

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Comparison between an AGV fleet and an AMR fleet

Rita Tomé de Almeida e Silva



Master in Electrical and Computer Engineering

Supervisor: Professor Pedro Gomes da Costa

Co-Supervisor: Héber Miguel Sobreira

October 31, 2024

Resumo

A indústria tem atravessado avanços tecnológicos significativos, mais recentemente impulsionados em parte pela pandemia da COVID-19, que motivou a utilização de robôs para compensar a redução do trabalho humano. No mundo pós-pandêmico, esta evolução continua a requerer adaptação constante às exigências do mercado em permanente mudança. Neste contexto, os Autonomous Mobile Robots (AMRs) e os Automated Guided Vehicles (AGVs) surgiram como inovações chave, cada vez mais utilizados em vários setores industriais devido à sua eficiência e versatilidade. Estas tecnologias são especialmente relevantes em tarefas de transporte dentro de linhas de produção e sistemas de armazenamento.

Os AGVs, introduzidos pela primeira vez na década de 1950, garantem alta precisão e eficiência, mas carecem de flexibilidade e são inadequados para ambientes dinâmicos. Em contraste, os AMRs representam uma inovação recente que utiliza sensores, câmaras e algoritmos sofisticados para navegar de forma autónoma, sem caminhos fixos, tornando-os adaptáveis a ambientes mutáveis. Os AMRs oferecem vantagens significativas no que toca à sua flexibilidade e adaptabilidade, tornando-os ideais para ambientes dinâmicos e complexos.

O objetivo deste trabalho focou-se na exploração das complexidades de coordenação de uma frota de robôs, em particular na comparação entre frotas de AGVs e AMRs em ambientes fechados. Para tal, foi realizada uma comparação abrangente entre frotas de AGVs e frotas de AMRs num ambiente simulado controlado. A pesquisa analisou algoritmos para a navegação autónoma de AMRs e testou diferentes tamanhos de frota para AGVs e AMRs. Os resultados indicaram que os AGVs são superiores aos AMRs relativamente a fatores como a taxa de sucesso das missões e conclusão de tarefas, embora os AMRs demonstrassem maior eficiência em termos de tempo. Este trabalho também destaca a importância da exploração contínua na gestão de frotas robóticas.

Abstract

The industry is currently experiencing significant technological advancements, driven in part by the COVID-19 pandemic, which necessitated the use of robots to compensate for reduced human labor. In the post-COVID-19 world, this evolution continues to require constant adaptation to ever-changing market demands. Within this context, Autonomous Mobile Robots (AMRs) and Automated Guided Vehicles (AGVs) have emerged as key innovations, increasingly used across various sectors for their efficiency and versatility. These technologies are especially relevant in transportation tasks within production lines and warehouse systems.

AGVs, which have been first introduced in the 1950s ensure high accuracy and efficiency but lack flexibility and are unsuitable for dynamic settings. In contrast, AMRs represent a recent innovation that uses sensors, cameras, and sophisticated algorithms to navigate autonomously without fixed paths, making them adaptable to changing environments. AMRs offer significant advantages in flexibility and adaptability, making them ideal for dynamic and complex environments.

The aim of this work was to explore the complexities of coordinating a fleet of robots, particularly by comparing fleets of AGVs and AMRs in closed environments. Thus, a comprehensive comparison was made between fleets of AGVs and fleets of AMRs in a controlled simulated environment. The research analyzed local planners for AMR navigation and tested different fleet sizes for both AGVs and AMRs. The results indicated that AGVs generally outperformed AMRs in terms of success rate and task completion, although AMRs demonstrated greater time efficiency. This work also underscores the importance of ongoing exploration in robotic fleet management.

Sustainable Development Goals

In 2015, the 2030 Agenda for Sustainable Development, a plan of action for peace and prosperity, keeping the people and planet in mind, tackling climate change and working to preserve oceans and forests, was adopted by all United Nations Member States. [1] The greatest challenge the United Nations recognizes is poverty, and eradicating it will stimulate sustainable development.

History and Background

The Millennium Development Goals were a set of 8 goals that preceded the Sustainable Development Goals. These goals were to be achieved by 2015 and counted with the commitment of all 191 United Nations member states members.

Big improvements were made in the effort to achieve the MDGs. In 2010, even before the 2015 deadline, the target 7C was partially achieved, a target on access to safe drinking water - to halve the proportion of the population without sustainable access to safe drinking water. This target includes providing the population with basic sanitation, but this part of the target wasn't met.

Additionally, new HIV infections declined 38% between 2001 and 2013, the percentage of undernutrition in children dropped from 28% in 1990 to 17% in 2013, and the number of infant deaths decreased from 12.7 million to 6.3 million between 1990 and 2013.

In 2015, the SDGs were adopted by all the United Nations members and in turn this initiative comprised of 17 goals that connect environmental, social and economic aspects of sustainable development.

Actuality

The 17 SDGs' short titles are No poverty (SDG 1), Zero hunger (SDG 2), Good health and well-being (SDG 3), Quality education (SDG 4), Gender equality (SDG 5), Clean water and sanitation (SDG 6), Affordable and clean energy (SDG 7), Decent work and economic growth (SDG 8), Industry, innovation and infrastructure (SDG 9), Reduced inequalities (SDG 10), Sustainable cities and communities (SDG 11), Responsible consumption and production (SDG 12), Climate action (SDG 13), Life below water (SDG 14), Life on land (SDG 15), Peace, justice, and strong institutions (SDG 16), and Partnerships for the goals (SDG 17).

The ambition of these goals combined with the COVID-19 pandemic from 2020 to 2023, which affected all 17 of them, resulted in it being a bigger challenge to meet these goals by 2030 but a global effort is being made for the SDGs to prioritize the needs for this plan.

Thesis Contextualization

In the context of this work, it is believed that it could impact some SDGs.

Table 1: SDG Analysis

SDG	Goal	Contribution	Performance Indicators and Metrics
Target 9: Industry, Innovation and Infrastructure	9.4 - By 2030, upgrade infrastructure and retrofit industries to make them sustainable, with increased resource-use efficiency and greater adoption of clean and environmentally sound technologies and industrial processes, with all countries taking action in accordance with their respective capabilities	The study made helps fulfill this goal. With the analysis and improvement of fleets of AGVs and AMRs, the future of the industry is improved and more advanced technologies can be used in industrial processes.	9.4.1 - CO2 emission per unit of value added
	9.a - Facilitate sustainable and resilient infrastructure development in developing countries through enhanced financial, technological and technical support to African countries, least developed countries, landlocked developing countries and small island developing States	With the progress in industrial processes, namely in warehouses and factories, infrastructure development in developing countries can be enabled greatly, if introduced. This would support countries that are less developed.	9.a.1 - Total official international support (official development assistance plus other official flows) to infrastructure
Target 8: Decent Work and Economic Growth	8.2 - Achieve higher levels of economic productivity through diversification, technological upgrading and innovation, including through a focus on high-value added and labour-intensive sectors	Economy productivity can be achieved by decreasing labour-intensive and repetitive jobs and easing the duty for workers by implementing vehicles, which constitutes a technological upgrade in the professional workplace.	8.2.1 - Annual growth rate of real GDP per employed person

Acknowledgments

Firstly, I would like to thank my supervisor, Professor Pedro Gomes da Costa, and my co-supervisor, Héber Miguel Sobreira, for always being available to answer my questions and for all the support given throughout the elaboration of my thesis.

Furthermore, I have to extend my thanks to Marina Brilhante, who greatly supported me during the last few months with exceptional patience, kindness and accessibility and with whom the elaboration of this thesis presented less difficulties.

Above all, I express my deepest gratitude to my parents, Fernanda and Rui, who have not only provided an essential support system throughout all my academic journey and particularly during these past five years but also served as role models, instilling in me strong values and a positive outlook on life. Equally, to my sister, Mafalda, whose unwavering support and encouragement provided for me has been invaluable.

Moreover, I could not but offer my heartfelt thanks to Mariana Almeida, my most treasured friend, who accompanied me through all the most significant and insignificant moments of my life during the last few years, who tirelessly supported me through all the mishaps of my academic and personal life and showed nothing but sincere friendship and altruism in order for me to achieve all my personal goals.

Last but not least, I would like to thank all my friends who supported me in the elaboration of this thesis, and with whose support I could have never go without. I am always grateful to be surrounded by people who help me grow not only academically but also as a person.

Rita Tomé de Almeida e Silva

*“À dolorosa luz das grandes lâmpadas eléctricas da fábrica
Tenho febre e escrevo.
Escrevo rangendo os dentes, fera para a beleza disto,
Para a beleza disto totalmente desconhecida dos antigos.*

*Ó rodas, ó engrenagens, r-r-r-r-r eterno!
Forte espasmo retido dos maquinismos em fúria!
Em fúria fora e dentro de mim,
Por todos os meus nervos dissecados fora,
Por todas as papilas fora de tudo com que eu sinto!
Tenho os lábios secos, ó grandes ruídos modernos,
De vos ouvir demasiadamente de perto,
E arde-me a cabeça de vos querer cantar com um excesso
De expressão de todas as minhas sensações,
Com um excesso contemporâneo de vós, ó máquinas!”*

Álvaro de Campos
Ode Triunfal

Contents

1	Introduction	1
1.1	Context and Motivation	1
1.2	Objectives	2
1.3	Structure	2
2	State of the Art	5
2.1	AGV Technology and Navigation	7
2.1.1	Navigation Types	9
2.1.2	Localization Systems	10
2.1.3	Locomotion Types	12
2.2	AMR Technology and Navigation	15
2.2.1	Localization Systems	15
2.2.2	Locomotion Types	16
2.3	Multi-robot systems	17
2.3.1	Heterogeneous and Homogeneous Fleets	18
2.4	Graph Search Algorithms	19
2.4.1	A*	19
2.4.2	Time Enhanced A*	19
2.5	Methods for Obstacle Avoidance	21
2.5.1	Reactive Methods	21
2.5.2	Natural Computing Algorithms	22
2.6	Fleet Managers in Path-planning Navigation	24
2.6.1	Industrial Solutions	24
2.7	ROS	26
2.7.1	ROS Navigation Stack	26
2.8	Synthesis and Relevance to the Study	27
3	Algorithms and Fleet Manager for Autonomous Navigation	29
3.1	Base Local Planner	30
3.2	TEB Local Planner	31
3.3	DWA Local Planner	34
3.4	Discussion of Results	37
3.5	Fleet Manager for Autonomous Navigation	39
4	Discussion and Results	43
4.1	Results without Fleet Manager	43
4.1.1	Comparisons between one AMR and one AGV	44
4.1.2	Comparisons between a fleet of five AMRs and five AGVs	47

4.1.3	Comparisons between a fleet of ten AMRs and ten AGVs	49
4.2	Results with Fleet Manager	53
4.2.1	Comparison between a fleet of five AMRs and five AGVs	53
4.3	Tests with a real robot	55
4.4	Discussion of results	59
4.4.1	Results without a Fleet Manager	60
4.4.2	Results with a Fleet Manager	61
4.4.3	Results of the tests with a real robot	62
5	Conclusions and Future Work	63
5.1	Goals Accomplishment	63
5.2	Future Work	64

List of Figures

2.1	Comparison Matrix for different transportation systems [6]	6
2.2	Automated Guided Cart. [14]	8
2.3	Forklift AGV. [15]	8
2.4	Assembly Line AGV. [16]	9
2.5	Some AGV navigation types. [18]	10
2.6	Overview of an omnidirectional wheeled AGV. [31]	12
2.7	Overview of an omnidirectional wheeled AGV. [32]	13
2.8	Analysis of various Hybrid Locomotion vehicles: Mobility in unstructured environments compared to speed and energetic efficiency. [34]	13
2.9	Hybrid Wheel-Tracked AGV motion in a flight of stairs [35]	14
2.10	Different types of AGVs.	14
2.11	Autonomous Mobile Robot in a warehouse setting. [38]	15
2.12	Different types of methods of MRS (Adapted from [50])	17
2.13	Example of a graph search algorithm.	20
2.14	The input map and the analysed neighbor cells focusing the cell with the same position of the AGV. [60]	20
2.15	Artificial Potential Field. (Adapted from [65])	22
2.16	Ant Colony Algorithm. (Adapted from [71])	23
3.1	Robot using TEB Local Planner to go around obstacle successfully.	33
3.2	Robot passing through previously detected obstacle.	34
3.3	AMR with Base Local Planner.	37
3.4	AMR with DWA Local Planner.	38
3.5	AMR with TEB Local Planner.	38
3.6	Laser Callback.	40
3.7	Number of positions in the range array calculated for the 90-degree angle.	41
3.8	Function that detects if two robots are within 1.5m of each other.	41
3.9	Main.	42
4.1	Map used for the simulations of both AGVs and AMRs in RViz.	43
4.2	Start and end vertice for the small mission.	44
4.3	Start and end vertice for the medium mission.	44
4.4	Start and end vertice for the long mission.	44
4.5	Trajectory followed by the guided vehicle to fulfill the mission.	46
4.6	Trajectory followed by the autonomous vehicle to fulfill the mission.	46
4.7	Five robots performing the mission with free navigation.	48
4.8	Trajectory followed by the guided vehicles to fulfill the mission.	48
4.9	Five robots following a set path to perform the mission.	49

4.10	Orange robot attempting to finish trajectory.	50
4.11	Final positions for the fleet of ten robots.	51
4.12	Fleet of ten robots following the autonomous trajectory to achieve their goals. . .	52
4.13	Fleet of ten robots performing the mission using guided navigation.	52
4.14	Goals planned for the guided navigation of the fleet of ten robots.	52
4.15	Robot used for trajectory testing.	55
4.16	Robot moving forward in the real setting and in the simulation.	57
4.17	Robot deflecting the static obstacle successfully.	57
4.18	Robot avoiding the dynamic obstacle.	58
4.19	Robot moving around an obstacle while choosing the clear pathway.	59
4.20	Average Arrival Time for the first and last robot for each of the missions.	61

List of Tables

1	SDG Analysis	vi
4.1	Simulation time results for the three distance missions, for the AMR and AGV.	46
4.2	Simulation results for the fleet of five free navigation robots.	47
4.3	Simulation results for the fleet of five guided navigation robots.	49
4.4	Simulation results for the fleet of ten free navigation robots.	51
4.5	Simulation results for the fleet of ten guided navigation robots.	53
4.6	Simulation results for the fleet of five free navigation robots with the fleet manager.	54
4.7	Simulation results for the fleet of five guided navigation robots with the improved TEA* algorithm.	55

Abbreviations and Symbols

ACO	Ant Colony Optimization
AGC	Automated Guided Carts
AGV	Automated Guided Vehicles
AMR	Autonomous Mobile Robot
APF	Artificial Potential Field
AUV	Autonomous Underwater Vehicles
CAD	Computer-Aided Design
CMVRP	Capacitated Multi-Vehicle Routing Problem
DWA	Dynamic Window Approach
FAEIF	Fizzy Adaptive Extended Information Filtering
FOV	Field of View
GA	Genetic Algorithm
GPS	Global Positioning System
HIV	Human Immunodeficiency Virus
HTA	Heavier Than Air
INU	Inertial Navigation Units
INS	Inertial Navigation Systems
KNN	K-Nearest Neighbors
KPI	Key Performance Indicators
LGV	Laser Guided Vehicles
LiDAR	Light Detection and Ranging
LTA	Lighter Than Air
MDG	Millennium Development Goals
MLP	Multilayer Perceptron
MRS	Multi-robot System
OPF	Optimum-Path Forest
RFID	Radio-Frequency Identification
ROS	Robot Operating System
SDG	Sustainable Development Goals
SLAM	Simultaneous Localization and Mapping
SVM	Support Vector Machine
TEA*	Time Enhanced A*
TEB	Timed Elastic Band
TF	Transform
ToA	Time of Arrival
UAV	Unmanned Aerial Vehicle
VFH	Vector Field Histogram
WLAN	Wireless Local Area Network

Chapter 1

Introduction

1.1 Context and Motivation

The industry is currently undergoing a period of significant technological advancement, particularly due to COVID-19, which made it necessary for robots to take place when human work was down. Therefore, in a post-COVID-19 world this evolution was productive and there needs to be a continuous adaptation to the ever-evolving market demands. Companies are being forced to innovate and find new ways to add value to their products so there has been product enhancement and growth. Within this context, Autonomous Mobile Robots (AMRs) and Automated Guided Vehicles (AGVs) have emerged as a key technological innovation, increasingly utilized across various sectors for their efficiency and versatility and studies that lean into these vehicles are needed.

AMRs and AGVs are primarily used in the transportation of goods between production lines or warehouse systems, and their applications extend to environments such as healthcare, hospitality, manufacturing systems and distribution centers.

AGVs, introduced in 1953 [2], operate by following predefined paths marked by wires, magnetic strips, or other physical guides. AGVs are typically used in environments with well-defined and repetitive tasks, where there are minimal changes to the layout. By following a given path, they ensure high accuracy and efficiency in their tasks but are not flexible to dynamic objects and environments, and can't be in a shared environment with a lot of AGVs, depending on how the environment is set.

On the other hand, the recent innovation of AMRs use sensors, cameras, and sophisticated algorithms to perceive and navigate their surroundings autonomously. Unlike AGVs, AMRs do not make use of fixed paths or physical guides as well as they are adaptable to their environments, making them highly flexible and suitable for dynamic settings. Despite their higher initial costs and the need for more sophisticated infrastructure and management systems, AMRs are ideal for environments that need high flexibility and adaptability.

The question that imposes is the ability and effectiveness of implementing these robot vehicles on a large scale. Coordinating the movement of a fleet of robots in confined spaces is a complex

task and its study is important to guarantee the growth in performance and efficiency of these vehicles in industrial environments.

Both AGVs and AMRs are utilized in material handling systems. However, their mechanisms, flexibility, and suitability for various environments can differ greatly, which poses the question for their different effectiveness and productivity in given situations. Understanding these differences is crucial to decide which technology to use in different industrial contexts.

To address this challenge, this research proposes a comparison between fleets of autonomous vehicles and guided vehicles when not equipped and when equipped with a fleet management system. This work aims to enhance fleet management's flexibility, efficiency, and robustness by comparing fleets of AGVs and AMRs while analyzing and addressing the possibilities of fleet management between both types of robots. It involves creating algorithms and testing different solutions, ensuring optimized and collision-free navigation paths, thereby improving overall productivity and efficiency in industrial settings.

1.2 Objectives

As both AGVs and AMRs continue to evolve, understanding the comparative advantages and limitations of each technology becomes increasingly important.

Comparative studies between fleets of AGVs and AMRs can provide critical insights into operational efficiency, cost-benefit analysis, flexibility, scalability, and implementation challenges.

Evaluating the performance of AGVs and AMRs in different operational settings helps determine which technology offers greater efficiency and productivity. These different settings vary according to different maps, the presence of dynamic obstacles in the way and the use of fleet management systems to help vehicles navigate more effectively and without collisions.

Assessing the adaptability of AGVs and AMRs to these changing operational needs provides insights into their scalability and potential for future growth. Identifying the technical and logistical challenges associated with deploying AGVs and AMRs helps in planning and mitigating risks.

1.3 Structure

The current chapter serves as an introductory chapter for this dissertation, with a contextualization of the subject matter in the current world and a description of the motivation and needs for the elaboration of the work in section 1.1. Additionally, the objectives established for this work are described in section 1.2.

Chapter 2 provides an overview of the two types of robots explored in this work: AGVs and AMRs. It thoroughly examines the technologies behind these vehicles and discusses current research relevant to comparing them.

Additionally, the chapter reviews the graph search algorithms commonly used by AGVs, particularly focusing on the TEA* algorithm implemented in this work. It also addresses methods for obstacle avoidance in AMRs.

Furthermore, the chapter analyzes multi-robot systems and discusses fleet management systems, referencing several industrial solutions. Finally, it contextualizes the simulation environment and tools utilized in this research.

In chapter 3, various algorithms for autonomous navigation are examined, along with a section that presents the findings from this analysis. Furthermore, this chapter also includes a description of the fleet manager developed for the AMR fleet.

The results of this work are presented in chapter 4, which presents the results of the work conducted, providing a thorough explanation and exploration of the results without the fleet manager. It includes the results of the fleet of five robots when a fleet manager was introduced. Additionally, it details the simulations performed in a real robot with the autonomous navigation algorithm chosen for this work.

Chapter 5 presents the conclusions of the work elaborated and the goals accomplished and proposes ideas for future development in the theme of this thesis.

Chapter 2

State of the Art

Chapter 2 focuses on a general overview of the two different types of robots that this work explores: the AGVs and the AMRs. The technologies of these two vehicles are thoroughly explored in sections 2.1 and 2.2.

A revision of the graph search algorithms mainly used by AGVs and specifically the TEA* algorithm used for this work is made in section 2.4, and in section 2.5, some methods used for obstacle avoidance by AMRs are approached.

An analysis of Multi-robot systems is made in section 2.3 and in section 2.6 fleet management systems are covered and some industrial solutions of fleet managers are referenced.

Finally, in 2.7, a contextualization of the simulation environment and tools is made.

Automated Guided Vehicles and Autonomous Mobile Robots are widely used to help with problems related to logistics, manufacturing, and high-risk labor. A rise in these vehicles was seen particularly during and after the COVID-19 pandemic, in which there was a need to reduce the risk of viral infection transmission in freight transport: these systems were able to substitute people in jobs that required them to transport items in semi-enclosed spaces, such as factories, warehouses, and workshops. [3]

AGVs follow a predefined trajectory, rely on tracks, and usually require operator supervision. These vehicles can perform tasks, such as transporting goods to designated locations, and are broadly used in warehouses and industrial environments. AGVs serve three primary purposes: materials handling, replacing manual labor, and introducing a mobile workbench [4]. The application of these vehicles has the advantages of reducing labor and accidents in the industrial setting, monitoring materials, and reducing loss rate.

On the other hand, AMRs typically navigate using sensors and computer vision to complete tasks without predefined routes. AMRs are mobile robots with the autonomy to make an accurate and timely response to environmental changes and can navigate freely from one place to another to complete tasks. AMRs are typically referred to as a modernization and development of the AGVs. They are typically used in factories, in healthcare, to assist disabled patients, and even in households, such as cleaning robots [4].

The recent COVID-19 pandemic resulted in a shortage of people doing in-person work and, thus, in the need to automate many processes. Due to the demand for the continuously increasing labor production, significant advancements were made in the AGV and AMR industries.

By 2020, JD.com delivered 90% of its retail orders on the same day or the day after the order was placed by using AGVs, referred to as storage rack-moving robots. This constituted a big advantage during the demand that arose with COVID-19, and has led to the annual saving of hundreds of millions of dollars, since the agility of these vehicles allowed the company to handle about ten times the normal volume of orders. [5]

Despite being a good advancement in process automation, it is important to note that AGVs have their shortcomings, as a study by mk Technology Group on the main differences between AGVs and conveyor belts determined: AGVs are, in fact, useful, but in limited applications, such as for long distances and small throughput, since the speed in which the robots analyzed in [6] run is low - 0.30 m/s on average.

A comparison matrix for AGV variants is shown below in Fig. 2.1.

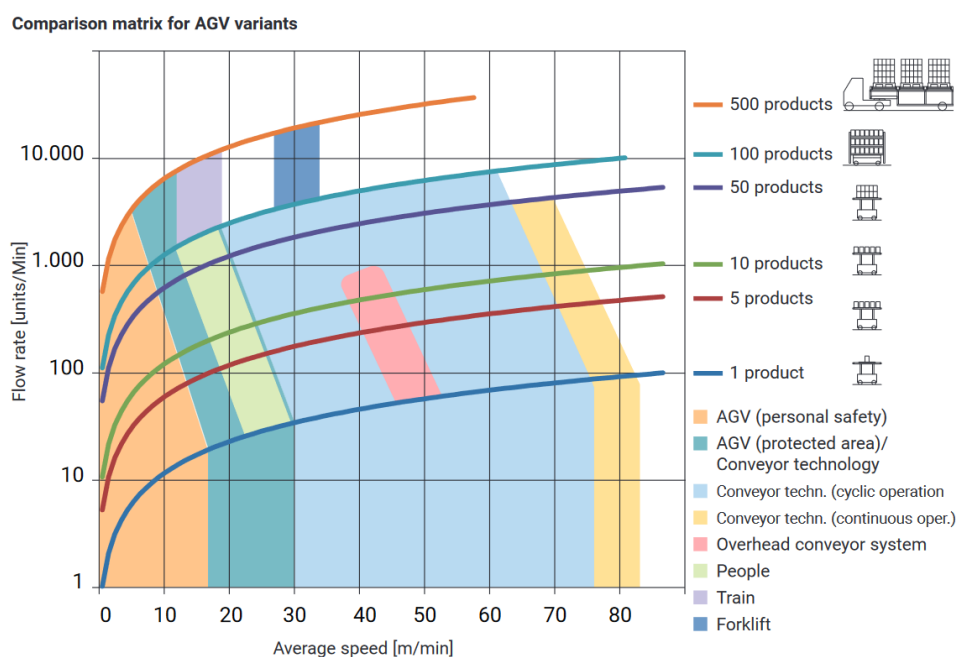


Figure 2.1: Comparison Matrix for different transportation systems [6]

Although limited in terms of velocity, these systems are key to integration in dynamic environments compared to stationary solutions.

The challenge of ensuring that AGVs move accurately is constantly being reviewed and improved. In the following study, a method for guiding AGVs was considered, in which a virtual route is created so that, using its current position given by position sensors, the AGV can travel autonomously to maintain the travel line better. [7]

Regarding Position and Trajectory Planning, there has been considerable focus on developing algorithms that help AGVs and AMRs plan optimal trajectories, such as Dijkstra's, A*, D*, and

other generic algorithms, considering efficiency and adaptability to dynamic environments. [4]

Studies have been conducted to explore various path-planning techniques such as Fuzzy Logic, Neural Networks, Neuro-Fuzzy systems, Genetic Algorithms, Particle Swarm Optimization, Ant Colony Optimization, and Artificial Immune Systems. Neuro-Fuzzy systems were the most effective in mobile robot navigation tasks, such as target-reaching and obstacle-avoidance, especially in unknown and dynamic environments. [8]

Efforts have been made to make AGVs efficient without the involvement of an onboard operator that considers optimizing control and traffic management by addressing conflict and deadlock issues that cause a decrease in system efficiency. Studies have been conducted on multi-load AGVs, which can pick up multiple loads at each station and items in multiple stations. [9]

Although AGVs were a great advancement in their times, AMRs have shown to be more flexible in terms of human interaction, being able to share space with workers and other tools and obstacles, without preprogrammed trajectories and boundaries, which constitutes a safer work environment compared to common AGVs. [10]

Studies to implement guidelines for designing outdoor AMRs were also made due to the growing need for more flexible and intelligent AMRs in outdoor environments. Using a systematic approach called Axiomatic Design, it is possible to develop a step-by-step guide outlining how to design and implement these outdoor AMRs effectively. [11]

Predicting the movement of human workers in a shared work environment with AMRs is also needed because, although AMRs are extremely flexible in handling logistics tasks, without predictive behavior, they move extremely slowly for safety reasons.

A study was conducted to combat this problem by comparing A* and D* algorithms. It was concluded that the AMR navigation stack must be changed, and dynamic obstacles must be considered. It was also assessed how human behavior could be predicted, using different approaches such as Physics-based approaches, which allow for short predictions, due to the conservation of momentum; Pattern-based approaches, black-box models that predict trajectories with the use of machine learning methods; and Planning-based approaches, which take into account general human behavior, for example by choosing the shortest route. [12]

2.1 AGV Technology and Navigation

An automated guided vehicle (AGV) is a mobile robot that follows a pre-defined trajectory [13]. They are often used in industrial applications to transport heavy materials around large industrial buildings, such as factories or warehouses.

Lower-cost AGVs are often called Automated Guided Carts (AGCs), pictured in 2.2, and are usually guided by magnetic tape. These vehicles are smaller and, therefore, used for transporting lightweight materials. They can be manually loaded and unloaded and are usually guided by tracks, wires, or sensors.

They are normally used in less industrial environments where manual transport can be replaced to transport smaller loads AGVs. These robots can have lasers to help with navigation and, if so, are called LGVs (Laser Guided Vehicles).



Figure 2.2: Automated Guided Cart. [14]

Tugger AGVs are designed to tow materials and are used for the transportation of materials. They navigate autonomously along predetermined paths, often using magnetic tape, laser guidance, or vision systems. Due to the fact that they can transport heavy loads over long distances, they are often used in warehouses, factories, and distribution centers for moving bulk materials or products from one location to another.

Forklift AGVs, represented in Fig. 2.3, are automated versions of traditional forklifts. They can lift and transport materials vertically and horizontally, equipped with forks or other lifting mechanisms.



Figure 2.3: Forklift AGV. [15]

Assembly line AGVs, pictured in Fig. 2.4, are designed to transport materials and components along a manufacturing or assembly line. They can be synchronized to match the assembling pace. They are common in industrial environments where several materials need to be delivered to various workstations.



Figure 2.4: Assembly Line AGV. [16]

2.1.1 Navigation Types

There are several navigation types for AGVs, some of which can be seen in Fig. 2.5. One navigation type is Wire Navigation, in which a wire is placed an inch below ground, and the AGV is equipped with a sensor on the bottom close to the ground. [17]

Another navigation type is the use of Guide Tape, which works the same way as Wire Navigation. A significant advantage of this method is that the path can be easily relocated. In this method, the tape can be colored or magnetic. Colored tape is less expensive, but has the disadvantage of becoming damaged or dirty when embedded in high-traffic areas. On the other hand, magnetic tape can be embedded in the floor like wire, however, it may get unpowered or passive, because of structural issues.

There is a different type of navigation called Laser Target Navigation, in which reflective tape is mounted on walls, poles, and fixed machines. In this navigation type, the AGV carries a sensor, which works both as a laser transmitter and receiver. [17] The angle and the distance to reflectors in line of sight and range are automatically calculated, and this information is compared to the map of the reflector layout stored in the AGV's memory. The current position of AGV is estimated by triangulation and compared to the path programmed into the reflector layout map. The steering is adjusted accordingly to keep the AGV on track, constantly updating the position.

The AGV can also follow a path using transponders and a gyroscope. This type of navigation is called Inertial Navigation or Gyroscopical Navigation and can be useful for operating in tight aisles and extreme temperatures. It can also be paired with the Guide Magnetic Tape Navigation.

Additionally, Vision Guidance is an important Navigation Type, where cameras are used to record features along the route with computer vision, allowing the AGV to replay the route by using the recorded features to navigate. This method uses 360° images to build a 3D map. [17]

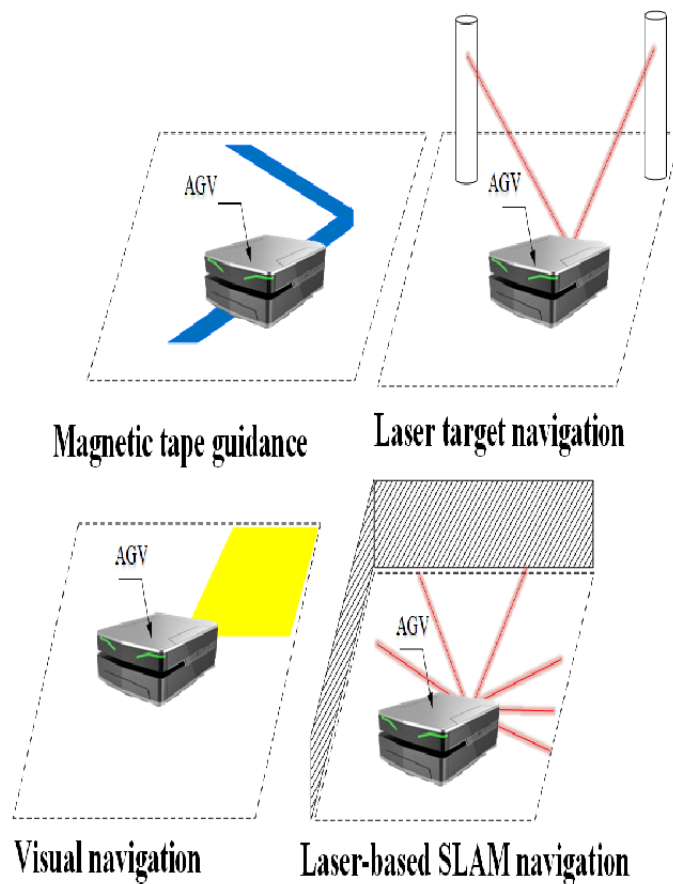


Figure 2.5: Some AGV navigation types. [18]

Finally, Evidence Grid Technology uses probabilities of occupancy for each point in space to compensate for the uncertainty in the sensors' performance and the environment.

2.1.2 Localization Systems

Some AGV localization systems use anchoring points, which constitute physical guides that allow free navigation. In this system, a grid of permanent magnets or transponders is set on the floor to determine the location of the AGV and magnetic proximity sensors are mounted on the bottom of the AGV. The path that the robot follows is preprogrammed. [19]

Additionally, a very common method of localization in AGVs is a Sensor-based localization system. This system uses ultrasonic and infrared sensors, emitting, respectively, sound waves and infrared signals, in order to measure the distances to objects and walls in the map, therefore helping the vehicle navigate and avoid obstacles this way.

Advantages to this method constitute the high reliability for short-range detection - which can turn to a disadvantage in certain environments where a bigger range is needed. [20]

It is also worth mentioning the possibility of using the Global Positioning System (GPS). This system provides location data when receiving satellite signals and is often combined with other localization methods for more accuracy. This method has some well-known advantages, such as having global coverage, providing flexible and precise navigation, and being useful for outdoor environments. However, GPS may be limited in indoor environments, due to signal blockage. [21]

Furthermore, a good localization method to combine with the GPS is Inertial Navigation Systems (INS), which use Inertial Navigation Units (INU), accelerometers and gyroscopes, to track the AGV's general movement based on its previous position and in relation to the ground. INS do not rely on external signals and are very robust systems. Despite this, some disadvantages of these systems are that they can accumulate errors over time and require calibration. [22]

LiDAR systems use laser beams to analyze the environment and create high-resolution maps. AGVs equipped with LiDAR technology compare live scans with their stored maps to accurately determine their position. [23]

Some advantages of these systems are their high accuracy and high effectiveness in dynamic environments. However, some disadvantages can be the high cost of these systems and the fact that if the environment has reflective surfaces and depending on the type of sensors, the functionality of the LiDAR technology can be affected. [23]

Camera-based Localization is also important as AGVs equipped with cameras are able to take pictures of their surroundings. The visual odometry technique analyses these images to estimate the vehicle's movement and position relative to recognized points. [24]

Some advantages of this localization method include the reliability of its rich environmental data and its effectiveness in object detection and avoidance. This theme will be explored throughout this work. The fact that it can be affected by environmental conditions, such as weather or lighting conditions, may constitute a disadvantage in some work environments where the vehicles are not maintained often, if the camera state is not checked regularly. [25]

Feature extraction is another localization method studied for AGVs, where these vehicles rely on sensors to extract relevant environmental features. These features can be categorized as semantic, e.g., road signs, lane markings, which are often conjugated with camera-based localization systems; or non-semantic, e.g., edges, corners, which can be detected with a sensor-based localization system. Deep learning methods are also commonly used for feature extraction. [26]

Moreover, Radio Frequency Identification (RFID) tags can be placed in specific locations in the environment so that RFID readers installed on the AGVs can be able to detect these tags, hence determining the vehicle's position. This localization method provides accurate position for a low cost, and it is known to be an easy method to install in most workspaces and areas. [27]

Furthermore, Hybrid localization systems can also be used. AGVs can combine several localization systems at the same time, in order to enhance reliability and accuracy, so that the advantages of a certain localization method can counteract the disadvantages of another localization method. Examples of this are the implementation of different kinds of sensors and using localization methods while deploying methods of machine learning to enhance precision. [28, 29]

2.1.3 Locomotion Types

Types of locomotion in AGVs are divided into Wheeled AGVs, Legged AGVs, Tracked AGVs.

Wheeled AGVs have high stability and mobility and are joint due to their simplicity and minimum friction loss. These vehicles use configurations like differential drive, omnidirectional wheels, and four-wheel steering. These configurations are discussed in [30]’s work, which examines precise control methods for AGVs using sensor fusion and advanced control algorithms.

To improve the movement flexibility of this vehicle when compared to a conventional wheeled AGV, omnidirectional motility is an interesting concept to implement, which gives the vehicle the possibility to drive in any direction in a horizontal plane. In 2.6, it is possible to see a configuration of an omnidirectional wheeled AGV that uses mecanum wheels to achieve this structure. [31]

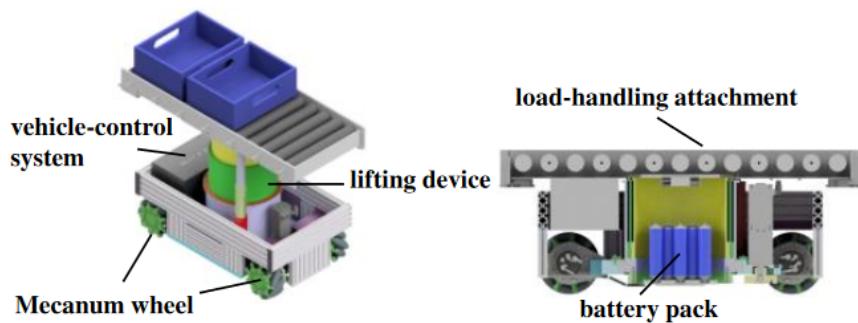


Figure 2.6: Overview of an omnidirectional wheeled AGV. [31]

Figure 2.7 shows some standard configurations of different types of wheeled AGVs, namely the differential AGV, with drive wheels in a) that provide traction but not change of direction; the tricycle AGV with steer wheels in b) that don’t provide traction but give the ability of changing direction; the quad AGV with both steer and drive wheels, also called motor-wheels in c) that provide both traction and change of direction; and in d) the omnidirectional AGV, with omniwheels, which give traction and change of direction including lateral movement [32].

Tracked AGVs use continuous tracks instead of wheels, making them suitable for both uneven and soft terrain. In [33], a wheel-rail composite chassis is proposed designed to be able to better the vehicle’s ability to work under complex conditions. This is because a single-tracked system has more ground contact area, making it a more stable module with better obstacle-climbing performance when compared to a wheel mechanism.

Legged robots are usually preferred in environments with irregular surfaces, even though they are less used in industrial applications because the legged AGV’s weight is limited. A leg comprises several links and joints, and the rise in actuators requires more power and better control systems.

Hybrid robots, represented in fig. 2.8 are vehicles that can combine any of the above configurations, where wheeled (W), legged (L) and tracked (T) can be combined in the following manners:

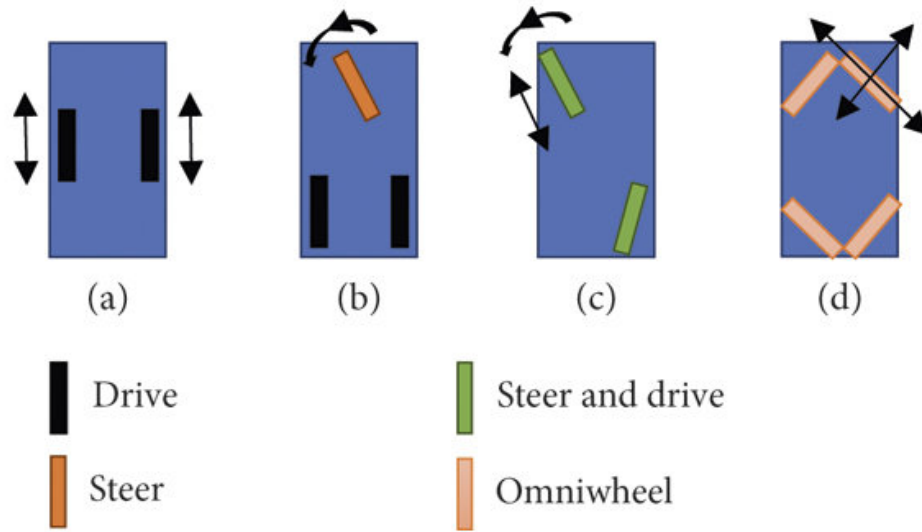


Figure 2.7: Overview of an omnidirectional wheeled AGV. [32]

LW, LT, WT, LWT. This is exceptionally beneficial for vehicles that need to face different environments and surfaces and also for better performance. In [34], an analysis of the different types of hybrid vehicles is made and the frameworks for different strategies of trajectories are studied, evaluating both speed and obstacle-avoidance.

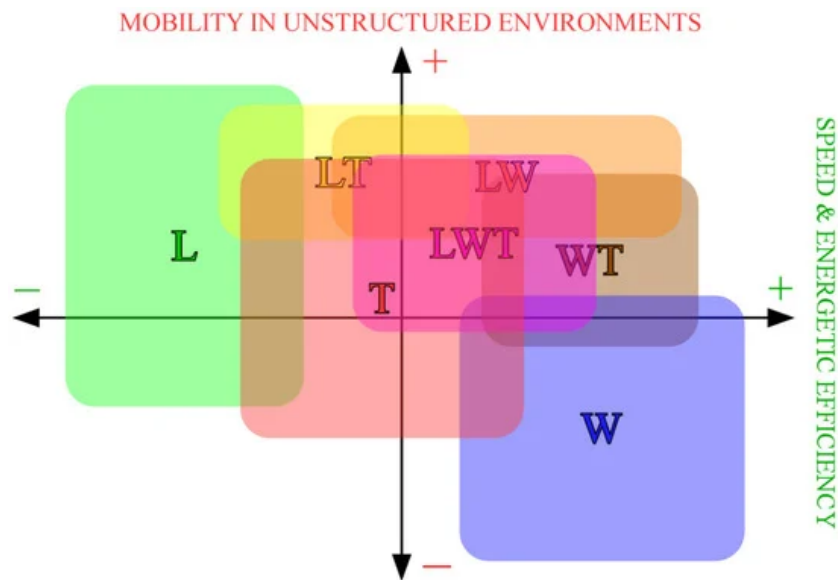


Figure 2.8: Analysis of various Hybrid Locomotion vehicles: Mobility in unstructured environments compared to speed and energetic efficiency. [34]

In Fig. 2.9 it is possible to observe an hybrid wheel-tracked AGV that is able to move through a flight of stairs. In this article, it is possible to see the benefits of an hybrid system, as the vehicle can walk through even and uneven surfaces, flights of stairs and even obstacles of complex

geometry [35]. This proves to be helpful in some environments and elevates the flexibility of these vehicles.

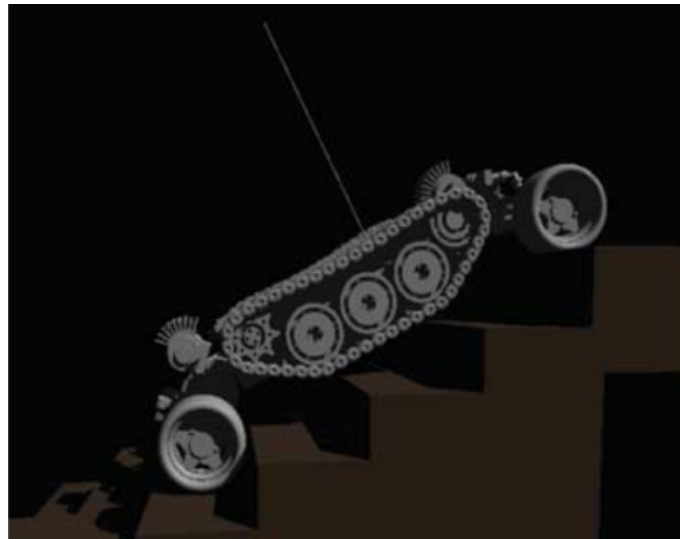


Figure 2.9: Hybrid Wheel-Tracked AGV motion in a flight of stairs [35]

An Aerial AGVs, also referred to as Unmanned Aerial Vehicle (UAV), is a pilotless aircraft, which uses radio remote control technology. These UAVs can be divided into fixed-wing UAVs, rotary-wing UAVs and hybrid UAVs (combining the two referenced vehicles). These vehicles, when inserted into the Guided Navigation category, are controlled remotely by human operators. [36]

They can also be divided into two types of vehicles: Lighter Than Air (LTA) and Heavier Than Air (HTA). LTAs are balloons and limps, the first being limited to elevation control, while the second has propellers to move laterally.

HTAs are gliders, planes, rotorcraft, and drones. Gliders and aircraft have wings, and rotorcrafts and drones use rotary blades or propellers to help with lateral and lift movement. Rotorcrafts can be easily controlled and are used daily on photography, inspection, and navigation applications.



(a) Wheeled AGV [37].



(b) Legged AGV [3].



(c) Supply delivery by an UAV, an aerial AGV [3].

Figure 2.10: Different types of AGVs.

Submersible/Aquatic AGVs, also known as Autonomous Underwater Vehicles (AUVs), typically resemble blimps that float using buoyant force and rotor blades that propel laterally. These

AGVs are used in scientific applications such as seafloor mapping, environmental monitoring, as well as industrial applications such as pipeline and cable inspections. Finally, Roll-handling AGVs carry rolls of materials like paper, plastic, and steel coils.

2.2 AMR Technology and Navigation

AMR and AGV technologies function similarly. However, AMRs have added features, such as smart navigation based on easily uploaded and updated maps, obstacle detection using sensors and machine vision, and emergency rerouting that avoids forbidden zones and reduces supply chain disruptions. AMRs are also highly and quickly adaptable and very useful for complex logistics operations. [13]



Figure 2.11: Autonomous Mobile Robot in a warehouse setting. [38]

The AMR technology is heavily supported by its Sensor Technology and they differentiate themselves from AGVs by being able to independently orient themselves in the environment without the need for external guidelines. [39] Several types of sensors are used in AMR Robots to gather data about the robot's surroundings, such as LiDAR, cameras, ultrasonic sensors, infrared sensors, etc. [38]

2.2.1 Localization Systems

Similarly to AGVs, the localization systems of AMRs do not vary much, although these vehicles exhibit greater flexibility and efficiency due to their advanced navigation features.

In [40], a study about the transition from large-scale navigation to docking precision explores a localization method named Particle Filter-based Large-scale Localization. This method is composed of a particle filter, where a set of samples - or particles - are statistically analyzed to predict and estimate the distribution of posterior locations of the robot, based on sensor data and movement commands.

Consequently, when approaching the docking area, the AMR uses Scan-Matching-based methods already mentioned in the AGV Localization System techniques, which consist of comparing current sensor scans with a pre-existing map in order to determine the robot's position and orientation for accurate docking. [40]

The Fuzzy Adaptive Extended Information Filtering (FAEIF) Scheme is studied in [41] by combining fuzzy logic with a variant of the Kalman filter. Fuzzy Logic is normally used to adaptively adjust the filter parameters based on the noise levels in the environment and is effective in handling imprecise information and making decisions based on a set of rules. Extended Information Filtering helps in tracking both the position and the orientation of the AMR with the help of Ultrasonic Sensors and considers the nonlinearities of the sensor model to improve the robot's trajectory.

Additionally, various methods of machine learning can be used to improve the AMR localization technology, as can be seen on [42] where several techniques were used, such as Bayesian Networks, Multilayer Perceptron (MLP), K-Nearest Neighbors (KNN), Support Vector Machines (SVM) and Optimum-Path Forest (OPF).

Navigation systems based on differential position measurement with GPS or GLONASS locator provide trajectory direction and can be used throughout all of the Earth's surface. Disadvantages of these navigation types is their failure underground and not providing orientation of the vehicle. [43, 44]

True gyroscopes are also important to note, as these systems are widely used in ships and submarines. However, as disadvantages, these systems are expensive and have large drift - the standard deviation of the bias. [43, 44]

Additionally, AMRs use low-cost two-axis electronic compasses. These systems are based on Hall or micro fluxgate sensors. Their main disadvantage is that they should be kept in a horizontal position, which is often impossible. [43]

2.2.2 Locomotion Types

Some applications of AMR are patient care, cleaning in healthcare environments, material handling, inventory management, quality control, and warehouse logistics. AMR has very similar locomotion types to AGV, with their navigation systems and types being differentiating points.

These locomotion types can be divided into Wheeled AMRs, Tracked AMRs and Legged AMRs for terrestrial vehicles. In [45], different types of locomotion modes are thoroughly analyzed, particularly of hybrid track/wheel-legged robots. The preferred types of robots for transportation of payloads are wheeled or tracked robots, although tracked are preferred in comparison to wheeled due to operational robustness [46]. Although these might be better for even terrains, for rough terrains such as steps and stairs, legged AMRs are optimal. Despite consuming more energy than wheeled or tracked vehicles, they ensure locomotion in all kinds of environments. [45]

AMRs can also be Aerial robots, such as UAVs (Unmanned Aerial Vehicles), which are AMRs, particularly drones, which move products to distribution centers with the help of RFID-scanning technology, and are pre-programmed to execute flights autonomously. [36]

2.3 Multi-robot systems

Joining robots to work in teams is ideally seen as a better prospect rather than working alone due to the complexity of the tasks they are expected to perform and the possibility of better performance when coordinating resources more efficiently and reliably.

According to [47], Multi-robot systems (MRS) are groups of robots programmed to perform specific tasks or behaviors collaboratively. In these types of systems, task completion time is reduced greatly, and there is a broader ability to perform activities that can't be fulfilled by one robot, even if it is bigger and more advanced. The simplicity in design is one of the advantages mentioned in [48].

A challenging problem of MRS is Multi-robot Task Allocation, which arises when it is difficult to assign each robot a set of tasks that optimizes the system's performance in a given environment and happens mostly when the robots have different capabilities and are different from each other. [48]

Another subject to tackle in MRS is collisions with collision avoidance algorithms, essentially for navigation and task completion with success.[49]

There are two methods in collision avoidance algorithms: the centralized method and the decentralized method, shown in Fig. 2.12.

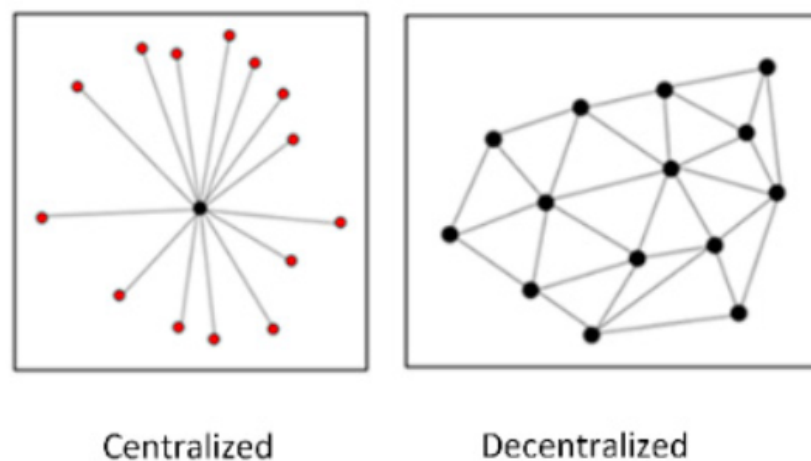


Figure 2.12: Different types of methods of MRS (Adapted from [50])

The centralized planning method relies on a centralized server that has the information of all robot positions through a global database [51]. In this system, a main control unit allocates tasks and resources to each robot in order for them to complete the objective mission. This method has

a few issues, such as the lack of scalability due to the complex computational power it demands from one centralized unit and the lack of reliability that the main unit provides in case of failure [49].

On the other hand, an advantage of centralized methods is that the main unit can produce optimal global plans while having a global overview of the environment [52]. For this reason, this method is to be used only in non-complex environments with few vehicles.

In decentralized methods, the system doesn't have a unique main control system but is composed of larger sensor networks where the vehicles that compose the system make individual and independent decisions [53]. In these systems, the robots that compose the system exchange information and process real-time data, which they use to path-plan effectively.

This method is more reliable, firstly due to the fact that it continues to work even in the event of a failure in one robot and also because it allows for more informed decision-making on behalf of the vehicles that compose the system. There is also room for more scalability and flexibility in a dynamic environment [49]

Hybrid methods that combine both centralized and decentralized methods involve a hierarchical structure where high-level planning is centralized, but low-level execution and adaptation are decentralized, and they aim make use of the advantages of both types of methods.

In [54]'s work, a hybrid decentralized task assignment approach in order to coordinate a MRS is presented, which performs better when compared to existing approaches.

2.3.1 Heterogeneous and Homogeneous Fleets

Homogeneous fleets are identical in terms of their design, functionality, and capabilities, and are usually composed of only one type of robot, may it be the AGV or the AMR.

Heterogeneous fleets mix both AMRs or AGVs, potentially varying in size and functionality. These fleets are suited for environments with diverse tasks and varying operational needs and pose several difficulties in coordination and fleet management.

In [55]'s work, a study was made by applying a modified cluster first-route second algorithm to a real-life heterogeneous fleet vehicle routing problem for a retail construction material store. In order to resolve the problem that is posed, the goal is to minimize the transportation costs with five different types of vehicles.

A study related to Deep-Reinforcement Learning on heterogeneous fleets of vehicles was also made in [56] where a model was proposed to fix the Capacitated Multi-Vehicle Routing Problem (CMVRP) with a limited number of vehicles.

In [57], the effect of a fleet that operates with different types of robots in a simulated smart warehouse is explored using a state-of-the-art algorithm. The conclusions of this work are that it is more beneficial to focus on the development of advanced management algorithms and in the management of the vehicles rather than investing in a heterogeneous fleet in a smart warehouse.

2.4 Graph Search Algorithms

There are a few search algorithms to help determine the best path for a vehicle to follow in the context of path-planning guided navigation. Two that will be mentioned are the A* algorithm and the Time Enhanced A* algorithm, an algorithm based on the A* algorithm, with the addition of temporal layers for optimization and the one used in this work for the AGVs.

The two algorithms analysed are heuristic, which means that they consider the cost of the edges between two nodes. This cost can be defined by the time it takes for a vehicle to traverse from one node to another or, in other cases, the distance between two nodes. The difficulty of traversing through the edge can be also taken into account [58].

2.4.1 A*

In [59], the A* algorithm was proposed, an heuristic search algorithm designed for weighted graphs.

Starting from an initial node, it constructs a tree by progressively extending paths one edge at a time until a specific condition is fulfilled. During each iteration, the algorithm decides which path to extend by evaluating the path's cost $g(n)$ from the start node to node n , and a heuristic function $h(n)$, that estimates the lowest cost from node n to the goal.

In path planning problems, common heuristic functions include the Euclidean distance and the Manhattan distance formulas. With this function $h(n)$ it is possible to calculate the path with Eq. 2.1.

$$f(s) = g(s) + h(s) \quad (2.1)$$

A simple example of this algorithm can be found in Fig. 2.13, and an example of this is that if the starting node is E and the end node is F, a possible function to calculate the path to follow associated with the costs of the edges would be 2.1 where $g(D) = 4$ and $h(D) = 11$. The path's cost $g(D)$ is the minimum cost from the start node to node D, in this case.

2.4.2 Time Enhanced A*

Regarding guided trajectories, an algorithm that accounts for collision-free routes must be considered. The TEA* algorithm, which will be explored in this section was the algorithm chosen for the work developed in this thesis when dealing with AGVs, due to the fact that it was the algorithm developed in INESC TEC and due to its good performance.

In [60], the TEA* algorithm, also known as Time Enhanced A*, was first introduced as an algorithm based on A*, where a time component was added, represented with a $k = [0, T_{max}]$, where T_{max} represents the maximum number of temporal layers. The map is obtained by cell-decomposition with some cells being identified as occupied and others as empty. Fig. 2.14 shows that the neighbouring cells of the current AGV position belong to the next temporal layer, which can be useful to predict the next position of the vehicle.

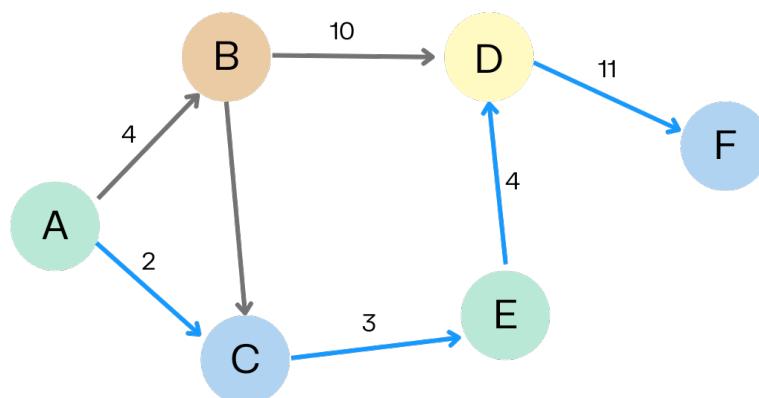


Figure 2.13: Example of a graph search algorithm.

In this case, this algorithm plans the path of each vehicle while considering the dynamic positions of other vehicles, which allows for a free route for each vehicle and is flexible to different numbers of AGVs and types of environments. The proposed algorithm represents time in temporal layers where each vehicle position is calculated. Therefore, in the initial path's calculation, present and future possible collisions could be identified and mitigated.

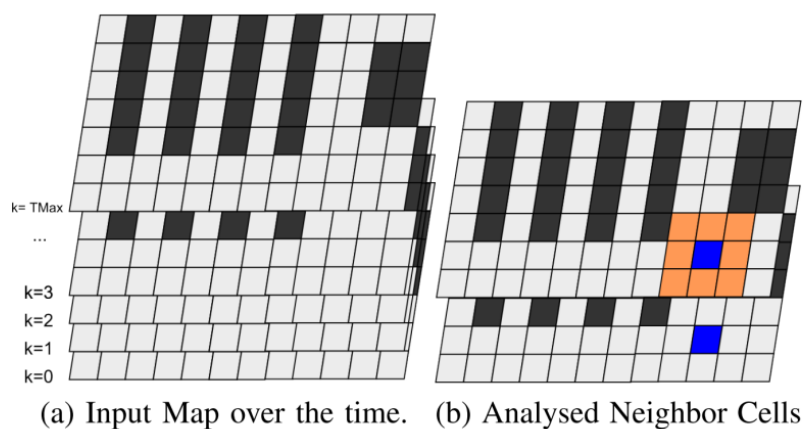


Figure 2.14: The input map and the analysed neighbor cells focusing the cell with the same position of the AGV. [60]

In a posterior phase, work was presented regarding an improvement in the TEA* algorithm, where the time spent in rotating the vehicle so it could move in different directions was accounted for. This improvement is presented in [61].

In [62], the TEA* algorithm is further explored and implemented in an environment where communication breaches occur. Therefore, some improvements are made, such as implementing a Task Scheduling function and a binary semaphore to detect communication faults.

In [63], there was an approach to this algorithm to ensure there would always be a path for a certain robot by implementing techniques such as Robot Priority Swapping, where the priority of

a higher-priority robot is swapped when its path blocks any possible path of a lower-priority robot and Moving Idle robots, which consists in moving an idle robot that is blocking the passage of a higher-priority robot by planning a path for it.

2.5 Methods for Obstacle Avoidance

AMRs and their ability to move through the environment independently and safely involves avoiding interference during their trajectory, primarily in dynamic environments. Therefore, obstacle avoidance is a key aspect in the development of AMRs and in the study of these vehicles' fleets.

The detection and circumvention of obstacles in the robot's path is essential to prevent collisions. This is critical not only to ensure proper operation and task completion but also to the safety of human operators who can be present in the work environment and constitute dynamic obstacles.

AMRs are used often to increase efficiency in industry processes and effective obstacle avoidance is valuable so robots can move through the environment without stops or detours, similarly to what happens to AGVs. By deflecting from obstacles, they optimize route planning and reduce travel times, which is a key advantage of AMRs relative to AGV fleets, due to the increase of productivity and decline of operating costs.

2.5.1 Reactive Methods

Reactive methods for obstacle avoidance in AMRs are techniques where the robot responds to obstacles in its immediate environment based on real-time sensor data. These methods involve simple path-calculations with the use of algorithms that rely on immediate reactions to sensory inputs.

The Artificial Potential Field (APF) is a method that uses a cost function, by which obstacles are considered negative forces and goals positive forces. The points that generate attractive force attract AMRs. The obstacles that produce repulsive force drive the AMR away. [64]

Figure 2.15 shows a graph with the three-dimensional visualization of how the APF method works, with the positive values being considered targets or goals and the obstacle generating negative or repulsive potentials.

The robot then navigates by following the resultant force vector calculated by this vector, which guides it toward the goal while avoiding obstacles. This method uses a simple math model, thus being easy to implement and being able to show a real-time response. [66]

A disadvantage of this method is that it is not optimized when the function reaches a local minimum, when the forces balance out. This issue is tackled in [66], where a modified version of the APF algorithm is proposed.

The Vector Field Histogram (VFH) method is a goal-oriented obstacle avoidance method, where a polar histogram is used to represent the local environment around the robot. The density of the obstacles is taken into account through a cost function that the robot uses to choose the path with minimal obstacle density in order to navigate through and towards the end goal. [67]

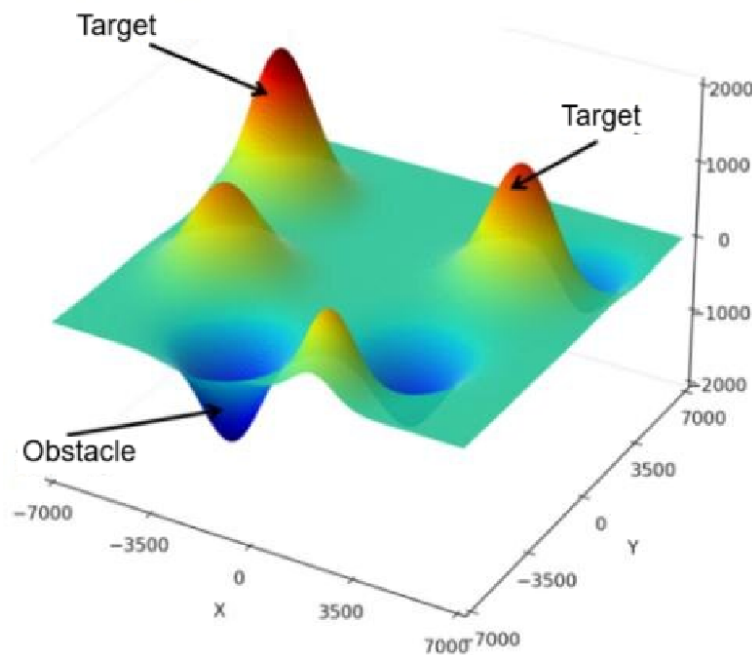


Figure 2.15: Artificial Potential Field. (Adapted from [65])

In [67], a new method is introduced widely based on the VFH, named VFH*. This new method is an adaptation of the A* algorithm associated with the VFH, where a global path point is generated by the A* as a temporary target point and the path is optimized using the A* search tree in order to improve the VFH method. This is because a big disadvantage that the VFH displays is the difficulty in navigating through narrow passages.

2.5.2 Natural Computing Algorithms

Natural Computing Algorithms is a field that integrates processes found in nature into computational methods, such as DNA and molecular computing. These algorithms are modeled after biological and physical phenomena and can be applied to various problems, specifically in optimizing path planning. [68]

These algorithms are good at finding global optimal solutions and are able to produce multiple alternative solutions, by handling complex and high-dimensional search spaces.

2.5.2.1 Ant Colony Optimization (ACO)

This algorithm is based on the Double Bridge experiment [69] and is depicted in fig. 2.16.

Ants use a chemical communication system to optimize foraging routes. As they travel, ants leave pheromone trails that attract other colony members. The more ants that follow a particular path, the stronger the pheromone signal becomes, creating a positive feedback loop.

In the Double Bridge experiment, ants were presented with two paths of different lengths to reach a food source. Initially, ants randomly chose either path. However, the shorter route allowed

ants to make more trips in less time, resulting in a more rapid accumulation of pheromones. This stronger chemical trail on the shorter path influenced subsequent ants to preferentially select that route.

Over time, this self-reinforcing process led the majority of the colony to converge on the optimal (shortest) path to the food. The experiment elegantly demonstrated how ant colonies produced collective problem-solving without centralized control or global knowledge of the environment. [69]

ACO algorithms are inspired by the foraging behavior of ant colonies. In path planning applications, virtual "ants" traverse the environment and deposit virtual pheromones to mark good paths. This allows the algorithm to converge on optimal or near-optimal paths over time by using swarm intelligence. [70]

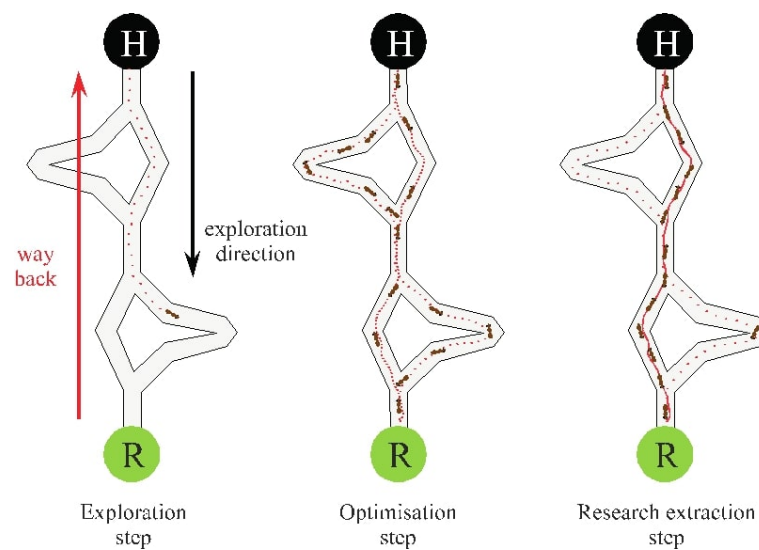


Figure 2.16: Ant Colony Algorithm. (Adapted from [71])

ACO has the advantages of having robust performance and good stability. In spite of this, it shows some disadvantages such as slow and premature convergence and solution inaccuracy when dealing with a large data set. [72]

2.5.2.2 Genetic Algorithm

In order to address the challenges of path-planning, recent progress in the world of robotics and artificial intelligence has motivated the use of contemporary optimization techniques, such as Genetic Algorithms (GAs).

A Genetic Algorithm is a search and optimization method inspired in natural selection and genetics to simulate evolution in nature through coding [73]. The principles of these algorithms were first published in [74].

A randomly generated set of candidate solutions or individuals constitutes a population. Each of these solutions is represented by a binary string of genes. Individuals are ranked in a fitness function by how well they solve a given problem. [73]

GA focuses on three probabilistic scenarios, them being the reproduction, the crossover and the mutation. Firstly, using reproduction and crossover, selected individuals are paired with the goal of producing a new generation and improving the fitness function. Afterwards, probabilistically, arbitrary changes are introduced to the offspring's genes and mutations appear, which maintain genetic diversity and motivate the analysis of new solutions. Finally, these solutions are evaluated using a bias where better chromosome solutions are more likely to be selected. [73]

New generations of individuals are evaluated using the fitness function and these processes are repeated iteratively from the selection step until a solution is found, concretely when the criteria for convergence is met. [73]

In [75], a modified crossover Genetic Algorithm is implemented in order to plan collision-free paths with static obstacles for differential drive mobile robots.

2.6 Fleet Managers in Path-planning Navigation

An AGV Fleet Manager is a comprehensive system designed to coordinate, supervise, and control the operations of multiple AGVs within an industrial or warehouse environment. [63]

This system ensures the efficient movement, task allocation, and navigation of AGVs, optimizing the productivity and the efficiency of task completion. By managing the interactions and routes of the AGVs, the fleet manager enhances overall operational efficiency, reduces downtime, and maintains smooth logistical operations in the industrial or warehouse setting.

The Fleet Manager used in this work is based on the TEA* algorithm [63] to ensure graph management, by using features like task allocation and scheduling, path planning and collision avoidance techniques.

An AMR Fleet Manager is a software platform designed to manage and optimize the operations of multiple Autonomous Mobile Robots. Some key components of this system is centralized control, real-time monitoring, task assignment and predictive maintenance, to ensure all robots' position and path is being tracked at all times. [76]

In [76], a fleet size selection mechanism is developed in order to improve task allocation. By defining Key Performance Indicators like constraint satisfaction and utilization rates in a decentralized Multi-robot task allocation problem using machine learning.

2.6.1 Industrial Solutions

OMRON Portugal

OMRON Portugal [77] presents a fleet manager software to the market, called OMRON FLOW Core 3.0 Fleet Manager. This software prioritizes efficiency by promising smart task attribution, movement management and traffic control at the same time as maximizing activity time, by giving automatic updates throughout the fleet and regulating the battery power of the fleet and rearranging paths, taking into account anchoring needs.

OMRON FLOW Core 3.0 provides selectable autonomy, by choosing when AMRs behave partially like an Autonomous Guided Vehicle or when using autonomous driving. It also tracks dynamic obstacles such as humans, objects like forklifts, or other moving vehicles. By predicting and avoiding collisions and moving around obstacles' paths, the software has better decision-making and path-planning.

In April of 2024, the new version of this software, OMRON FLOW Core 4.0, was released, where the fleet resources can be managed and dimensioned to the client's needs, with bigger flexibility and mistake tolerance for continuous operation. Additionally, this version supports 3D LiDAR for all AMR models, and anchoring strategies were improved to better the path-planning in regards to battery and power levels. [78]

MiR - Mobile Industrial Robots

MiR [79] - Mobile Industrial Robots - is an industry that focuses on developing AMRs and also has released its own software into the market, MiR Software which is divided into two main categories: MiR Fleet, their centralized fleet control and management software and MiR Insights, a software that optimizes the robot fleet path-planning by monitoring the data provided.

MiR Software allows for control of MiR Robots (AMRs) may it be individually or in large fleets. It allows for rapid rerouting of paths, fast switch of maps, and its good user interface grants optimized driving patterns and avoidance of bottlenecks.

DENSO, a MiR client, states that MiR Fleet is intuitive when compared to other platforms and highlight the ease of mapping, mission creation and location changes. MiR250, an AMR developed by the company, uses REST APIs, which are Application programming interfaces that use design principles of the REST architectural style, which makes it possible to support on-time deliveries and other important features for DENSO. [80]

Another client, Faurecia, states that the transport orders are completely automated and data is generated via sensors at the loading and unloading docks, which are sent to a controller that transfers this data to MiR Fleet. The Fleet management system then coordinates and distributes the orders to the AMRs. [81]

AETHON

While the solutions presented before were in a warehouse and factory environment, AETHON [82] is a company that focuses on using AMRs for both healthcare and hospitality.

AETHON fleet management in healthcare promises deliveries of medication through the hospital, of specimens to the laboratory and of meals, linens and other needed objects in the care facility. These AMRs are camera and laser-based systems.

In hospitality it promises deliveries to guests, housekeeping support and storage management like moving carts of supplies. AETHON mentions dynamic traffic patterns, elevator use and handles robot fleets of any size, unlimited map sizes and delivery points, no matter the square footage or the number of floors.

KUKA

KUKA [83], a company that offers automation solutions in the food industry, also came out with its own fleet management software solution, called KUKA.NavigationSolution, consisting of an industrial PC that overviews the AGV system and the navigation software as well as the fleet management software. This solution relies on WLAN for communication and exchange of commands for mobile robots, while based on Industry 4.0 for the management of data.

KUKA.NavigationSolution uses AMRs which are laser and wheel-sensor based and uses the SLAM method for mapping. It responds to changes in the environment and uses virtual paths to make it possible to move the robots along pre-defined routes.

The software has high-precision positioning through relative positioning with CAD-based object recognition and tracking and fine localization of a vehicle within an environment.

The software integration relies on an Eclipse-based environment, allowing for programming in Java and the integration of third-party software. KUKA's solution is hardware-independent and can be used for different platform kinematic systems, and handles all types of vehicles including holonomic vehicles with Mecanum wheels.

2.7 ROS

ROS (Robot Operating System) is an open-source, and meta-operating framework designed to provide tools and libraries to build, write and run code on multiple computers, and services such as hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. [84]

ROS consists on a number of processes on a number of different hosts, connected in a peer-to-peer network. [85] It allows for writing code in several languages, such as C++, Python, Octave and LISP. ROS is also designed to be as thin as possible so that code written can be used for other robot software frameworks, making ROS an easy tool to integrate. [84] [85] It also includes simulation tools like RViz, which allows for testing and development in a virtual environment. RViz is the simulation tool used for the simulations made throughout this work.

2.7.1 ROS Navigation Stack

In implementing a system where an autonomous vehicle can move, it is important to note two concepts: global navigation and local navigation. Global navigation considers the environment's map and computes a trajectory accounting for the robot's geometric properties, which can become quickly outdated. In the latter, the local navigation takes into account the robot's sensor readings, which allows the robot to analyze changes in the environment while following the global plan as closely as possible, also considering the dynamic properties of the robot. [86]

Simultaneous Localization and Mapping (SLAM) is a method used for (autonomous) vehicles, in which the vehicle is acquiring a spacial map at the same time as it's recording its location in it [87]. This method allows the vehicle to map unknown environments and is very useful in path planning and obstacle avoidance.

There are two types of SLAM approaches: the sensor-dependent processing, which depends on sensors such as cameras, LiDARs to estimate obstacle position and motion; and the sensor-independent processing, which registers pose graphs and optimizes these graphs.

SLAM navigation is done with the help of the `move_base` package, (in this case). The `move_base` package implements a model where, when given a goal in a certain environment, the robot will try to reach it with a mobile base. This model supports global and local planners in helping the mobile base achieve its navigation goal.

Real-world environments usually contain obstacles, whether they are dynamic or not, which require real-time path planning. Using a local planning approach is the key to re-planning the local route when using obstacle avoidance methods. [88]

A global planner computes the optimal path with the prior mapping of the environment and static obstacles, while the local planner is essential to evaluate the dynamic obstacles and build the appropriate path in this setting. The `move_base` node supports any global planner in the `nav_core::BaseGlobalPlanner` interface. [89]

The `move_base` node also supports two costmaps, one for the global planner and one for the local planner, which are used to map the environment and help complete navigation tasks. [89]

2.8 Synthesis and Relevance to the Study

This chapter provides an overview of Automated Guided Vehicles and Autonomous Mobile Robots, which serves as an essential first step in understanding the foundation for the work that follows in this document.

It then explores multi-robot systems in order to gain a deeper understanding of the strategies for effectively managing and coordinating these systems of vehicles. A review of several graph search algorithms is presented, with particular emphasis on the TEA* algorithm, which is the method used in this work for the simulations involving guided path-following robots.

In addition, obstacle avoidance methods are studied to extend the knowledge of obstacle detection and avoidance techniques applicable to autonomous robot fleets. This study is complemented by an overview of fleet management systems, which discusses current industrial solutions available in the fleet management market, providing insight into existing technologies.

Finally, the environment in which the work will be developed is described, focusing on the Robot Operating System (ROS) and related frameworks. This overview facilitates the understanding of the implementation and testing of the proposed approaches throughout the thesis.

Chapter 3

Algorithms and Fleet Manager for Autonomous Navigation

In this chapter, algorithms for autonomous navigation are discussed, such as the Base Local Planner, the TEB Local Planner and the DWA Local Planner. There is a section where the results from this analysis are discussed. Additionally, in this chapter, there is a description of the fleet manager developed for the simulations of the autonomous navigation fleet.

A global planner is responsible for creating the best possible path from the robot's current position to its goal position with the environment and static obstacles in mind due to previous map information. [90]

Although the global planner can predict the best path based on the map conditions, a local planner is needed in order to consider dynamic obstacles or vehicle constraints. Therefore, the local planner takes the global planner and recalculates the path at a programmed rate as the vehicle navigates, whereby considering the sensor data of the vehicle. [90]

In a primary approach, some simulations were done with only one robot to focus on perfecting the robot's autonomous trajectory, by testing three different local planners and tuning their parameters to evaluate how these planners behaved in respect to path following and focusing on how well they performed in narrow passages, i.e. choke points in maps and in deflecting obstacles, may them be static or dynamic. A lot of focus was given to the move_base package and specifically to its Local Planner component. This package has both a global and a local planner plugin. In order to be used, these plugins must conform to an interface defined by the ROS node [91].

This simulation is performed using the ROS framework. The move_base ROS node is used for the simulation of AMRs. Both cost maps are displayed in Fig. 3.1a: the global cost map in light blue and the local cost map in light pink. The global planner's computed trajectory is shown in green and the local planner's is shown in orange, as presented in Fig. 3.1c.

The aim of the study developed in this section is to identify the optimal local planner for implementation in the AMRs fleet.

3.1 Base Local Planner

The Base Local Planner is a simple planner that allows one to set some parameters to compute a local trajectory. It is based on a grid map which is created using a value function created by the base planner and supports both holonomic and non-holonomic robots.

This planner is also used for robots with lower acceleration limits, which need a planner that considers the acceleration limits at every step.

In the Base Local Planner, tuneable parameters can be grouped into the following categories:

- Robot
- Goal Tolerance
- Forward Simulation

Robot

The robot parameters are tuned based on the type of robot available. In this case, the values for the velocities and accelerations were computed by observing the robot's behavior. Additionally, the robot being simulated is a holonomic robot, which is a robot that can move in any direction independently of its orientation at the time.

Therefore, the values achieved were the following:

```

max_vel_x: 0.4 m/s
min_vel_x: 0.1 m/s
max_vel_theta: 0.26 rad/s
min_vel_theta: -0.26 rad/s

acc_lim_theta: 3.0 rad/s2
acc_lim_x: 2.5 m/s2
acc_lim_y: 2.5 m/s2

holonomic_robot: true

```

Goal Tolerance

In this section, the goal tolerance is calculated to account for how far from the actual goal it is acceptable for the robot to be, as sometimes it is very difficult to reach the end goal, and the robot can attempt to reach it indefinitely, which is not sustainable.

In this case, the xy goal tolerance and the yaw goal tolerance were computed according to the needs of the simulation, and are not high enough values so the robot doesn't stop at a different location but has flexibility in its final positioning.

The parameters were set as the following:

```
yaw_goal_tolerance: 0.05 rad
xy_goal_tolerance: 0.10 m
latch_xy_goal_tolerance: false
```

Forward Simulation

To compute the simulation it's necessary to define a simulation time, which was defined as 0.8, which is a low value but enough to compute accurately the trajectory of this local planner for this robot and environment.

The value given to the parameter of the simulation time is the following:

```
sim_time: 0.8 s
```

3.2 TEB Local Planner

Another local planner was implemented, named `teb_local_planner`, which is more complex than the `base_local_planner`. This was done to see some improvements in movement, as well as to compare the planners. This planner package implements a plugin to the `base_local_planner`, with a method called Timed Elastic Band (TEB), which optimizes the robot's trajectory execution time, considering the kinodynamic constraints at runtime.

In TEB, tuneable parameters can be grouped into the following categories:

- Robot
- Goal Tolerance
- Obstacles

Robot

In this section, the parameters related specifically to the type of robot are tuned, e.g. the parameters defined by the manufacturer.

The robot footprint model serves as an approximation of the 2D contour of the robot to optimize the trajectory process. The robot footprint model constitutes a dedicated parameter instead of loading the footprint from the common `costmap_2d` parameters in TEB, because it is very useful for the distance calculations, which can be very complex, influencing greatly the computation time.

Because, in this case, the robot constitutes a complex model, the type of footprint is a closed polygon. This polygon is then defined in a list of vertices and its axis of rotation is assumed to be $[0, 0]$.

The footprint model is defined as:

```

footprint_model:
  type: polygon
  vertices: [[-0.355, -0.235], [0.355, -0.235], [0.355, 0.235], [-0.355,
    0.235]]

```

The velocity and acceleration parameters are also defined considering the robot's hardware limitations. Parameters with '_x' specify linear kinematics, whereas parameters with '_theta' specify angular kinematics.

These parameters were defined as:

```

acc_lim_x: 0.2  $m/s^2$ 
acc_lim_theta: 0.2  $rad/s^2$ 
max_vel_x: 0.5  $m/s$ 
max_vel_x_backwards: 0.5  $m/s$ 
max_vel_theta: 0.3  $rad/s$ 

```

Goal Tolerance

The Goal Tolerance is needed to define how much deviation from the actual goal is acceptable to exist.

xy_goal_tolerance is the linear distance away from the goal, in meters, which is acceptable and yaw_goal_tolerance is the acceptable deviation in the robot's orientation, in radians.

These parameters were defined as:

```

xy_goal_tolerance: 0.2 m
yaw_goal_tolerance: 0.2 rad
free_goal_vel: false

```

It was taken into account that the xy value should not be too high to allow the robot to stop at a different location other than the goal, and the yaw should not be too low so that the robot would have enough flexibility to rotate to enhance efficiency.

Obstacles

This section defines the behavior of the robot during trajectory planning. There are two parameters that were essential in the tuning of the robot, them being the min_obstacle_dist, which is the minimum desired separation from the obstacle in meters, and the inflation_dist, which is the buffer zone around obstacles with non-zero penalty costs.

If the `min_obstacle_dist`'s parameter was too high, then the robot couldn't pass between a hallway. The robot would get stuck in obstacles if it was too low. Keeping this in mind, it was then necessary to tune this parameter in combination with the `inflation_dist`, because the latter should balance the first so the robot would pass between hallways and go around the obstacles.

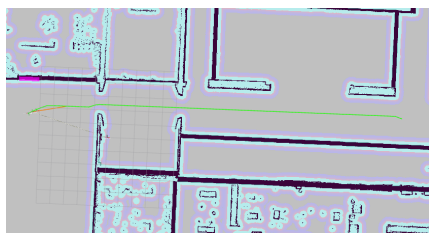
After tuning these parameters, the optimal results for the proper trajectory with obstacles were the following:

```
min_obstacle_dist: 0.2 m
include_costmap_obstacles: true
inflation_dist: 1.3 m
include_dynamic_obstacles: false
```

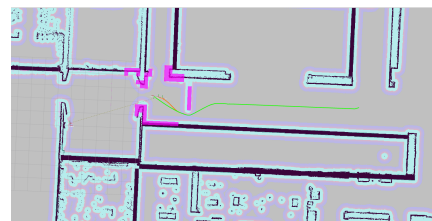
Furthermore, it was observed that the robot could follow a trajectory well without bumping into walls but had difficulty in going around obstacles and following a straight line, so the solution was to add another inflation layer.

An inflation layer serves as a visual enhancement tool, by inflating selected features. This layer adds new values around lethal obstacles to make the costmap represent the robot's configuration space.

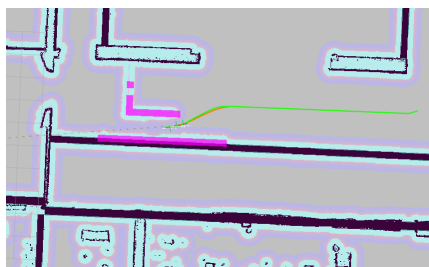
Adding the second inflation layer, which was focused on detecting the static obstacles, ensured that the path was well calculated and that the robot followed a straight line and went around obstacles without difficulty. Different obstacles detected by different sensors can be detected differently by changing parameters related to each inflation layer such as the "inflation_radius" and the "cost_sealing_factor" when initializing multiple inflation layers. In this case, both the inflation layers were defined with the same values for the relevant parameters.



(a) Robot approaching obstacle.



(b) Robot detecting obstacle.



(c) Robot deflecting obstacle.



(d) Robot approaching final goal.

Figure 3.1: Robot using TEB Local Planner to go around obstacle successfully.

In simulation, this local planner behaves as expected and deflects easily from obstacles. In the series of figures in Figure 3.1, the vehicle approaches an object while navigating freely throughout the environment. The global planner - in green - does not at first detect the obstacle, but as soon as the sensors detect it, it automatically changes, and the local planner - in orange - changes as well, guided by the information the lasers transmit to the vehicle. The vehicle then deflects the obstacle and ends the trajectory.

When posed with a new goal, in which the vehicle has to pass through the obstacle again, the global planner already knows it exists as it was already mapped in advance by the sensors of the robot and, by doing so, automatically creates a path that goes around said object. This is shown in Fig. 3.2.

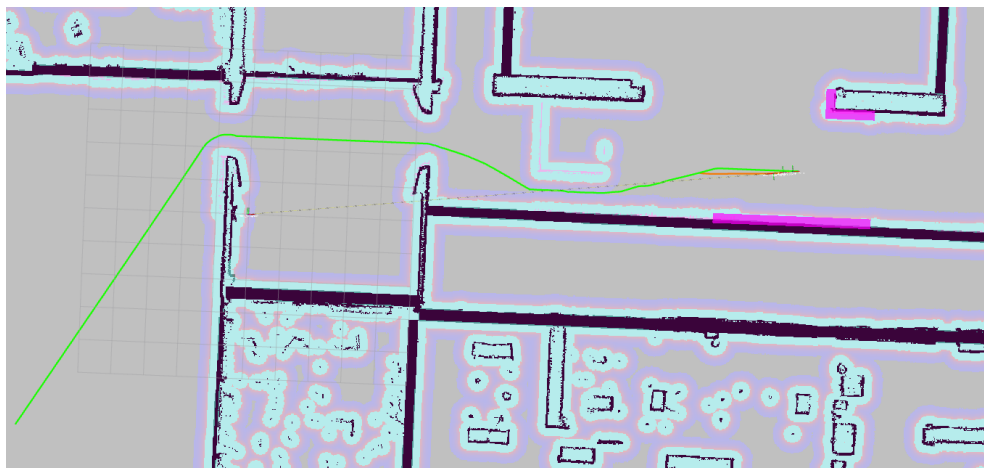


Figure 3.2: Robot passing through previously detected obstacle.

3.3 DWA Local Planner

The Dynamic Window Approach (DWA) algorithm is a collision avoidance strategy for mobile robots while incorporating the robot's dynamics. It generates control velocities to achieve a goal position while optimizing the distance to obstacles and its velocity. [86]

This model discretely samples velocity pairs in the robot's control space (dx , dy , $d\theta$). Subsequently, it uses forward modeling and simulation of the robot's current state to predict how it would behave for a period of time using the sampled velocity. It assesses each result from the forward simulation regarding proximity to obstacles, the goal, the global path, and speed and discards illegal trajectories, like the ones that include collision with obstacles. Based on the score of each assessment, the model then chooses the highest-scoring trajectory and uses the velocity to make the local path.

In DWA, tuneable parameters can be grouped into the following categories

- Robot
- Goal Tolerance

- Forward Simulation
- Trajectory Scoring

Robot

In this section, similarly to the Robot Parameters category of the TEB Local Planner, the parameters related to the dynamics and type of robot are defined.

Therefore, the robot footprint model was also defined in the same way as before, as a polygon, and with the list of vertices, the axis of rotation assumed to be $[0, 0]$.

Thus, the footprint model of the robot was defined as:

```
footprint_model:
  type: "polygon"
  vertices: [[-0.355, -0.235], [0.355, -0.235], [0.355, 0.235], [-0.355,
    0.235]]
```

The limit, maximum, and minimum values of the velocities and accelerations of the robot in the x and y plane, as well as its rotational velocity and acceleration parameters, are also defined in this section. These values were defined as the default values for starters and then did not need to be changed for the robot to move perfectly, which leads to the conclusion that the default values were the optimal values for this particular robot.

Goal Tolerance

In this section, the parameters define how much is an acceptable deviation from the goal, particularly in terms of `xy_goal_tolerance`, which is the tolerance in meters for the controller in the x and y distance when achieving a goal and the `yaw_goal_tolerance`, which is the tolerance in radians for the controller in yaw/rotation when achieving its goal. The `latch_xy_goal_tolerance`, which is a parameter that defines whether the robot rotates in place when the robot reaches the goal xy location, which can lead to it being outside of the goal tolerance while doing so was defined as `false`, which is its default value.

The goal tolerance parameters were defined as the following:

```
xy_goal_tolerance: 0.10 m
yaw_goal_tolerance: 0.15 rad
latch_xy_goal_tolerance: false
```

Similarly to the TEB local planner, it was considered that the xy value should not be too high and the yaw value should not be too low. These values were the last to be tuned before achieving the most optimal trajectory, leading to the robot being more flexible in calculating trajectories. Before this tune, and with the default values for these parameters, the robot sometimes assumed that the trajectory was not feasible and did not move towards the goal.

Forward Simulation

The `sim_time` value should be set between 2.0 and 5.0, because values smaller than 2.0 result in poor performance from the robot. When setting the `sim_time` to values higher than 5.0, the local planner's curves will be too long and not too flexible, while also using too much computer processing power. In the case of this robot, the optimal `sim_time` was thought to be 2.2. The `sim_granularity` parameter was set as its default value. The `vx` sample value was set to 20 due to the high-velocity value on the x-axis. The `vy` samples value was set to 3, due to the low-velocity value on the y-axis. The `vth` samples value was set to 40, due to the need for more computational power in turning, rather than in moving straight.

The forward simulation parameters were therefore defined as the following:

```
sim_time: 2.2 s
sim_granularity: 0.025 m
vx_samples: 20
vy_samples: 3
vth_samples: 40
```

Trajectory Scoring

According to ROS's DWA Local Planner guide, the cost of the objective function is calculated like this:

$$\text{cost} = k_1 \cdot d_p + k_2 \cdot d_g + k_3 \cdot c_{\max} \quad (3.1)$$

where:

d_p = distance (m) to path from the endpoint of the trajectory

d_g = distance (m) to a local goal from the endpoint of the trajectory

$c_{\max}$ = maximum obstacle cost along the trajectory (0 – 254)

k_1 = `path_distance_bias`

k_2 = `goal_distance_bias`

k_3 = `occdist_scale`

The goal for this objective function is to get the smaller cost. The path distance bias defines how similar the local planner path should be to the global planner path. Because the global planner is accurate, the weight given to this parameter was very high.

The goal distance bias defines how much the robot should try to reach its goal, which can sometimes mean that increasing this parameter leads the robot not to follow the global path as much. This value was defined as 18.0 to allow the robot to follow the global path while attempting to reach the local goals.

The occdist scale parameter allows the robot to avoid obstacles as much as possible. A high value for this parameter can sometimes make the path unviable, and the robot may be unable to move and decide which trajectory to follow. To mitigate this issue, a value of 0.8 was chosen.

The trajectory scoring parameters were therefore defined as the following:

```

path_distance_bias: 40.0
goal_distance_bias: 18.0
occdist_scale: 0.8
forward_point_distance: 0.45 m
stop_time_buffer: 0.2 s
scaling_speed: 0.25 m/s
max_scaling_factor: 0.2
publish_cost_grid: true

```

3.4 Discussion of Results

Regarding the algorithms for Autonomous Navigation, three were analysed: the Base Local Planner, the TEB Local Planner and the DWA Local Planner.

For the simulations with AMRs it was important to build a good foundation and define the best planner with the tuned parameters that would work better in the environment provided in the simulation. In this case, the deflection of obstacles and passage through choke points were specially important due to the map the vehicles were moving through, therefore attention was given to these aspects.

All three planners were tuned so as the robot moved in a straight line and deflected from obstacles perfectly.

The Base Local Planner albeit simple, performed good with the parameters tuned to its robot, in simulation, as can be seen in 3.3. Despite this, when detecting an obstacle, and recalculating a new path around said object, the local planner orange line can sometimes be desfigured, although in all tests it had a success rate of 100% when going around objects.

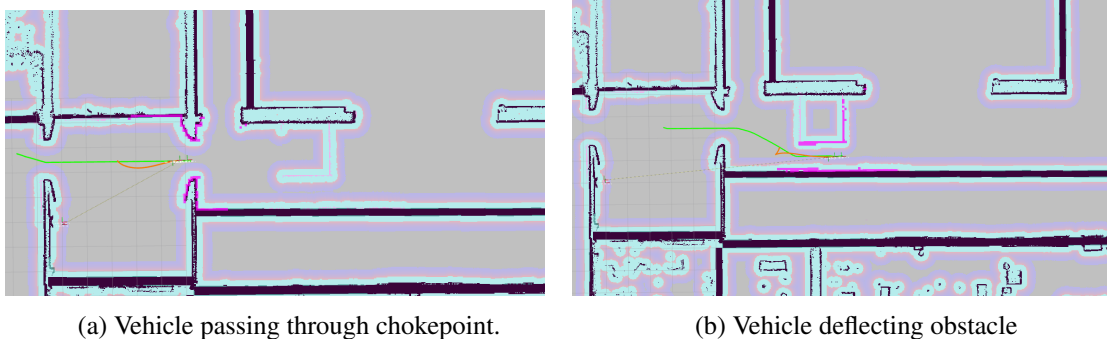


Figure 3.3: AMR with Base Local Planner.

The DWA Local Planner is a more complete version of the Base Local Planner, due to the fact that it implements the same algorithm, the Dynamic Window Approach algorithm. Despite being a more complete algorithm, it has some disadvantages, such as simplified predictions and therefore suboptimal solutions and also the inability to go backwards when predicting a path, which makes going into tight spaces like a chokepoint difficult, particularly in a fleet.

It showed good behaviour in the simulation with only one robot, as shown in Fig. 3.4.

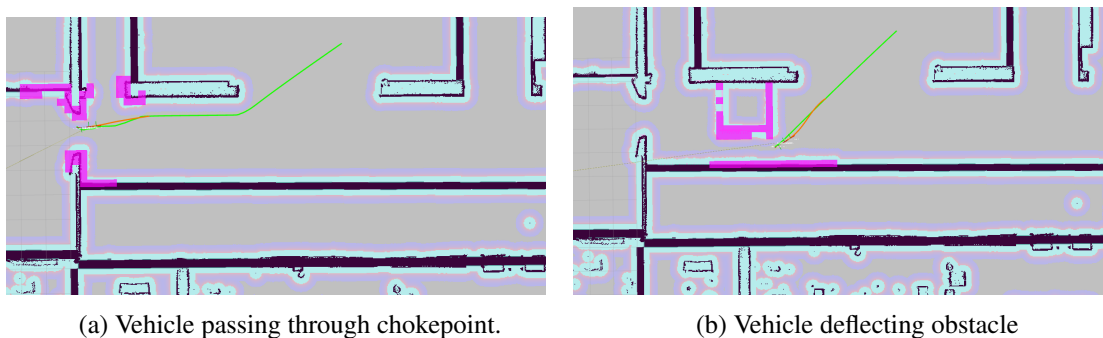


Figure 3.4: AMR with DWA Local Planner.

The TEB Local Planner is the most complete planner, with a section of parameters to be tuned exclusively related to obstacle deflection, an important area to focus on, mainly due to the handling of dynamic obstacles effectively, useful in the fleet management environment.

Tests with a real robot were carried out and it is important to note that the robot used was unidirectional and only had a laser in the front of the vehicle, so a parameter named `weight_kinematics_forward_drive` had to be adjusted in order to force the robot to choose only forward directions.

TEB is the only local planner that is able to define whether the robot can only move in one direction or both, important for tests with the real robot. It also showed good clearance when traveling in narrow regions, as can be seen in figure 3.5.

Because this planner is a very known planner and a complete one, it was easy to find information online easily, which made this planner the most intuitive to tune and use.

From these three algorithms, the TEB Local Planner was the one chosen to implement in the several tests made for the comparison of AGV and AMRs, in the autonomous navigation.

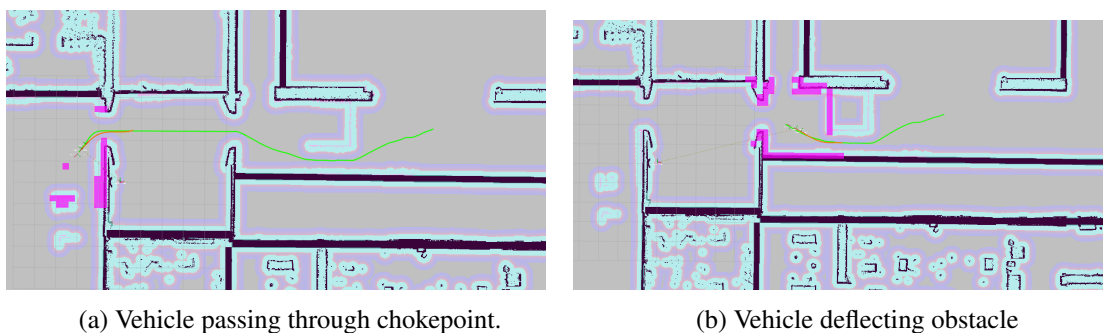


Figure 3.5: AMR with TEB Local Planner.

3.5 Fleet Manager for Autonomous Navigation

To achieve better results, specifically in terms of reduction in the number of collisions, a fleet manager was implemented so that the robots would detect each other and not collide. Although the `move_base` package implements this thought process, with the obstacle detection and deflection, it was necessary to implement code that did this more efficiently in order to get reliable and trustworthy results.

The created fleet manager processes information from the laser regarding obstacles in its vicinity and analyzes this data to establish a security area of 1.5 meters around the robot. When other robots enter the security area, it implements a priority management system that reduces the robots' velocity to 60%.

In order to better explain the thought process behind the fleet manager, below are block diagrams made to explain the thought process behind the functions and code developed for this section.

For this Fleet Manager, one function, four subscribers and one publisher were created:

- Function *isCollision*: returns True if the robot is within less than 1.5 meters from another robot and False if not;
- Subscriber - *fleetManagerCallback*: subscribed to the `/move_base/feedback` topic, which contains the robot's current position in the map;
- Subscriber - *robotManagerCallback*: subscribed to the `/move_base/status` topic, which contains an array that provides status information of the robot, 1 meaning that the robot is active and the action server is currently processing the goal;
- Subscriber - *laserCallback*: subscribed to the `/robot_0` topic, which corresponds to the front laser of the robot and from where all information relating this sensor can be analysed;
- Subscriber - *velocityCallback*: subscribed to the `/cmd_vel`, the topic with the current velocity commands of the robot;
- Publisher that publishes to the `/cmd_vel` topic in order to set a new velocity of the robot (when it detects an obstacle, it sets the velocity to 60%).

Firstly, in order to build the fleet manager, it is important to have a vector with the active robots updated at all times, the *fleetManagerCallback* and the *robotManagerCallback* being essential to manage it. The first callback was used to populate the vector with new active robots and the second callback to erase any robot that isn't active anymore. This vector, called *rob_vector* is used as a reference to know how many and which robots are active at all times and their position in the map.

In the *fleetManagerCallback*, the subscribed topic is `/robot_id/move_base/feedback`, which contains the robot's current position in the map. If the *rob_vector* is empty, the callback checks if there is an active robot by referring to the topic and populates the vector with the robot's current position and name.

In order to erase no longer active robots from the vector, the *robotManagerCallback* is called. In this callback, the subscribed topic is */robot_id/move_base/status*, which provides status information on the goals sent to the *move_base* action. In this case, this callback first checks *rob_vector* for the number of active robots in the vector. It then loops through the vector and checks if the status of any robot is different than 1, which means that the robot is no longer moving and therefore not active anymore. If so, this robot is erased from the *rob_vector*.

The *laserCallback*, represented as a diagram in Fig. 3.6, is subscribed to the */robot_id/scan_0* topic, which corresponds to the front laser of the robot. This topic has information about the sensor, and from this topic it was possible to obtain the ranges array, which has the range data of the laser.

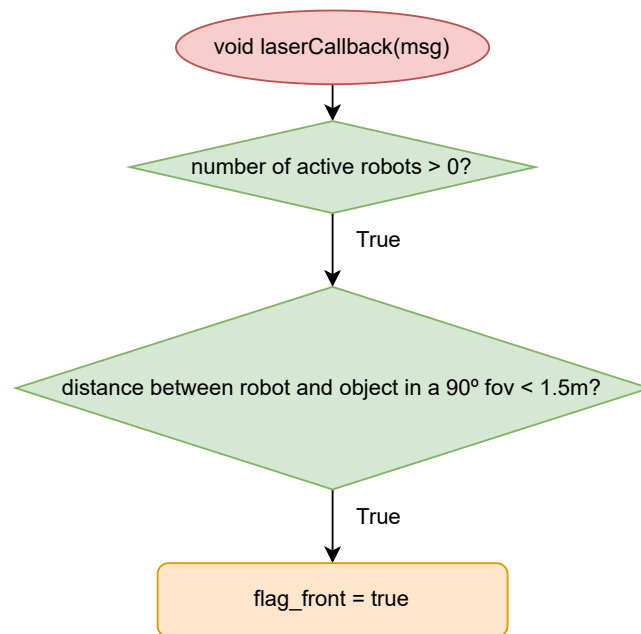


Figure 3.6: Laser Callback.

The ranges array has 508 positions, with values ranging from 0 to 29.0, 0 being the minimum range of the laser and 29.0 being the maximum range value of the laser in meters. These 508 positions have values for the robot's Field Of View (FOV), which is a 270° angle.

For the laser to only detect obstacles within a 90° FOV, it was necessary to calculate the number of positions in the range array equivalent to this angle, which was 170 positions. This thought process is explained in figure 3.7.

In the callback, the array of ranges is looped through from the beginning of the 170 positions, calculated from the $-170/2$ position regarding the middle of the array until the $170/2$ (85th) position. The values are verified, and if one or more are within 1.5 meters, it signals a flag that an

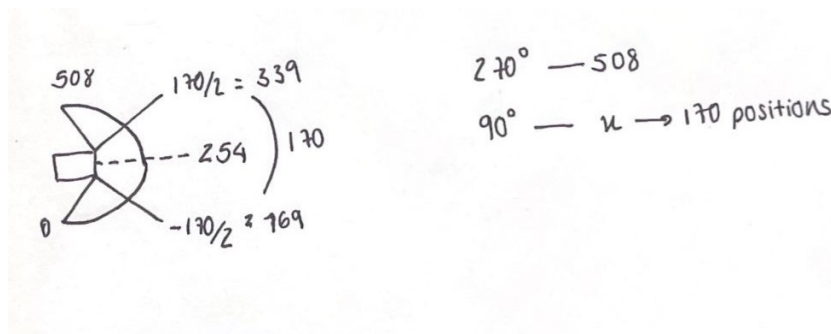


Figure 3.7: Number of positions in the range array calculated for the 90-degree angle.

obstacle was detected.

The *isCollision* function, represented in the diagram in fig. 3.8, receives two positions of two active robots and calculates its distance, returning a boolean value that is true if the two robots are within the value set in the variable *SAFE_DISTANCE*, which is defined to be 1.5 meters in the simulations made. This ensures that the fleet manager only detects robots and not other static obstacles and walls.

In the main function, represented in Fig. 3.9, the code developed made it possible to know when the robots are near each other, close to collision, and to slow one of them down to allow for the other to move forward.

When the *rob_vector* is populated with more than one robot, it means that two robots are moving at least and, therefore, constitute a fleet that can be managed. The vector is looped through twice to get the index of two moving robots that can be compared.

Then, the code evaluates which of the robots that are being compared has an activated laser. If robot1 is detecting robot2, robot1 is behind robot2, which indicates that robot1 should be the one to be slowed down, therefore prioritizing the movement of robot2. When it is detected that a robot has to slow down its movement, a flag called *velocity_flag* is activated.

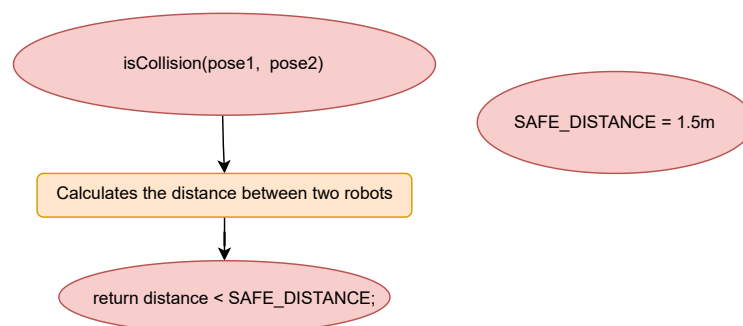


Figure 3.8: Function that detects if two robots are within 1.5m of each other.

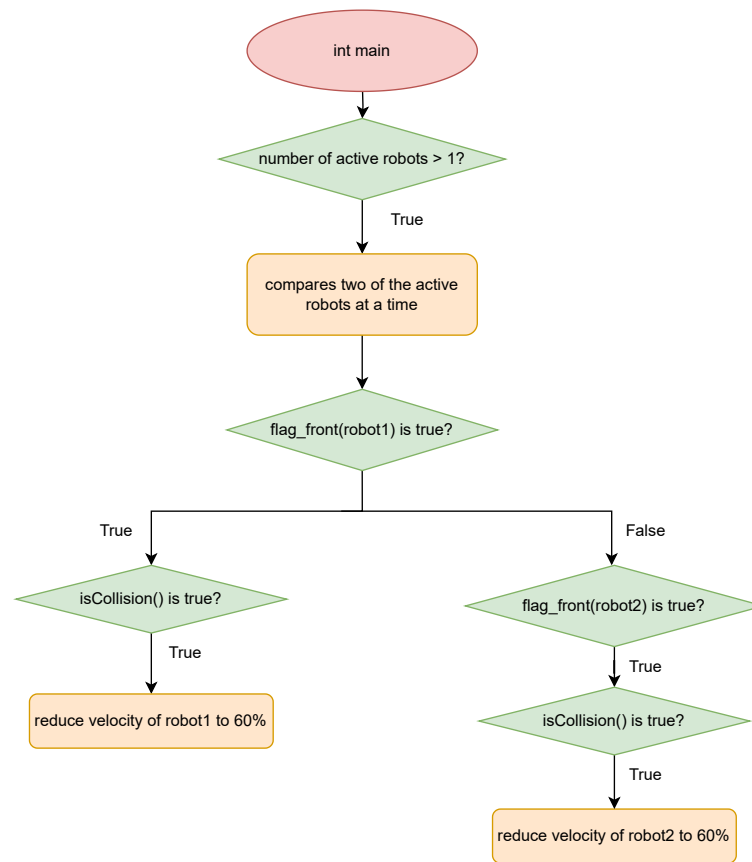


Figure 3.9: Main.

The *velocityCallback* is a callback subscribed to the */robot_id/cmd_vel* that, when *velocity_flag* is activated, rights to a publisher that sets the velocity to 60% of the current velocity of the robot, signaled by the topic. Therefore, the collision is avoided.

Chapter 4

Discussion and Results

The results of the work developed are presented in chapter 4 with a complete explanation and exploration of the results without a fleet manager. The results for a fleet of five robots with a fleet manager are further explored afterward. Finally, the simulation results of the local planner chosen with a real robot are displayed.

To facilitate a comparative analysis of the two fleets, a simulation environment was created. The RViz and Stage simulation tools were used to create a visual representation of the environment. These tools allow the visualisation of robot models, laser data, TF (Transform package) and the map of the environment. The simulation environment is shown in figures 4.4, 4.7 and 3.5.

A map that allowed for the free and guided navigation of vehicles was loaded to the RViz environment. This map can be seen in Figure 4.1.



Figure 4.1: Map used for the simulations of both AGVs and AMRs in RViz.

4.1 Results without Fleet Manager

This section presents the results of the simulations without the use of a Fleet Manager for autonomous navigation.

Comparisons between both types of robots are made in the following configurations: individually, in a fleet of five robots, and in a fleet of ten robots. These comparisons are made by defining missions and some KPIs to evaluate the performance of the vehicles. Finally, the results obtained are discussed.

4.1.1 Comparisons between one AMR and one AGV

To begin the comparisons between the two types of robots, it was necessary to define some KPIs for guided and autonomous navigation.

The two most effective KPIs to measure the differences between the two were duration of time to perform the mission and the success rate of three pre-defined missions for both robots. These pre-defined missions were defined with pointing a vertice in the map that was easily accessible when simulating with both fleets of guided navigation robots and free navigation robots.

Three different distanced missions were defined, a short, medium and long distance mission. In Figs. 4.2, 4.3 and 4.4 the starting vertice of both an AGV and an AMR is circled in red and the end vertice of the mission is circled in yellow and the three missions are showed.

Three attempts were made for each of the missions and for both types of vehicles and, in order for the AGV and the AMR simulation to be more reliable, each of the edges had a velocity of 0.5 m/s while the maximum velocity of the planners for the autonomous navigation was also of 0.5 m/s.

The autonomous vehicle performed better in terms of time at an average of 7 seconds and 320 milliseconds quicker and the success rate of both vehicles was of 100%.

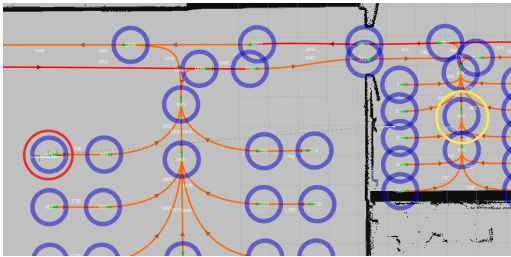


Figure 4.2: Start and end vertice for the small mission.

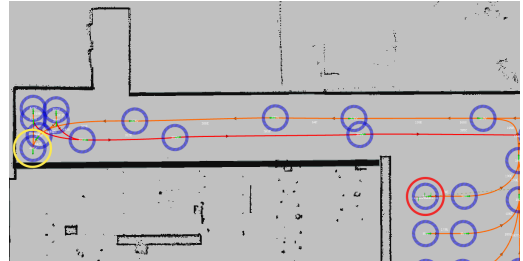


Figure 4.3: Start and end vertice for the medium mission.

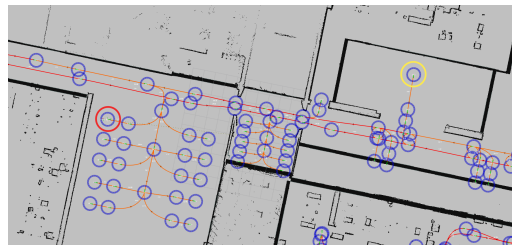


Figure 4.4: Start and end vertice for the long mission.

In a separate file, in order to define the mission for guided navigation, the starting vertice was defined as Id 11083, and is the vertice where the singular robot starts its trajectory. The mission is defined by the trajectory made by any robot between the starting vertice, which has the Id 83 and the end vertice, defined as 11065, 11441 and 11196 for the small, medium and long mission, respectively.

Below is the file where the edges of the missions were also defined, as the trajectories between the origin vertice and the destination vertice.

```

Vertices:
- { Id: 11196, FrameId: map/nn0, X: 14.015871047973633, Y:
    8.253355026245117, ThetaRads: -1.5068833261949972 }
- { Id: 11065, FrameId: map/nn0, X: 2.5130701065063477, Y:
    0.28277501463890076, ThetaRads: 1.5702982354039472 }
- { Id: 11441, FrameId: map/nn0, X: -30.19129753112793, Y:
    1.587499976158142, ThetaRads: -1.5992225501887076 }
Edges:
- { Id: 11003, Origin_ID: 83, Destination_ID: 11196 }
- { Id: -11003, Origin_ID: 11196, Destination_ID: 83 }

- { Id: 11011, Origin_ID: 83, Destination_ID: 11065 }
- { Id: -11011, Origin_ID: 11065, Destination_ID: 83 }

- { Id: 11012, Origin_ID: 83, Destination_ID: 11441 }
- { Id: -11012, Origin_ID: 11441, Destination_ID: 83 }

```

The guided vehicle follows a defined trajectory shown in Fig. 4.5 for the long mission. The three attempts were made using an iPhone stopwatch and the start and stop reaction error is not relevant. The results are presented in the table 4.1 with the average ToA for the three distance missions.

The success rate of these missions for the guided navigation was of 100%.

For the free navigation, the start and end vertices were the same as the ones defined in the guided mission and the vehicle would navigate to the end vertice using move_base, with the TEB Local Planner. The end goal was set in the RViz simulation environment, on top of the vertice defined as the end vertice in the guided mission.

The trajectory followed is presented in Fig. 4.6 for the longer mission. The three attempts for the three missions were made using an iPhone stopwatch and the start and stop reaction error is not relevant. The results are presented in the table 4.1 below. The success rate of these missions for the autonomous navigation was of 100%.

Autonomous navigation is a faster method of navigation for a single path-following vehicle. AGVs are restricted to following a predetermined path, passing through every node and edge of

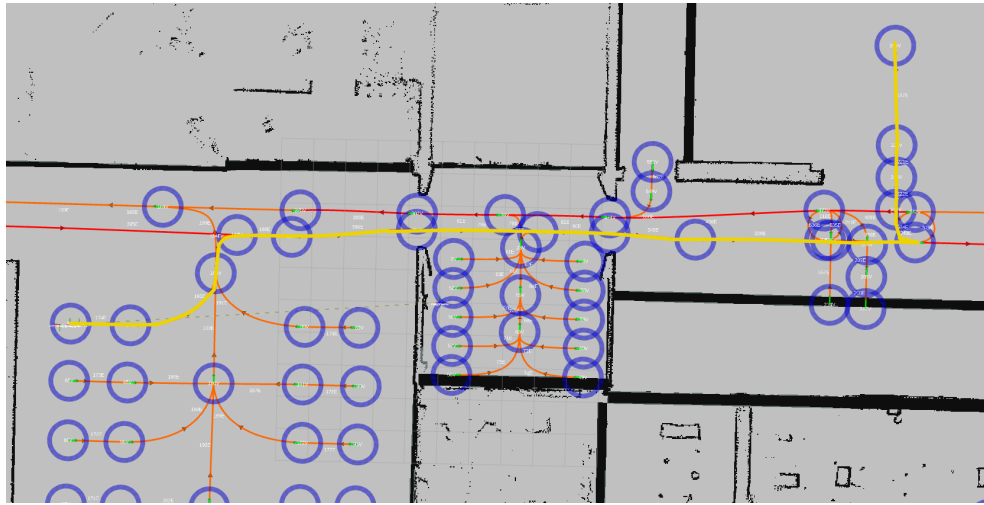


Figure 4.5: Trajectory followed by the guided vehicle to fulfill the mission.

Table 4.1: Simulation time results for the three distance missions, for the AMR and AGV.

Missions	Average ToA for AMR	Average ToA for AGV
Short	00:35,41	00:40,38
Medium	00:46,38	01:02,70
Long	01:06,63	01:13,95

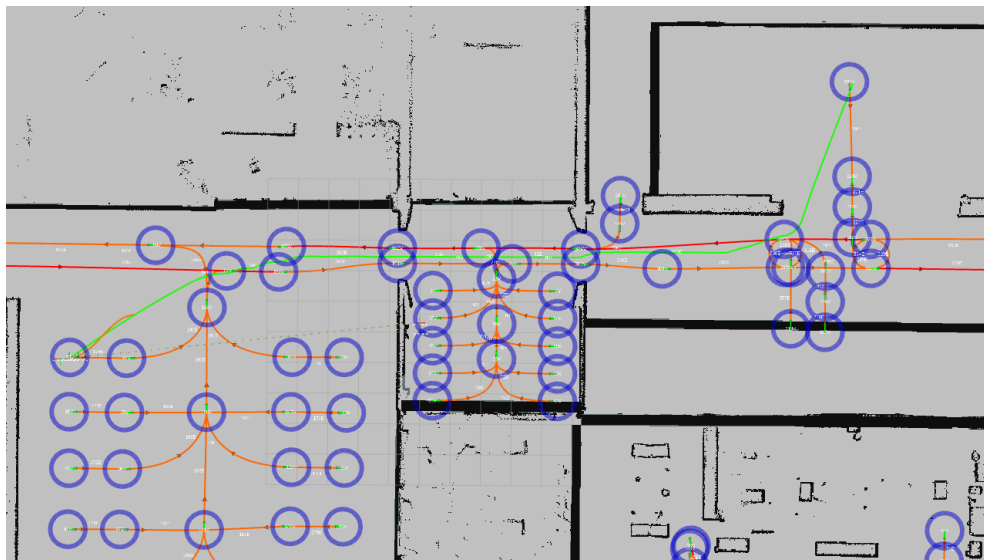


Figure 4.6: Trajectory followed by the autonomous vehicle to fulfill the mission.

that path. AMRs, on the other hand, are not constrained in this way and aim to take the shortest route to their destination.

Furthermore, AMRs can navigate around both static and dynamic obstacles, whereas AGVs are programmed to stop and wait until an obstacle is removed from their path. This shows that

free navigation is a more time-efficient solution.

4.1.2 Comparisons between a fleet of five AMRs and five AGVs

In order to compare a fleet of five AMRs and five AGVs it was necessary to create five goals next to each other in the area of the goal defined in the previous mission. This is because the simulation provided doesn't allow for more than one agent to have the same target.

Additionally, for these simulations, a script was created that published the ID of the goal vertex for each of the robots, so all the robots would initiate their mission at the same time.

Ten attempts were made to allow for more accurate results and conclusions and three KPIs were evaluated, them being the success rate, the duration of the mission and the number of collisions. The duration of the missions were recorded using a smartphone's stopwatch, and the start and stop reaction is to be ignored, as it is not high enough to be relevant for the results.

4.1.2.1 Mission evaluation for five AMRs

The goals provided for the 5 vehicles are as displayed in Fig. 4.7b and the mission results are presented in table 4.2.

This mission is difficult for the robots with autonomous navigation because of a narrow passage showed in Fig. 4.7a, which is necessary to pass in order to achieve the goal. In this passage, some difficulties arose at times which made the robots take more time to arrive to the goal and, in worst cases, collide with each other.

Simulation	First robot's ToA	Last robot's ToA	Number of collisions
1	01:07,67	01:36,89	0
2	01:07,81	01:37,10	0
3	01:07,77	01:34,58	0
4	01:07,60	01:36,86	0
5	01:08,22	01:41,55	1
6	01:07,89	01:35,61	0
7	01:08,23	01:39,32	1
8	01:08,36	01:31,74	0
9	01:07,18	01:37,48	1
10	01:07,56	01:33,59	1

Table 4.2: Simulation results for the fleet of five free navigation robots.

In the table 4.2, the worst and best ToA (Time of Arrival) for each of the robots appears in red and green respectively. When analysing the table, the average ToA of the first robot was 01:07,83, which doesn't constitute a big difference from the range of times achieved by the 10 simulations. The first robot was always recorded as being the yellow robot, because its trajectory starts closer to the passage, which allows it to pass through it without problems.

The average ToA of the last robot was 01:36,42, and the range between both the best and worst ToA is of 10 seconds, which constitutes a big difference. This happened because the rest of the robots would get stuck in the narrow passage shown in 4.7a and would take a while to pass through it, due to high traffic. Because the fleet is not intelligent enough to allow some robots to pass through first, the timestamps are not consistent and sometimes there are collisions.

The success rate of these missions was of 100%. and the rate of collisions was of 40%, given that when it happened, there was only one collision at a time.

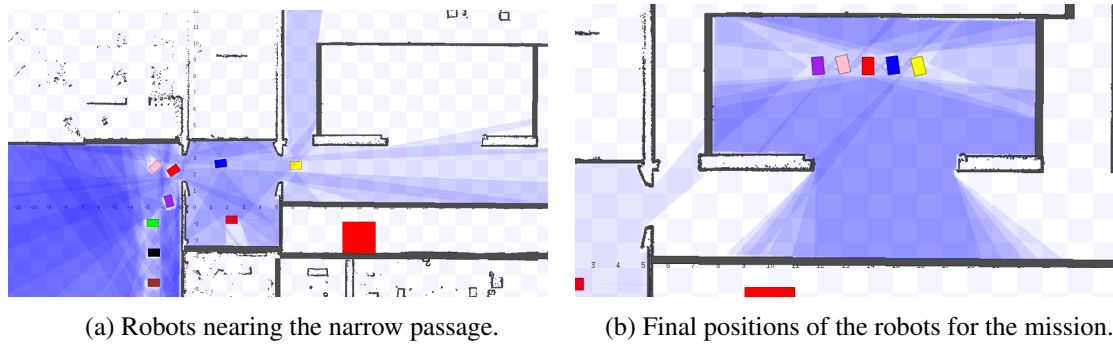


Figure 4.7: Five robots performing the mission with free navigation.

4.1.2.2 Mission evaluation for five AGVs

The simulation for the Automated Guided Vehicles was made with the use of the TEA* algorithm and the path that the vehicles followed is shown in Fig. 4.8 in yellow.

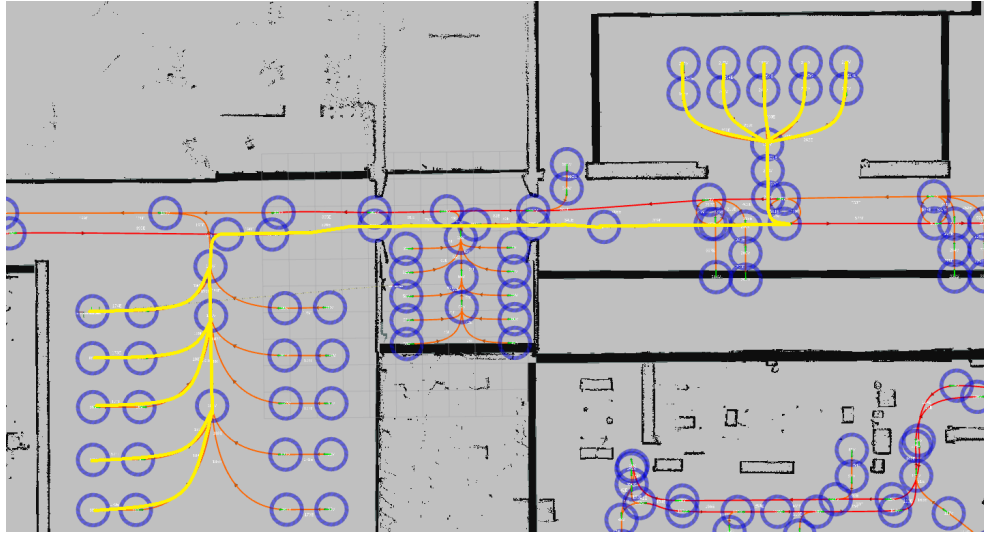


Figure 4.8: Trajectory followed by the guided vehicles to fulfill the mission.

Ten simulations were performed in order to have more accurate results and these are presented in table 4.3. The average Time of Arrival of the first robot was of 01:20,51 and the range between

the best and worst ToAs was of 15 seconds. The average Time of Arrival of the last robot was of 02:38,23 and the range between the best and worst ToA was of more than 20 seconds.

This happens because the priority management of the TEA* algorithm used in this simulation isn't developed fully, which meant that the first robot to arrive was unmethodical, depending on the simulation and which robot was able to leave the initial position of five robots first. Thus, no robot followed an order of arrival, which also resulted in very different times of arrival.

On the other hand, the number of collisions in the guided simulation was of zero, due to the fact that there was always at least one vertex in between two vehicles. The success rate of these missions was of 100%. and the rate of collisions was of 0%.

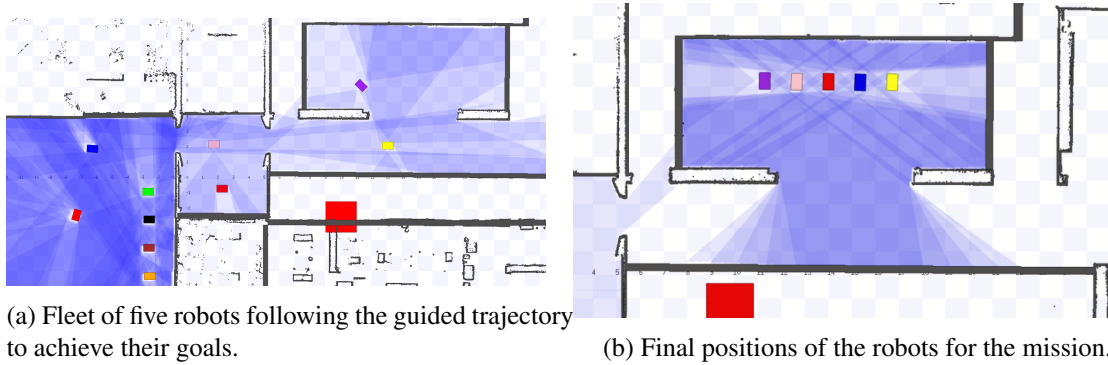


Figure 4.9: Five robots following a set path to perform the mission.

Simulation	First robot's ToA	Last robot's ToA	Number of Collisions
1	01:23,15	02:38,38	0
2	01:29,83	02:48,10	0
3	01:30,04	02:40,86	0
4	01:15,67	02:44,52	0
5	01:15,24	02:30,35	0
6	01:16,34	02:41,02	0
7	01:16,23	02:36,44	0
8	01:17,42	02:27,86	0
9	01:16,86	02:30,82	0
10	01:24,35	02:43,96	0

Table 4.3: Simulation results for the fleet of five guided navigation robots.

4.1.3 Comparisons between a fleet of ten AMRs and ten AGVs

For this mission, ten goals were created side-to-side, similar to the fleet of five robots. The script created for this mission was also similar to the one created for the fleet of five vehicles, adapted to the ten vehicles.

The simulations were repeated ten times and the following KPIs were evaluated: the success rate, the duration of the mission and the number of collisions. The duration of the missions were recorded using a smartphone's stopwatch.

4.1.3.1 Mission evaluation for ten AMRs

Firstly, the mission for the AMRs was made so that the goals of the 10 robots were in two lines of five robots. This was not the best position for them, because there was the possibility of one of the three inside robots from the top line not be able to complete its' mission. This happened one time and is shown on Fig. 4.10. The trajectory planning is not smart, so that the robots never find themselves in this situation.

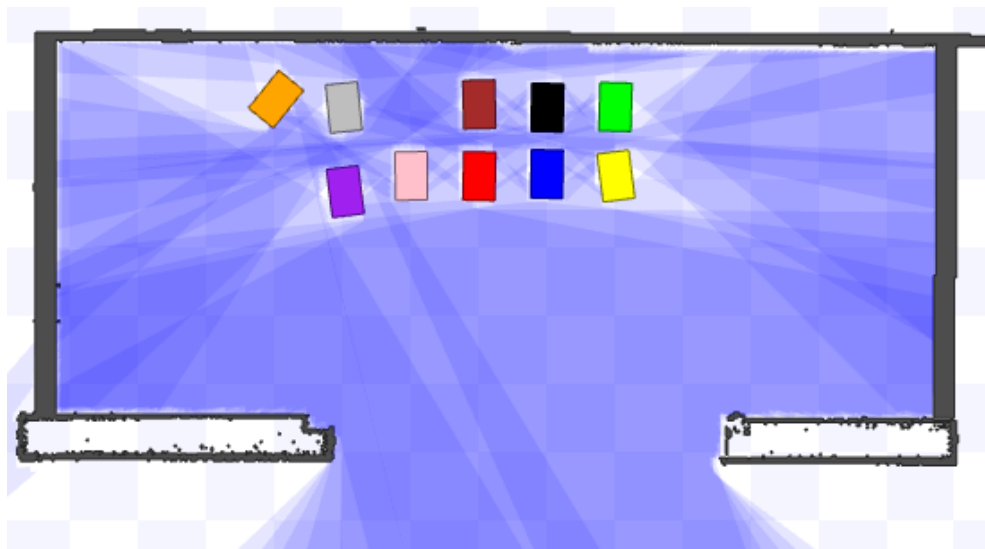


Figure 4.10: Orange robot attempting to finish trajectory.

The goals for this mission were therefore defined to be side-by-side as shown in Fig. 4.11.

The results of the simulations are presented in table 4.4. The average time of arrival of the first robot was of 00:54,74 and the range between the best and worst time was of 2 seconds. The first robot to arrive was also consistently robot 6, and it was observed that this happened not because of the smartness of the vehicles but because robot 6 was always closer to the passage, it was always the first one to arrive and pass through.

The average time of arrival of the last robot was of 02:00,87 and the range between the best and worst ToA was of approximately 18 seconds. Similarly to the mission for the fleet of five robots and as it is presented in Fig. 4.12 the robots try to pass through the passage and due to not having a set order of passage, they can take not only different times to arrive but also the order of arrival is very different and the last robot to arrive is different in most of the simulations.

In the simulation 6, the blue vehicle was unable to recalculate the route when encountering obstacles, so that mission didn't succeed completely.

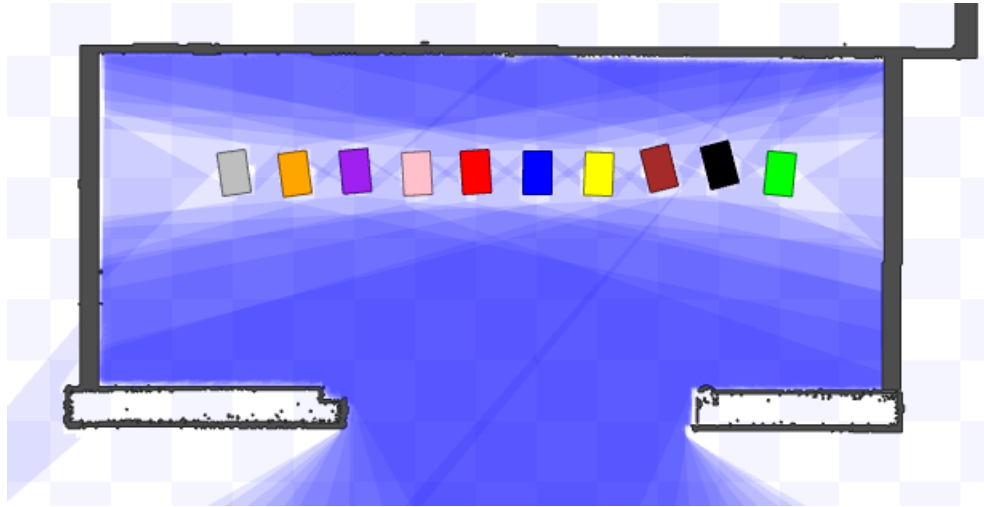


Figure 4.11: Final positions for the fleet of ten robots.

A big complication to analyse is also the number of collisions, which increased, when compared to the fleet of five robots.

The success rate of these missions was of 90%. and the rate of collisions was of 80%, given that the average number of collisions in a mission was of 1.8 collisions.

Simulation	First robot's ToA	Last robot's ToA	Number of Collisions
1	00:54,64	02:10,57	1
2	00:55,88	01:57,94	1
3	00:53,23	01:57,51	3
4	00:55,74	01:59,37	1
5	00:54,16	01:52,45	0
6	00:54,39	01:57,55	0
7	00:54,16	02:01,77	2
8	00:55,31	02:03,68	2
9	00:55,33	02:05,83	3
10	00:54,51	02:02,01	5

Table 4.4: Simulation results for the fleet of ten free navigation robots.

4.1.3.2 Mission evaluation for ten AGVs

The goals planned for the fleet of ten robots are presented in Fig. 4.14. In Fig. 4.13a, it is possible to observe that the vehicles are exiting their initial positions unmethodically and following the guided trajectory towards their ultimate goal.

In Fig. 4.13b, the last robot, in this case the purple robot, has started its trajectory in order to achieve the goal and it's possible to observe that there are usually around four robots performing the trajectory at the same time, which explains the excessive time it took for all ten of the fleet of robots to achieve their goals.

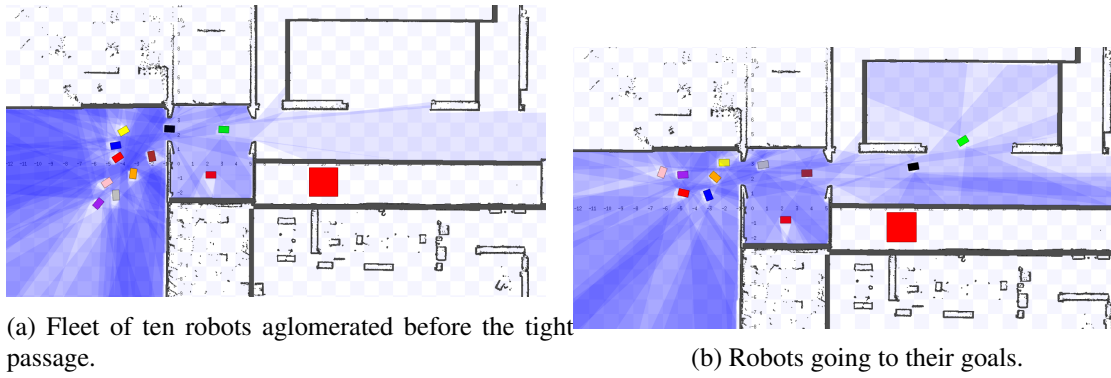


Figure 4.12: Fleet of ten robots following the autonomous trajectory to achieve their goals.

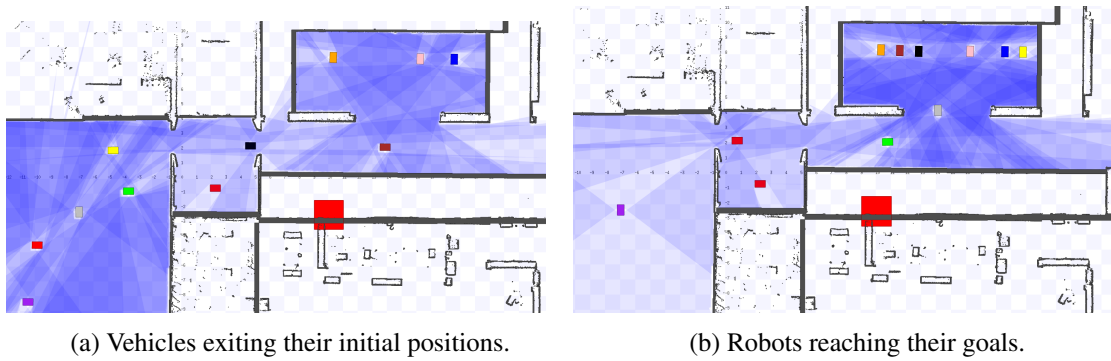


Figure 4.13: Fleet of ten robots performing the mission using guided navigation.

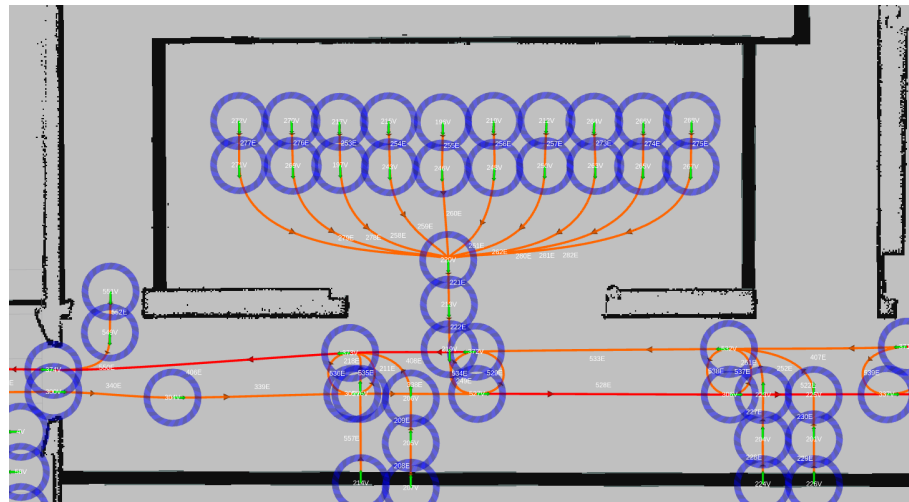


Figure 4.14: Goals planned for the guided navigation of the fleet of ten robots.

The results of these simulations are displayed in table 4.5. The average ToA for the first robot was of 01:24,14 and the range between the best and worst ToA was of approximately 11 seconds. The first robot to arrive wasn't consistently the same, because the algorithm of priorities for this simulation wasn't optimized and therefore the range between the best and worst ToA was so high.

The average ToA of the last robot was of 04:23,26. This fleet of robots took a lot longer to perform the mission when compared to the fleet of ten autonomous vehicles. The range between the worst and best ToA was of approximately 29 seconds, which constitutes a very high range of time.

The number of collisions in these simulations was zero, due to the planning made before to ensure that the robots would not collide. The success rate of these missions was of 100%. and the rate of collisions was of 0%.

Simulation	First robot's ToA	Last robot's ToA	Number of Collisions
1	01:21,85	04:09,44	0
2	01:24,64	04:16,58	0
3	01:26,34	04:38,62	0
4	01:29,04	04:35,75	0
5	01:18,06	04:15,07	0
6	01:27,07	04:11,01	0
7	01:22,54	04:21,08	0
8	01:23,79	04:33,83	0
9	01:26,72	04:27,80	0
10	01:21,30	04:23,26	0

Table 4.5: Simulation results for the fleet of ten guided navigation robots.

4.2 Results with Fleet Manager

Fleet management systems are critical in coordinating and optimizing the operations of both free navigation and guided navigation robot fleets. These systems provide a centralized platform for managing multiple robots, regardless of their navigation type, increasing efficiency and productivity in various industrial environments. A robust fleet management system is essential to maximize the potential of robot fleets in industrial environments.

In this work, a fleet management system for the AMR fleet was implemented and tested and was already introduced in chapter 3. The TEA* algorithm, talked about in the state of the art, in chapter 5, was further improved during the development of this work, and the simulations are included in this section.

4.2.1 Comparison between a fleet of five AMRs and five AGVs

For this comparison, the missions performed were the same as the missions done before for the comparison between a fleet of five AMRs and five AGVs without the fleet manager. Ten attempts were made for these simulations, in order to get more reliable conclusions, and the three KPIs evaluated were the success rate, the duration of the mission and the number of collisions.

The missions' duration was recorded with a smartphone's stopwatch, and the start and stop reactions weren't relevant to the results.

4.2.1.1 Mission evaluation for five AMRs

In order to implement a Fleet Manager in the free navigation fleet, it is important to define which kind of technique to follow. The option to check for possible collisions inside of a safety zone between two vehicles was explored. A collision prevention system was created by receiving laser data and monitoring the robot's surroundings, designed to ensure that each robot can navigate safely in its environment.

The goals planned for the fleet of five AMRs were the same as the ones planned in the free navigation simulations without the fleet manager. They are displayed in Fig. 4.7b.

Table 4.6 displays the simulations' results. In this table, the worst and best ToA appear in red and green, respectively. The average ToA for the first robot was 01:02,83, constituting a time improvement of five seconds when compared to the simulation without the fleet manager.

The success rate for this mission was of 100%. The average ToA for the last robot was 01:36,56, which is similar to the average ToA for the last robot in the same simulation without the fleet manager. This means that, time-wise, the fleet manager wasn't effective.

The biggest and most important difference that can be seen when comparing the two missions is that in this mission, while using the fleet manager, there were no collisions recorded in any of the simulations performed, which was a key point in the comparison between the fleet of AGVs and the fleet of AMRs.

Simulation	First robot's ToA	Last robot's ToA	Number of Collisions
1	01:03,78	01:47,18	0
2	01:02,01	01:39,17	0
3	00:58,23	01:26,14	0
4	01:03,40	01:30,11	0
5	01:01,12	01:31,03	0
6	01:03,47	01:33,19	0
7	01:04,73	01:37,14	0
8	01:01,17	01:37,10	0
9	01:03,02	01:43,43	0
10	01:07,37	01:41,12	0

Table 4.6: Simulation results for the fleet of five free navigation robots with the fleet manager.

4.2.1.2 Mission Evaluation for five AGVs

The improvement of the TEA* algorithm introduced the need to test it in order to get a clear comparison between the performance of the fleet of AGVs with the latest version of the algorithm and the performance of the fleet of AMRs with the fleet manager developed.

The goals planned for the fleet of five AGVs were the same as the ones planned in the guided navigation simulations without the fleet manager. The simulations' results are shown in table 4.7. The worst and best ToA is depicted in red and green, respectively.

Simulation	First robot's ToA	Last robot's ToA	Number of Collisions
1	01:19,59	02:14,99	0
2	01:25,49	02:18,04	0
3	01:25,96	02:14,39	0
4	01:17,15	02:12,01	0
5	01:25,50	02:20,50	0
6	01:19,68	02:05,49	0
7	01:26,03	02:15,48	0
8	01:19,79	02:07,09	0
9	01:26,16	02:14,52	0
10	01:24,60	02:25,87	0

Table 4.7: Simulation results for the fleet of five guided navigation robots with the improved TEA* algorithm.

The average ToA for the first robot was 01:22,99, which is two seconds higher than the average ToA for the first robot in the first simulations, and two very similar values were recorded. The average ToA for the last robot was 02:14,84, which was recorded to be about 23 seconds more time efficient than the average ToA of the last robot in the first simulations.

The success rate for this mission was 100% and there were no collisions. It is clear that the improvements in the algorithm greatly improved the time efficiency of the fleet management for the AGVs.

4.3 Tests with a real robot

In order to validate the local planner chosen, the TEB local planner and the results obtained in a simulated environment, some tests were performed in a real environment, in the omnidirectional robot Platform Hangfa Discovery Q2, represented in figure 4.15, equipped with a Hokuyo UST-10LX frontal laser.



Figure 4.15: Robot used for trajectory testing.

Since this robot doesn't have a laser on its rear, it doesn't move backward, when sending a goal that is in the opposite direction, it rotates in order to start its route forward.

The trajectory testing consisted of four different tests, them being: a forward drive without obstacles, a drive with a static obstacle, a person standing; a drive with a dynamic obstacle, a person that walked in front of the robot; a drive with a dynamic obstacle near a wall, where a person walked in front of the laser toward a wall to evaluate the robot's ability to choose to move away from the wall.

Because the robot used to perform the tests differs from the robot in the simulations, some parameters were changed. Both the `global_costmap` and the `local_costmap` files were changed to only include the front laser.

Additionally, the `teb_local_planner_params` file was changed to include the `footprint_model` of the robot in the tests; the acceleration and velocity in the y axis was added; the `weight_kinematics_nh` parameter was reduced significantly in order to adjust the tradeoff between compliant longitudinal motion and non-compliant lateral motion, due to the robot being an omnidirectional robot and the `weight_kinematics_forward_drive` is set to a very high level so that the motion forward of the robot is prioritized.

```
acc_lim_y: 0.2
max_vel_y: 0.5

allow_init_with_backwards_motion: false

weight_kinematics_nh: 10.0
weight_kinematics_forward_drive: 100.0
```

The tests performed can be seen in a video in the following URL: <https://youtu.be/M5M3IP5XnXg>.

Forward drive without obstacles

In this test, the forward trajectory of the robot was tested to evaluate the robot's stability in movement. The robot behaved as expected and moved straight without relevant oscillations, keeping a straight trajectory without veering off course and with constant speed.

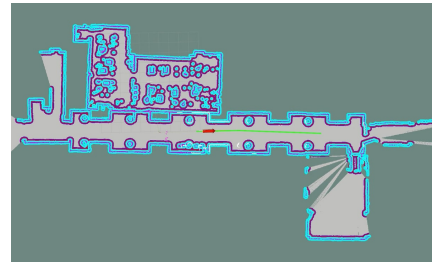
The images of the tests made towards the forward drive without obstacles are in figure 4.16, with an image of the robot moving forward and its path calculated and displayed on RViz.

Movement with static obstacle

The test where a static obstacle was introduced allowed to observe the algorithm's ability to deflect static objects. Upon detecting the static object, the robot reduced its speed and quickly calculated an alternative path, as shown in Fig. 4.17b to navigate around the obstacle. Once the obstacle was out of the FOV of the robot, it resumed its path toward the goal, shown in Fig. 4.17d.

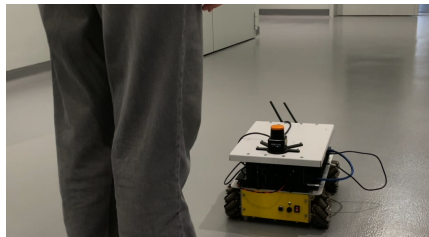


(a) Robot performing the forward motion.

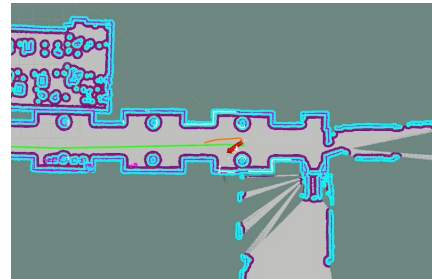


(b) Forward movement of the robot in RViz.

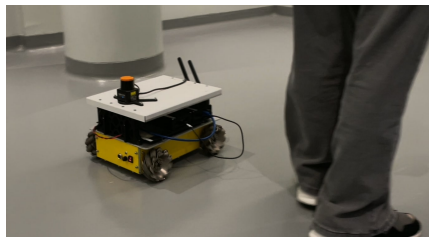
Figure 4.16: Robot moving forward in the real setting and in the simulation.



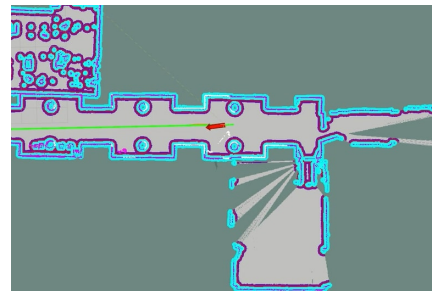
(a) Robot approaching obstacle.



(b) RViz display of the alternative path.



(c) Robot navigating around the object.



(d) RViz display of robot resuming trajectory.

Figure 4.17: Robot deflecting the static obstacle successfully.

Movement with dynamic obstacle

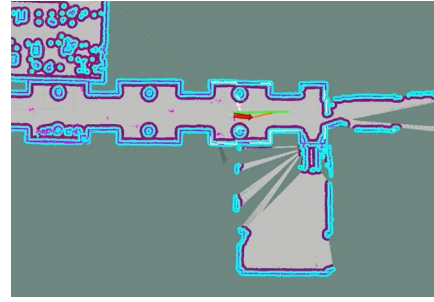
This test was necessary to see if a real robot can adapt to robust environments, with the presence of dynamic and unexpected objects, in order to prove its reliability.

When a dynamic obstacle was introduced, in this case, a person, the robot deflected from the obstacle as expected. It uses its sensors to detect and continuously track the position of the moving object. Simultaneously, it reroutes and adapts the path as the object changes position.

In Fig. 4.18, it is shown that the robot assumes a different trajectory as the obstacle is moving. When the obstacle is no longer within the laser's range, it resumes its course toward the goal.



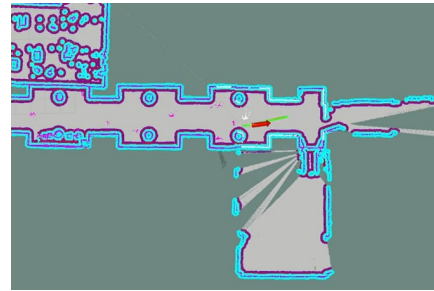
(a) Robot detecting and navigating around dynamic obstacle.



(b) RViz display of the robot calculating a new path for the dynamic obstacle.



(c) Robot after navigating around obstacle.



(d) RViz display of robot resuming its trajectory towards its goal.

Figure 4.18: Robot avoiding the dynamic obstacle.

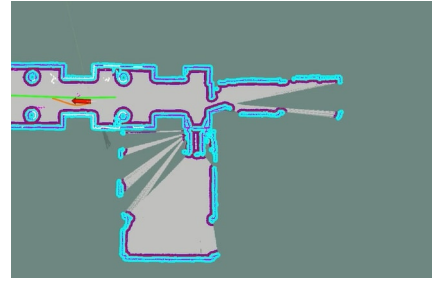
Movement with dynamic obstacle near a wall

This test was performed so as to evaluate the TEB's ability to choose the best possible path in a specific situation where a dynamic or static object is near a robot and on one side, the robot has a map limit, and on the other side, a clear way to go through. The move_base's global planner maps the entire environment while the local planner maps its surroundings. When properly implemented, the local planner algorithm should be able to choose the clear path as opposed to the wall.

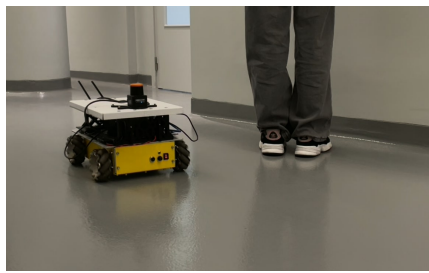
In this test, the robot also behaved as expected. When detecting the dynamic obstacle, which was walking in the direction of the wall, it started leaning toward it. Quickly, the algorithm realized that the dynamic obstacle was too close to the wall and that there was not a clear path in that direction, so it rerouted the path to the hallway, shown in Fig. 4.19.



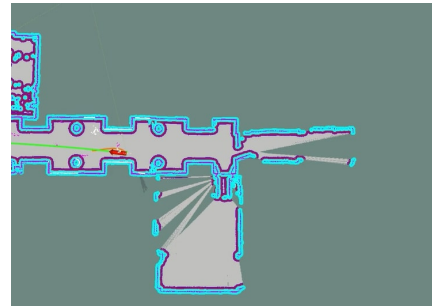
(a) Robot deflecting dynamic obstacle and moving toward the wall.



(b) RViz display of the robot calculating the path toward the wall.



(c) Robot after rerouting moving to its goal.



(d) RViz rerouting the path to deflect the obstacle in the right direction.

Figure 4.19: Robot moving around an obstacle while choosing the clear pathway.

4.4 Discussion of results

In this chapter, the results achieved from the simulations and tests performed were presented and described and will now be discussed to draw further conclusions from this work.

Some simulations were performed to conclude which fleet of robots would perform better in an environment without a fleet manager for the fleet of AMR. This was essential to evaluate the efficiency of the TEB Local Planner at managing a fleet of robots, given its efficiency at deflecting from static and dynamic obstacles.

In that case, the results were promising and leaning toward the fleet of AGV as the one to apply due to its reliability and robustness. The high number of collisions revealed the fleet of AMR to be unfeasible in a real industrial scenario, being this the only KPI that wasn't met. Yet, the AGV fleet showed to be behind in terms of time efficiency and flexibility.

In light of these results, a fleet manager was implemented to mitigate some of the issues that came up during the first simulations. This fleet manager was essential to the development of this work, and showed great results in the simulations that followed, in which there were no collisions during the 10 attempts at the same mission of a fleet of five robots.

When compared to the fleet of AGVs, with a newer version of the TEA* algorithm, the fleet of AMRs performed better in every KPI, even though the fleet of AGVs showed better results in terms of time when compared to the simulations with the older version of the TEA* algorithm.

Finally, the TEB Local Planner implemented for the simulations and comparisons performed was tested in a real robot, to evaluate how it behaved in a real setting and some different tests that simulated occurrences in a real environment were carried out, which proved this algorithm to be robust and a good choice to carry out the simulations for the fleet of AMRs.

4.4.1 Results without a Fleet Manager

This study focused on the comparison between a fleet of AGVs and a fleet of AMRs, without the implementation of a Fleet Manager. The AMRs used the TEB Local Planner for path planning and the AGVs used the TEA* algorithm.

In Fig. 4.20 a graph with the average arrival time for each of the types of robots and fleet sizes is presented for the same mission - the long mission in the case for one robot.

The tests showed that both navigation methods have their disadvantages and can be perfected. In the case of the AGV robot, it took longer to achieve the final mission, while moving at the same velocity as the AMR, because the AGV is constricted to move through a pre-defined path that is never as short as the path to the goal defined by an AMR. Additionally an AMR was always able to deflect from obstacles during the simulations, which makes it the ideal solution for an individual vehicle test.

In the case of the fleet of five robots and ten robots, the difference between the average time of arrival of the first and last robot didn't prove to be significant but the arrival times of the AGV robots were higher, which not only confirmed the tests with only one robot, but also confirmed that the AMR fleet is still more time efficient than the AGV fleet.

Additionally, the order of arrival of the robots was seemingly random and this is because the vehicles did not follow a specific sequence when performing the trajectory, which resulted ultimately in not only inconsistency in the arrival time but a longer average arrival time as well.

This happened because, when simulating, the mission goals were sent simultaneously for all robots. When receiving an overload of orders at the same time, ROS computed the tasks at different times and orders, which resulted in random arrival these tasks to the TEA* algorithm and because this algorithm attributes the highest priority to the order that arrives first, the priorities were different for each test.

This explains the big difference in average time of arrival for the last robot of the fleet of ten AGVs when compared to the fleet of ten AMRs, specially because of the big number of robots that have to perform the mission, and have to follow a specific route.

Despite there being some flaws in terms of time efficiency regarding AGVs fleet, during the autonomous navigation missions, due to not being controlled by a Fleet Manager, the AMRs experienced numerous collisions, whereas the AGVs did not.

The number of collisions among AMRs increased as more robots constitute the fleet, particularly around choke points in the environment. To mitigate this issue, implementing a fleet management system could enhance coordination and traffic control among the robots. With an effective fleet management system, AMRs could improve in efficiency, reduce delays, and increase operational flexibility.

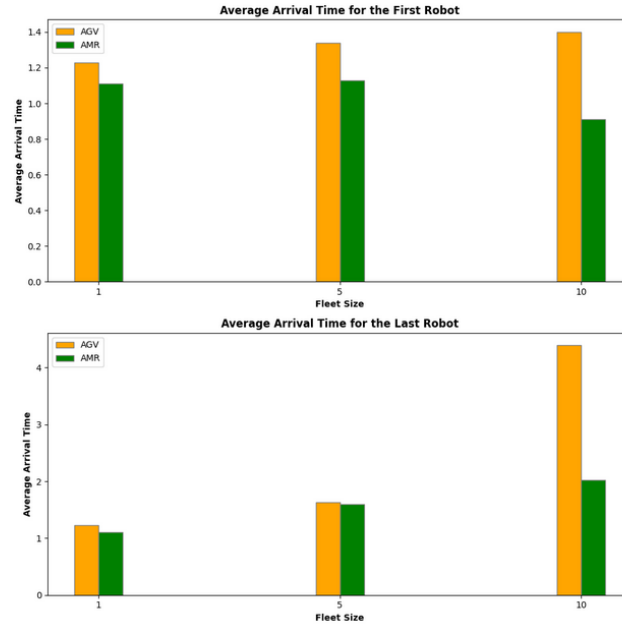


Figure 4.20: Average Arrival Time for the first and last robot for each of the missions.

4.4.2 Results with a Fleet Manager

In this section, the importance of implementing fleet management systems in fleets of AMRs was studied, and the fleet of AGVs was tested with the updated version of the TEA* algorithm in order to have a reliable discussion of the recent results.

When evaluating the simulation results for the AMR fleet of five robots, the fleet manager was necessary to improve the number of collisions. Collisions are fatal in a real workplace scenario, such as a warehouse, so it is important to lower their occurrence in order to have more efficient path-planning processes. The fleet showed better time results, a 100% success rate and no collisions during the simulations.

When looking at the results for the AGV fleet of five robots, the time of arrival of all the robots was greatly improved, but it was higher than the free navigation system by about 40 seconds, making these fleets a lot less time-efficient than AMR fleets. They show reliability for having no collisions and a 100% success rate.

In conclusion, when implementing a robust and well-programmed AMR fleet manager, the fleet behavior is highly reliable. Additionally, free navigation paths are always more time-efficient

than guided navigation paths, which makes the AMR fleet a powerful choice when comparing the two.

4.4.3 Results of the tests with a real robot

Throughout the four tests made, the robot behaved as expected, and the TEB Local Planner was confirmed to be ideal for autonomous moving in the conditions presented not only in simulation but also in the real world with a robot of its dimensions.

In the forward drive, the algorithm showed stability in maintaining the same trajectory when not encountering an obstacle and didn't show oscillations or other unexpected behavior,

In the presence of a static obstacle, the robot was able to detect and move around it with a successful outcome, returning to its global planner's intended trajectory when the obstacle was no longer in its path.

In the presence of a dynamic obstacle, the robot would replan its trajectory quickly and effectively using the local planner's algorithm. It was successful at moving around the obstacle and finding different paths to overcome it until the object was no longer in its path, never colliding with the obstacle. Subsequently, the robot continued its trajectory toward its goal.

When posed with a specific situation where a dynamic obstacle is moving toward a wall in front of the robot, at first, when there was still space to pass between the dynamic obstacle and the wall, it tried to plan a route in this direction.

Despite this, the TEB algorithm was effective at rerouting as soon as the space between the dynamic obstacle and the wall was no longer ample, choosing to move through the open hallway in a timely manner and without colliding with the dynamic obstacle or showing signs of a rerouting buffer time, such as erratic movement, oscillation or oversteering in an attempt to recalculate the route and implement new path calculations in real-time.

Chapter 5

Conclusions and Future Work

This chapter will comprise of the conclusions of the work elaborated and the goals accomplished, along with proposals of ideas to be developed further following the end of this thesis that could be valuable.

5.1 Goals Accomplishment

This thesis has provided a comprehensive comparison between fleets of AGVs and AMRs within a controlled simulated environment. Through a thorough examination of fleet management systems, the goals that were set and accomplished were the analysis of local planners for AMR navigation and the testing of varying fleet sizes for both AGVs and AMRs, before and after implementing a fleet management system. The first simulations showed AGVs outperform AMRs in terms of success rate and task completion, even though AMRs have increased time efficiency. When a fleet manager was implemented, the fleet of AMRs surpassed the fleet of AGVs in task completion, success rate, and time efficiency.

The advantages of AMRs become apparent when a fleet management system is integrated. Such a system significantly reduces the high collision rates that were first observed, allowing AMRs to demonstrate higher flexibility and superior obstacle avoidance capabilities. This indicates that AMRs currently hold an advantage in performance metrics due to the fact that task completion with a big success rate and time-efficiency is prioritized. In addition, the absence of collisions makes AMRs superior in every indicator.

It is essential to note that the scalability of both fleets remains a question, particularly given the elevated collision rates associated with AMRs, that rise with the growth of the fleet. Overall, this thesis highlights the need for ongoing exploration in the field of robotic fleets, paving the way for future advancements that could enhance the efficiency and effectiveness of automated systems in various industries.

5.2 Future Work

As mentioned before, future research should focus on exploring the scalability of these fleets, particularly of fleets of AMRs and developing more advanced fleet management solutions to optimize the performance of AMRs in larger operational settings.

The study conducted in this thesis could also be extended to heterogeneous fleets that include both Automated Guided Vehicles (AGVs) and Autonomous Mobile Robots (AMRs) operating simultaneously. This scenario would require the development of a robust fleet management system capable of effectively overseeing these heterogeneous fleets.

In addition, testing in different environments using different maps could be conducted to evaluate the performance and interactions of the fleet under different conditions. The study of the response of an AGV fleet to a static obstacle could also be valuable so that these types of fleets could become more flexible in decision-making and path-following.

By elaborating on these studies, it would ensure a bigger understanding of the challenges and requirements associated with managing a heterogeneous fleet of both autonomous and guided vehicles.

Bibliography

- [1] United Nations. *The 17 goals*. <https://sdgs.un.org/goals> [Accessed: 02/06/2024].
- [2] Günter Ullrich. *The History of Automated Guided Vehicle Systems*, pages 1–14. 12 2015. ISBN 978-3-662-44813-7. doi: 10.1007/978-3-662-44814-4_1.
- [3] Manuel Cardona, Allan Palma, and Josué Manzanares. Covid-19 pandemic impact on mobile robotics market. *2020 IEEE ANDESCON*, pages 1–4, 2020. doi: 10.1109/ANDESCON50619.2020.9272052.
- [4] J. Zhang, X. Yang, W. Wang, J. Guan, L. Ding, and V. C. Lee. Automated guided vehicles and autonomous mobile robots for recognition and tracking in civil engineering. *Automation in Construction*, 146(104699), 2023. doi: <https://doi.org/10.1016/j.autcon.2022.104699>.
- [5] Hengle Qin, Jun Xiao, Dongdong Ge, Linwei Xin, Jianjun Gao, Simai He, Haodong Hu, and John Carlsson. Jd.com: Operations research algorithms drive intelligent warehouse robots to work. *Interfaces*, 52(42-55), 2022. doi: doi:10.1287/inte.2021.1100.
- [6] mk Technology Group. *AGV or classic conveyor technology?: White Paper*, 2023. <https://www.mk-group.com/en/service/download-center/whitepaper.html> [Accessed: 09/04/2020].
- [7] Jun-Yeong Jang, Su-Jeong Yoon, and Chi-Ho Lin. Automated guided vehicle (agv) driving system using vision sensor and color code. *Electronics*, 12(1415), 2023. doi: doi:10.3390/electronics12061415.
- [8] Dayal R. Parhi Prases K. Mohanty. Controlling the motion of an autonomous mobile robot using various techniques: a review. *Journal of Advance Mechanical Engineering*, 1(24-39), 2013. doi: doi:10.7726/jame.2013.1003.
- [9] R. Yan, L. Jackson, and S. Dunnett. A study for further exploring the advantages of using multi-load automated guided vehicles. *Journal of Manufacturing Systems*, 57(19-30), 2020. doi: <https://doi.org/10.1016/j.jmsy.2020.08.005>.
- [10] Xiang Zhao and Thiagarajan Chidambareswaran. Autonomous mobile robots in manufacturing operations. *2023 IEEE 19th International Conference on Automation Science and Engineering*, (1-7), 2023. doi: doi:10.1109/CASE56687.2023.10260631.

- [11] D. Clauer, Fottner, Rauch J., E., and M. Prüglmeier. Usage of autonomous mobile robots outdoors - an axiomatic design approach. *Procedia CIRP*, 96(242-247), 2021. doi: <https://doi.org/10.1016/j.procir.2021.01.081>.
- [12] Andreas Löcklin, Maurice Artelt, Tamás Ruppert, Hannes Vietz, Nasser Jazdi, and Michael Weyrich. Trajectory prediction of moving workers for autonomous mobile robots on the shop floor. *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, (1-8), 2022. doi: [doi:10.1109/ETFA52439.2022.9921493](https://doi.org/10.1109/ETFA52439.2022.9921493).
- [13] Y. Bekishev, Z. Pisarenko, Arkadiev, and V. FMEA. Fmea model in risk analysis for the implementation of agv/amr robotic technologies into the internal supply system of enterprises. *Risks*, 11(10):172, 2023. doi: <https://www.mdpi.com/2227-9091/11/10/172>.
- [14] Bastian Solutions. *Automated Guided Carts*. <https://www.bastiansolutions.com/solutions/technology/automated-guided-vehicles/automated-guided-carts/> [Accessed: 31/07/2024].
- [15] Humberto Barberá and David Herrero-Perez. Development of a flexible agv for flexible manufacturing systems. *Industrial Robot*, 37:459–468, 2010. doi: [10.1108/01439911011063281](https://doi.org/10.1108/01439911011063281).
- [16] MWES. *Automated Assembly Line with AGVs*. <https://www.mwes.com/recent-projects/automated-assembly-line/> [Accessed: 31/07/2024].
- [17] Hai-Won Yang Sok-Yong Lee. Navigation of automated guided vehicles using magnet spot guidance method. *Robotics and Computer-Integrated Manufacturing*, 28(3):425–436, 2012. doi: [10.1016/j.rcim.2011.11.005](https://doi.org/10.1016/j.rcim.2011.11.005).
- [18] Ata Jahangir Moshayedi, Jinsong Li, and Liefia Liao. Agv robot: Mission and obstacles in design and performance. *Journal of Simulation & Analysis of Novel Technologies in Mechanical Engineering*, 12:5–18, 2019.
- [19] Sok-Yong Lee and Hai-Won Yang. Navigation of automated guided vehicles using magnet spot guidance method. *Robotics and Computer-Integrated Manufacturing*, 28(3):425–436, 2012. doi: [10.1016/j.rcim.2011.11.005](https://doi.org/10.1016/j.rcim.2011.11.005).
- [20] Tanjim Mahmud, Israt Ara, Rishita Chakma, Koushick Barua, Dilshad Islam, Mohammad Hossain, Anik Barua, and Karl Andersson. Design and implementation of an ultrasonic sensor-based obstacle avoidance system for arduino robots. *2023 International Conference on Information and Communication Technology for Sustainable Development (ICICT4SD)*, pages 264–268, 2023. doi: [10.1109/ICICT4SD59951.2023.10303550](https://doi.org/10.1109/ICICT4SD59951.2023.10303550).
- [21] André Broekman and Petrus Johannes Gräbe. A low-cost, mobile real-time kinematic geolocation service for engineering and research applications. *HardwareX*, 10, 2021. doi: [10.1016/j.ohx.2021.e00203](https://doi.org/10.1016/j.ohx.2021.e00203). URL <https://www.sciencedirect.com/science/article/pii/S2468067221000328>.

- [22] Wei Yu, Jie Li, Jing Yuan, and Xi Ji. Indoor mobile robot positioning based on uwb and low cost imu. *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, 5:1239–1246, 2021. doi: 10.1109/IAEAC50856.2021.9390754.
- [23] Zhenghao Zhang, Yonteng Zhong, Jiawei Xiang, Yongying Jiang, and Zhiling Wang. Research on the performance and improvement of uniform linear sensors array-based impact localization method under vibration conditions. *IEEE Sensors Journal*, 20(24):14932–14939, 2020. doi: 10.1109/JSEN.2020.3009981.
- [24] H. Rapp W. Hess, D. Kohler and D. Andor. Real-time loop closure in 2d lidar slam. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1271–1278, 2016. doi: 10.1109/ICRA.2016.7487258.
- [25] Raul Mur-Artal and Juan Tardos. Orb-slam2: an open-source slam system for monocular, stereo and rgb-d cameras. *IEEE Transactions on Robotics*, 2016. doi: 10.1109/TRO.2017.2705103.
- [26] Anas Charroud, Karim El Moutaouakil, Vasile Palade, Ali Yahyaouy, Uche Onyekpe, and Eyo U. Eyo. Localization and mapping for self-driving vehicles: A survey. *Machines*, 12, 2024. doi: 10.3390/machines12020118.
- [27] Shaoping Lu, Chen Xu, Ray Y. Zhong, and Lihui Wang. A rfid-enabled positioning system in automated guided vehicle for smart factories. *Journal of Manufacturing Systems*, 44: 179–190, 2017. doi: 10.1016/j.jmsy.2017.03.009.
- [28] Chuanqian Yuan, Ye Wang, and Jun Liu. Research on multi-sensor fusion-based agv positioning and navigation technology in storage environment. *Journal of Physics: Conference Series*, 2378:012052, 2022. doi: 10.1088/1742-6596/2378/1/012052.
- [29] Zhen Jia, Arjuna Balasuriya, and Subhash Challa. Sensor fusion-based visual target tracking for autonomous vehicles with the out-of-sequence measurements solution. *Robotics and Autonomous Systems*, 56(2):157–176, 2008. doi: 10.1016/j.robot.2007.05.014.
- [30] Weijie Fu, Yan Liu, and Xinming Zhang. Research on accurate motion trajectory control method of four-wheel steering agv based on stanley-pid control. *Sensors*, 23(16), 2023. doi: 10.3390/s23167219.
- [31] Jie Zhang and Xiaobo Liu-Henke. Model-based design of the vehicle dynamics control for an omnidirectional automated guided vehicle (agv). *2020 International Conference Mechatronic Systems and Materials (MSM)*, pages 1–6, 2020. doi: 10.1109/MSM49833.2020.9202248.
- [32] J. Sierra-García and Matilde Santos Peñas. Mechatronic modelling of industrial agvs: A complex system architecture. *Complexity*, 2020:1–21, 2020. doi: 10.1155/2020/6687816.

- [33] Yaowei Li, Liguozang, Tuo Shi, Tian Lv, and Fen Lin. Design and dynamic simulation analysis of a wheel-track composite chassis based on recurdyn. *World Electric Vehicle Journal*, 13(1), 2022. doi: 10.3390/wevj13010012.
- [34] Luca Bruzzone, Shahab Nodehi, and Pietro Fanghella. Tracked locomotion systems for ground mobile robots: A review. *Machines*, 10:648, 2022. doi: 10.3390/machines10080648.
- [35] Michael Rachkov, Alexey Emelyanov, and Vitaliy Kolot. Reconfigurable autonomous wheel-tracked robot. *2019 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)*, pages 1–5, 2019. doi: 10.1109/ICIEAM.2019.8742987.
- [36] Junhai Luo, Yuxin Tian, and Zhiyan Wang. Research on unmanned aerial vehicle path planning. *Drones*, 8:51, 2024. doi: 10.3390/drones8020051.
- [37] Madeline Vassaux. *The future is AGV*, 2023. <https://www.maxongroup.com/en/knowledge-and-support/blog/the-future-is-agv-17502> [Accessed: 05/06/2024].
- [38] Umar Waseem. *Understanding AMR Robots: A Comprehensive Guide*, 2023. <https://www.wevolver.com/article/understanding-amr-robots-a-comprehensive-guide> [Accessed: 12/02/2024].
- [39] Radim Hercík, Radek Byrtus, Rene Jaros, and Jiri Koziorek. Implementation of autonomous mobile robot in smartfactory. *Applied Sciences*, 12:8912, 2022. doi: 10.3390/app12178912.
- [40] Abdurrahman Yılmaz and Hakan Temeltas. A multi-stage localization framework for accurate and precise docking of autonomous mobile robots (amrs). *Robotica*, pages 1–24, 2024. doi: 10.1017/S0263574724000602.
- [41] Hung-Hsing Lin, Ching-Chih Tsai, and Jui-Cheng Hsu. Ultrasonic localization and pose tracking of an autonomous mobile robot via fuzzy adaptive extended information filtering. *IEEE Transactions on Instrumentation and Measurement*, 57(9):2024–2034, 2008. doi: 10.1109/TIM.2008.919020.
- [42] J. S. Almeida, L. B. Marinho, J. W. Mendes Souza, E. A. Assis, and P. P. Rebouças Filho. Localization system for autonomous mobile robots using machine learning methods and omnidirectional sonar. *IEEE Latin America Transactions*, 16(2):368–374, 2018. doi: 10.1109/TLA.2018.8327388.
- [43] Wei Li and Jinling Wang. Magnetic sensors for navigation applications: An overview. *Journal of Navigation*, 67(02):263–275, 2014. doi: 10.1017/S0373463313000544.
- [44] J. Včelák, P. Ripka, J. Kubík, A. Platil, and P. Kašpar. Amr navigation systems and methods of their calibration. *Sensors and Actuators A: Physical*, 123-124:122–128, 2005. doi: 10.1016/j.sna.2005.02.040.

- [45] Jie Wang and Krispin Davies. Autonomous locomotion mode transition in quadruped track-legged robots: A simulation-based analysis for step negotiation. *Simulation Modelling Practice and Theory*, 123:102893, 2024. doi: 10.1016/j.simpat.2024.102893.
- [46] S.M. Shafaei and H. Mousazadeh. Motion energy perspective of tracked locomotion system of autonomous tractor-trailer robot. *Energy*, 264:126520, 2023. doi: 10.1016/j.energy.2022.126520.
- [47] Mohamed Shelkamy, Catherine M. Elias, Dalia M. Mahfouz, and Omar M. Shehata. Comparative analysis of various optimization techniques for solving multi-robot task allocation problem. *2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, pages 538–543, 2020. doi: 10.1109/NILES50944.2020.9257967.
- [48] Alaa Khamis, Ahmed Hussein, and Ahmed Elmogy. Multi-robot task allocation: A review of the state-of-the-art. *Studies in Computational Intelligence*, 604:31–51, 2015. doi: 10.1007/978-3-319-18299-5_2.
- [49] Qiuyun Zhang, Shimiao Zhang, and Zike Zhou. Multi-robot collision avoidance : improved on voronoi-based method. *2021 2nd International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT)*, pages 52–56, 2021. doi: 10.1109/AINIT54228.2021.00019.
- [50] Mikuláš Černý, Marian Gogola, Stanislav Kubal'ák, and Ján Ondruš. Blockchain technology as a new driver in supply chain. *Transportation Research Procedia*, 55:299–306, 2021. doi: 10.1016/j.trpro.2021.06.034.
- [51] A Rosenfeld. Two adaptive communication methods for multi-robot collision avoidance. *Cambridge University Press, Robotica*, 37(5):851–867, 2019. doi: 10.1109/AINIT54228.2021.00019.
- [52] Z. Yan, N. Jouandeau, and A. Cherif. A survey and analysis of multi-robot coordination. *Int. J. Adv. Robot System*, 10(399):376–386, 2013. doi: 10.5772/57313.
- [53] J. Chen and P Dames. Distributed multi-target tracking for heterogeneous mobile sensing networks with limited field of views. *IEEE International Conference Robotics and Automation*, pages 9058–9064, 2021. doi: 10.1109/ICRA48506.2021.9561888.
- [54] Khalil Mohamed, Ayman Elshenawy Elsefy, and Hany Harb. A hybrid decentralized coordinated approach for multi-robot exploration task. *The Computer Journal*, 62, 2018. doi: 10.1093/comjnl/bxy107.
- [55] Pavee Sinruk and Yuttakm Tangmo. Heterogeneous fleet vehicle routing problem for retail construction material stores. *2017 4th International Conference on Industrial Engineering and Applications (ICIEA)*, pages 234–238, 2017. doi: 10.1109/IEA.2017.7939213.

- [56] José Manuel Vera and Andres G. Abad. Deep reinforcement learning for routing a heterogeneous fleet of vehicles. *2019 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, pages 1–6, 2019. doi: 10.1109/LA-CCI47412.2019.9037042.
- [57] George S. Oliveira, Jônata T. Carvalho, and Patricia D. M. Plentz. On the effect of heterogeneous robot fleets on smart warehouses’ order time, energy, and operating costs. *2022 XII Brazilian Symposium on Computing Systems Engineering (SBESC)*, pages 1–8, 2022. doi: 10.1109/SBESC56799.2022.9964584.
- [58] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev. Multi-heuristic a. *The International Journal of Robotics Research*, 35:224–243, 2016. doi: 10.1177/0278364915594029.
- [59] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *Association for Computing Machinery*, page 37, 1972. doi: 10.1145/1056777.1056779.
- [60] Joana Santos, Pedro Costa, Luís F. Rocha, A. Paulo Moreira, and Germano Veiga. Time enhanced a*: Towards the development of a new approach for multi-robot coordination. *2015 IEEE International Conference on Industrial Technology (ICIT)*, pages 3314–3319, 2015. doi: 10.1109/ICIT.2015.7125589.
- [61] Pedro Moura, Pedro Costa, José Lima, and Paulo Costa. A temporal optimization applied to time enhanced a*. *AIP Conference Proceedings*, 2116:220007, 2019. doi: 10.1063/1.5114225.
- [62] Diogo Matos, Pedro Costa, José Lima, and Paulo Costa. Multi agv coordination tolerant to communication failures. *Robotics*, 10:55, 2021. doi: 10.3390/robotics10020055.
- [63] D. Pereira, D. Matos, P. Rebelo, F. Ribeiro, P. Costa, and J. Lima. Multi-robot coordination for a heterogeneous fleet of robots. *ROBOT 2022. Lecture Notes in Networks and Systems*, vol 590, 2023. doi: 10.1007/978-3-031-21062-4_19.
- [64] Kaddour Messaoudi, Omar Sami Oubbati, Abderrezak Rachedi, Abderrahmane Lakas, Tahar Bendouma, and Noureddine Chaib. A survey of uav-based data collection: Challenges, solutions and future perspectives. *Journal of Network and Computer Applications*, 216: 103670, 2023. doi: 10.1016/j.jnca.2023.103670.
- [65] Yue Guan, Xiang Zhang, Danhe Chen, and Shuhui Fan. Artificial potential field-based method for multi-spacecraft loose formation control. *Journal of Physics: Conference Series*, 2746, 2024. doi: 10.1088/1742-6596/2746/1/012053.
- [66] Seyyed Mohammad Hosseini Rostami, Arun Kumar, Jin Wang, and Xiaozhu Liu. Obstacle avoidance of mobile robots using modified artificial potential field algorithm. *EURASIP*

- Journal on Wireless Communications and Networking*, 2019, 03 2019. doi: 10.1186/s13638-019-1396-2.
- [67] Wenqiang Chen, Ning Wang, Xiaofeng Liu, and Chenguang Yang. Vfh* based local path planning for mobile robot. *2019 2nd China Symposium on Cognitive Computing and Hybrid Intelligence (CCHI)*, pages 18–23, 09 2019. doi: 10.1109/CCHI.2019.8901916.
- [68] Seán McGarraghy Anthony Brabazon, Michael O’Neill. *Natural Computing Algorithms*. Springer, 2015.
- [69] S. Goss, Serge Aron, Jean-Louis Deneubourg, and Jacques Pasteels. Self-organized shortcuts in the argentine ant. *naturwissenschaften*. *Naturwissenschaften*, 76:579–581, 1989. doi: 10.1007/BF00462870.
- [70] T. Dorigo, M Stützle. *Ant Colony Optimization: Overview and Recent Advances*. Springer International Publishing, 2019.
- [71] Géza Katona, Balázs Lénárt, and Janos Juhasz. Parallel ant colony algorithm for shortest path problem. *Periodica Polytechnica Civil Engineering*, 63, 2019. doi: 10.3311/PPci.12813.
- [72] Jin Yu, Xiaoming You, and Sheng Liu. A heterogeneous guided ant colony algorithm based on space explosion and long–short memory. *Applied Soft Computing*, 113, 2021. doi: 10.1016/j.asoc.2021.107991.
- [73] Gihan Nagib and Wahied Gharieb. Path planning for a mobile robot using genetic algorithms. *Electrical, Electronic and Computer Engineering*, 2004., pages 185– 189, 2004. doi: 10.1109/ICEEC.2004.1374415.
- [74] D. E. Goldberg. *Adaptation in Natural and Artificial Systems*. Addison-Wesley Publishing Company, 1989.
- [75] Nia Saputri Utami, Achmad Jazidie, and Rusdhianto Effendi Abdul Kadier. Path planning for differential drive mobile robot to avoid static obstacles collision using modified crossover genetic algorithm. *2019 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, pages 282–287, 2019. doi: 10.1109/ISITIA.2019.8937184.
- [76] Jana Gödeke and Peter Detzner. A simulative approach to amr fleet sizing in decentralized multi-robot task allocation. *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8, 2023. doi: 10.1109/ETFA54631.2023.10275717.
- [77] OMRON Portugal. *OMRON Portugal*, 2024. <https://omron.pt/pt/home> [Accessed: 30/07/2024].
- [78] OMRON Portugal. *A OMRON anuncia o lançamento do FLOW Core 4.0 e do Fleet Manager virtual para uma implementação escalável de AMR*,

2024. <https://industrial.omron.pt/pt/news-events/news/omron-announces-flow-core-4-0-and-virtual-fleet-manager> [Accessed: 01/06/2024].
- [79] MiR. *Mobile Industrial Robots*, 2024. <https://mobile-industrial-robots.com/> [Accessed: 30/07/2024].
- [80] MiR. *DENSO*, . <https://mobile-industrial-robots.com/de/falle/denso> [Accessed: 01/06/2024].
- [81] MiR. *Fleet of MiR AMRs support Faurecia in their journey towards higher efficiency and digital transformation*, . <https://mobile-industrial-robots.com/cases/faurecia> [Accessed: 01/06/2024].
- [82] Aethon. *Aethon | Autonomous Mobile Robots*, 2024. <https://aethon.com/> [Accessed: 30/07/2024].
- [83] Aethon. *KUKA*, 2024. <https://www.kuka.com/> [Accessed: 30/07/2024].
- [84] Open Robotics. *ROS Introduction*, 2018. <https://wiki.ros.org/ROS/Introduction> [Accessed: 09/06/2024].
- [85] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Ng. Ros: an open-source robot operating system. *ICRA Workshop on Open Source Software*, 3, 2009.
- [86] Matej Dobrevski and Danijel Skočaj. Adaptive dynamic window approach for local navigation. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6930–6936, 2020. doi: 10.1109/IROS45743.2020.9340927.
- [87] Sebastian Thrun. *Simultaneous Localization and Mapping*. In *Jefferies, M.E., Yeap, WK. (eds) Robotics and Cognitive Approaches to Spatial Mapping*. Springer, Berlin, Heidelberg, 2007.
- [88] A. Vagale, R. Oucheikh, and R.T. et al. Bye. Path planning and collision avoidance for autonomous surface vehicles i: a review. *J Mar Sci Technol* 26, page 1292–1306, 2021. doi: 10.1007/s00773-020-00787-6.
- [89] Eitan Marder-Eppstein. Move base ros wiki, 2020. http://wiki.ros.org/move_base [Accessed: 22/05/2024].
- [90] Pablo Marín, Ahmed Hussein, David Martín Gómez, and Arturo de la Escalera. Global and local path planning study in a ros-based research platform for autonomous vehicles. *Journal of Advanced Transportation*, pages 1–10, 2018. doi: 10.1155/2018/6392697.
- [91] Christoph Roesmann. nav_core. [wiki.ros.org](http://wiki.ros.org/nav_core), 2020. http://wiki.ros.org/nav_core [Accessed: 25/07/2024].