "The focussed d* algorithm for real-time replanning," in *International Joint Conference on Artificial Intelligence*, Aug 1995, accessed: May 30, 2023. [Online]. Available: https://www.semanticscholar.org/paper/The-Focussed-D*-Algorithm-for-Real-Time-Replanning-Stentz/4d95e5c7c3b7d8a2a8a273e69a5b4fb98557b912

[72] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[73] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Unknown, Unknown, Technical Report Unknown, Unknown 1998.

[74] L. Janson, E. Schmerling, A. Clark, and M. Pavone. (2015, Feb) Fast marching tree: a fast marching sampling-based method for optimal motion planning in many dimensions. https://arxiv.org/abs/1306.3532. ArXiv preprint arXiv:1306.3532.

[75] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, "Maprm: a probabilistic roadmap planner with sampling on the medial axis of the free space," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, May 1999, pp. 1024–1031 vol.2.

[76] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 4569–4574.

[77] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, Jun 2011.

[78] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 2997–3004.

[79] ——, "Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 3067–3074.

[80] P. Bhattacharya and M. Gavrilova, "Voronoi diagram in optimal path planning," in *Proceedings - ISVD 2007 The 4th International Symposium on Voronoi Diagrams in Science and Engineering*, Jul 2007, pp. 38–47.

[81] K. Åström and T. Hägglund, *PID Controllers: Theory, Design, and Tuning*. ISA - The Instrumentation, Systems and Automation Society, 1995.

[82] H. Wu, W. Su, and Z. Liu, "Pid controllers: Design and tuning methods," in *2014 9th IEEE Conference on Industrial Electronics and Applications*, 2014, pp. 808–813.

[83] P. K. Padhy, T. Sasaki, S. Nakamura, and H. Hashimoto, "Modeling and position control of mobile robot," in *2010 11th IEEE International Workshop on Advanced Motion Control (AMC)*, 2010, pp. 100–105.

[84] C. S. Shijin and K. Udayakumar, "Speed control of wheeled mobile robots using pid with dynamic and kinematic modeling," in *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, 2017, pp. 1–7.

[85] H. Xiaoqian, H. Karki, A. Shukla, and Z. Xiaoxiong, "Variant pid controller design for autonomous visual tracking of oil and gas pipelines via an unmanned aerial vehicle," in *2017 17th International Conference on Control, Automation and Systems (ICCAS)*, 2017, pp. 368–372.

[86] C. Chandni, V. Sajith Variyar, and K. Guruvayurappan, "Vision based closed loop pid controller design and implementation for autonomous car," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2017, pp. 1928–1933.

[87] T. Sangyam, P. Laohapiengsak, W. Chongcharoen, and I. Nilkhamhang, "Path tracking of uav using self-tuning pid controller based on fuzzy logic," in *Proceedings of SICE Annual Conference 2010*, 2010, pp. 1265–1269.

[88] K. Warwick and B. Minbashian, "Intelligent parallel control," in *Advanced Methods in Adaptive Control for Industrial Applications*, K. Warwick, M. Kárný, and A. Halousková, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 258–276.

[89] T. Fukao, H. Nakagawa, and N. Adachi, "Adaptive tracking control of a nonholonomic mobile robot," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 5, pp. 609–615, 2000.

[90] J.-M. Yang and J.-H. Kim, "Sliding mode control for trajectory tracking of nonholonomic wheeled mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 3, pp. 578–587, 1999.

[91] F. Lin, Z. Lin, and X. Qiu, "Lqr controller for car-like robot," in *2016 35th Chinese Control Conference (CCC)*, 2016, pp. 2515–2518.

[92] J. B. Rawlings, "Tutorial overview of model predictive control," *IEEE Control Systems Magazine*, vol. 20, no. 3, pp. 38–52, June 2000.

[93] Rover wheels - NASA mars. https://mars.nasa.gov/mars2020/spacecraft/rover/wheels/. Accessed Jan. 17, 2023.

[94] U. Nagarajan, G. Kantor, and R. Hollis, "The ballbot: An omnidirectional balancing mobile robot," *The International Journal of Robotics Research*, vol. 33, no. 6, pp. 917–930, May 2014.

[95] Á. Odry, R. Fullér, I. J. Rudas, and P. Odry, "Fuzzy control of self-balancing robots: A control laboratory project," *Computer Applications in Engineering Education*, vol. 28, no. 3, pp. 512–535, 2020.

[96] (2023) Ninebot s | self-balancing scooter | segway official store. Online. Accessed on December 25, 2023. [Online]. Available: https://store.segway.com/ninebot-s

[97] B. Dynamics. (2023) Legacy robots. Online. Accessed on December 25, 2023. [Online]. Available: https://bostondynamics.com/legacy/

[98] (2023) Wowwee®- mip. Online. Accessed on December 25, 2023. [Online]. Available: http://wowwee.com/mip

[99] H. Hellman and H. Sunnerman, "Two-wheeled self-balancing robot," Bachelor's Thesis in Mechatronics, KTH Royal Institute of Technology, Stockholm, Sweden, 5 2015, tRITA MMK 2015:8 MDAB061.

[100] L. Gabrić. (Year you accessed the repository) Arduino - self balancing robot. https://github.com/lukagabric/Franko/tree/master. Accessed: September 2, 2023.

[101] I. Al Bluwi, T. Simeon, and J. Cortes, "Motion planning algorithms for molecular simulations: A survey," *Computer Science Review*, vol. 6, no. 4, pp. 125–143, 2012.

[102] X. Tang, B. Kirkpatrick, S. Thomas, G. Song, and N. M. Amato, "Using motion planning to study rna folding kinetics," *Journal of Computational Biology*, vol. 12, no. 6, pp. 862–881, 2005.

[103] M. C. Lin, A. Sud, J. Van den Berg, R. Gayle, S. Curtis, H. Yeh, S. Guy, E. Andersen, S. Patil, J. Sewall, and D. Manocha, "Real time path planning and navigation for multi-agent and crowd simulations," in *Motion in Games*, A. Egges, A. Kamphuis, and M. Overmars, Eds. Berlin, Heidelberg: Springer, 2008, pp. 23–32.

[104] D. Thalmann and S. Raupp Musse, *Crowd Simulation*. London: Springer London, 2007.

[105] V. Bulitko, Y. Bjo rnsson, N. R. Sturtevant, and R. Lawrence, "Real time heuristic search for path finding in video games," in *Artificial Intelligence for Computer Games*, P. A. Gonzalez Calero and M. A. Gomez Martın, Eds., 2011, pp. 1–30.

[106] S. Karaman and E. Frazzoli, "Sampling based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[107] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt*," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 1478–1483.

[108] M. Boumediene, L. Mehennaoui, and A. Lachouri, "FDA*: A focused single-query grid based path planning algorithm," *Journal of Automation, Mobile Robotics & Intelligent Systems (JAMRIS)*, vol. 3, no. 2021, pp. 37–43, May 2022.