Dedication

Acknowledgement

Abstract

Keywords:

Contents

List of Figures

List of Tables

Glossary of Acronyms

NN RNN FL: fuzzy Logic DWA MP RRT GA SA PSO OMPL RL CC

-
-

-

General Introduction

The latest events of the current decade have highlighted the challenges that manufacturers, suppliers, and end customers face during fluctuations in logistics and supply chain processes. Living in a VUCA world—Volatile, Uncertain, Complex, and Ambiguous—requires us to continuously adapt to changes and anticipate future events by preparing our developed environments and scaling our solutions. Simultaneously, it is crucial to maintain high standards that ensure productivity, enhance work safety, and optimize ergonomics.

In this context, the primary objective of intralogistics is to optimize, integrate, automate, and manage internal logistical flows of material and information within distribution centers, warehouses, or manufacturing plants. This subfield focuses on increasing operational efficiency by employing new technologies, such as autonomous robots.

Modernizing industrial environments through intralogistics offers significant potential for companies that adopt and adapt to it. However, convincing potential customers of the efficiency and impact of intralogistics robots presents challenges. These limitations include high training and implementation costs, changes to work routines, and the need for space and process adaptations.

A recent study from CBRE, the world's largest real estate services provider, revealed that European industrial and logistics investments increased by 16% in Q1 of 2024 compared to Q1 of 2023. Despite this, many warehouses are old, repurposed buildings that are unorganized due to the nature of their daily tasks. These brownfield warehouses are expensive to maintain and digitalize but represent ideal grounds for developing and utilizing fully autonomous systems. Unlike AGVs, autonomous vehicles possess the intelligence and capability to plan and execute their plans efficiently. They are designed to adapt to uneven terrains and unorganized working environments given the revolutionary technologies that they hold.

In this context, STILL, a KION group company, has been developing smart intralogistics solutions since its establishment more than a 100 years ago, successfully integrating automation into logistics. STILL offers a wide variety of products that cater to industries ranging from food retail to automotive manufacturing and chemical sectors. Their solutions address various customer challenges, such as reaching high shelves, order picking, palletizing, fleet management, and providing consulting services. Trusted by leading

German companies like Siemens, STILL's products and services are renowned for their reliability and efficiency. The STILL Autonomous Robots department focuses on developing and enhancing smart vehicles. These autonomous robots, with minimal cost-effective input from the warehouse environment, can perceive their surroundings, estimating their positions, efficiently planning future tasks, controlling their movements to reach destinations, executing desired actions, and making corrections if necessary. This focus on smart, autonomous vehicles demonstrates STILL's commitment to pushing the boundaries of intralogistics and automation.

In light of this, this thesis aims to contribute to the process of palletizing by optimizing a local path planning approach applied in the warehouse's stations near the shelves or spots where pallets are located for picking or in free placing areas. The developed approach seeks to plan the near-field path optimally while simultaneously avoiding obstacles. The objective is to create predictable, repeatable, and explainable vehicle behaviors, demonstrating the autonomous vehicle's ability to generate effective solutions tailored to each specific scenario. By focusing on optimal, pattern-based near-field path planning, this thesis addresses the challenge of navigating complex intralogistics environments, ensuring maximum efficiency and safety in operations. This approach not only enhances the vehicle's performance but also showcases the potential of autonomous technology in transforming modern intralogistics.

This work encloses 4 chapters:

- **Chapter 1** gives a deep insight about the host company's structure, activities and products. Then it dives into the project context and it motivations, the studied problematic, the fundamental aspects of the work, the thesis specifications and, the work methodology.
- **Chapter 2** delves into the state of the art of the work area, then goes through a review of the literature that served as a base of the thesis and gave an overview of the existing solutions. Finally, it presents milestones followed in the course of the thesis work.
- **Chapter 3** explains the development steps of the approach: it presents the mathematical aspect of splines and their implementation in robotic path planning, explains the geometric division of the stations into transition zones, discusses the studied path discrimination approaches, and finally it explores the optimization approaches for the local path planning problem.
- **Chapter 4** explicits the steps it takes to implement the developed approach in the RACK framework, test them in the RACK simulation system, then on the automated vehicle, run different test scenarios and states the obtained results.

Chapter 1
Host company and Project context

Chapter 1

Host company and Project context

Introduction

This chapter is reserved to present STILL GmbH as the host company, its organizational structure, the mother company KION group. It will then proceed to describe the range of products that the company produces. The second part is dedicated to set the project context by explaining the problem statement, the motivation behind this thesis project, and its specifications. The final part will emphasize the work methodology adopted to carry out this project.

1 Host company: STILL GmbH

This section introduces the host group and company through their activities, products, and activities

1.1 KION Group and STILL GmbH

STILL GmbH, based in Hamburg, Germany, is a leading manufacturer of intralogistics solutions with 14 locations in Germany and a global sales network spanning 246 locations. Operating under the KION Group, Europe's largest forklift truck manufacturer, STILL boasts over 100 years of experience. The company develops highly efficient, client-tailored products, serving businesses of all sizes with a wide range of forklift trucks—from manually driven forklifts to high-reach trucks and fully automated vehicles—alongside consultancy services and software solutions.

STILL prioritizes smart logistics and energy optimization while maintaining award-

winning product quality, catering to industries such as food and retail, automotive, and electronics. Employing over 9,000 people across departments like sales and marketing, research and development, production, mechatronics, and quality assurance, STILL remains at the forefront of intralogistics innovation.

KION Group is one of the global leaders in the fields of industrial trucks and supply chain solutions. It is the mother company of: Linde, Dematic Baoli, OM, Fenwick, and STILL who produce the goods and services of the group as detailed in Figure 1.1.

Present in 4 continents and hiring more than 42000 employees, KION's startegy is to ensure profitable ans sustainable growth while focusing on Automation and robotics deplyment as one of the main leaders of this growth.

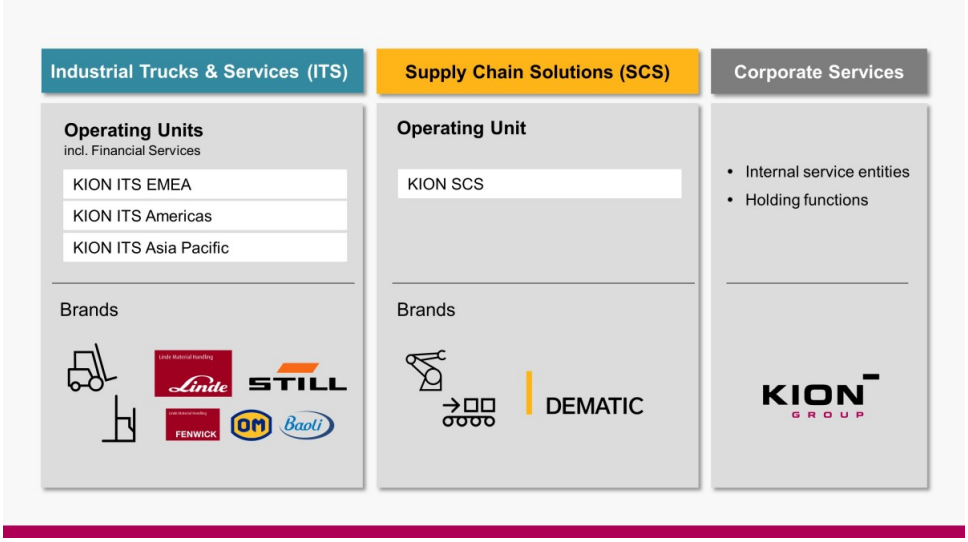

Figure 1.1: KION segment services and companies [1]

1.2 KION Management Hierarchy

The company is composed of departments managing the operations in all companies that are divided by scope of interest like R&D, Management, finances, etc.. Figure 1.2 illustrates the different areas of responsibility of the Executive Board. The Autonomous vehicles team belongs to the Mobile Automation department under CTO.

1.3 STILL Products

The 2017-established Autonomous vehicles team aims to develop fully automated solutions that leverage novel technologies to create innovative services delivered through forklift trucks. The vehicles are developed while keeping safety and high-performance as the main priorities.

iGo neo shown in Figure 1.6 is one of the main products developed by the department, it is a low level order picker transformed into the agent's autonomous assistant. Functioning in autonomous or semi-autonomous modes, it can follow the operator and their pace while avoiding obstacles and perceiving their surroundings as well as pick and place pallets in designed areas. Its added value is in preserving ergonomics of the operators by preventing heavy load carrying for long distances and decreasing the driving ascents and descents by 75% thus increasing the personal and collective performances [3].

Figure 1.2: KION Executive Board responsibilities as of 01.2024 [2]

As STILL specializes in forklift trucks, it counts many other products. Trucks are either Diesel or Gas fueled, or electric trucks that use Li-Ion batteries. Depending on the client's warehouse type, they can choose from a vast range of reach trucks Figure 1.3, hand pallet trucks Figure 1.4, double stacker trucks Figure 1.5, and Automated industrial Trucks Figure 1.6 [4].

Figure 1.3: STILL tractor

Figure 1.4: STILL hand truck

Figure 1.5: STILL rider truck

Figure 1.6: STILL iGo neo

Despite the impressive capabilities of the iGo neo and similar autonomous vehicles, the implementation of such advanced technology brings up several challenges, particularly in ensuring reliable and predictable behavior under all operating conditions. This leads to a key motivation for further investigation and improvement in the field.

2 | Graduation project Motivation and Problem Statement

2.1 Motivation

While autonomous vehicles can be highly reliable and efficient in carrying out various tasks, their behavior is not always predictable or easily explained. The output often exhibits a stochastic nature. For example, an obstacle-avoiding solution planned by the autonomous vehicle may be safe and correct but might follow an unusually shaped path.

Such stochastic behaviors can lead to a lack of trust and interest in robotized forklift trucks from a customer's perspective. This unpredictability can cause customers to question the system's repeatability, fearing that it may not perform consistently in critical situations. Moreover, the unexpected nature of these behaviors can make it difficult for operators to understand and anticipate the vehicle's actions, further reducing confidence.

Adding to these concerns, many autonomous systems, particularly in the intralogistics sector, require significant commissioning efforts before they can be implemented in a new environment and begin their service. Whether it's a required software integration, sensors installations, or measurements, these systems demand substantial time, information and financial investment—three crucial resources that we aim to optimize. Innovation in automation should involve the development of optimal solutions that are easy to commission in a new environment. These so-called "plug-and-play" solutions reduce the effort required and allow customers to start benefiting from the autonomous features with just the physical truck on-site and available information about the warehouse. the rest, is online recognition and processing. This approach significantly enhances the impact and convenience of the technology.

The autonomous vehicles department focuses on creating reliable, efficient, and explainable systems to build trust with customers, encouraging adoption of the technology. With a focus on explainable AI, the technology becomes more interpretable, helping customers understand decision-making processes and increasing their confidence in its safety and reliability.

2.2 Problem statement

Autonomous navigation is a topic where explainable intelligence can be integrated. Robotic forklifts execute missions such as transporting pallets from one place to another: picking up or dropping pallets. The mission is splitted into:

- a movement to a source
- a pickup
- a movement to the destination
- a drop

The movement to the source or the destination is the movement in figure 1.7 where *station*1 is considered the source station that the truck is headed to. This movement is planned globally and is not the scope of this work. Figure 1.7 is a section of the model of a real warehouse. In this work, the interest is around the following items of the model:

- Dark Blue outline: walls of the warehouse
- Yellow Rectancles marked as stations: Warehouse stations: Actions like picking up or dropping are performed in specific locations inside the warehouse called stations, examples of which are named *station*1, *station*2, and *station*3 on the figure 1.7. Each station is designed to support specific warehouse functions and their general mission is to facilitate and organize material handling operations. Each station contains a **shelf** that contains the pallets to be picked or dropped.
- Black rectangles: storage shelves (also called racks), where the pallets and material are stored.
- Turquoise blue lines: Global paths that connect the stations. The truck takes these routes when navigating from a station to the other while avoiding obstacles online if they occur.
- Gray rectangles are starting position of the truck around the warehouse. For instance, the truck can be placed at these positions before starting its autonomous navigation and task fulfillment.

However, the truck arrives at a distant pose from the shelf or pallet, the distance is clearly seen in figure 1.8. In addition, its forks are facing the wrong direction because driving the truck should happen like the arrow shows on figure 1.7 to avoid hitting humans or materials with the forks.

The distance is intended because the truck needs a maneuver to make its forks dock (enter) the shelf as they are the part that picks up or drops the pallet.

The path that involves this maneuver is called the link path that guides the truck to to the pallet or the drop pose (the shelf inside the station in figure 1.8).

Online linking of the robotic forklift to the goal pallet or location takes place inside the station. Online linking is mandatory because warehouse areas are subject to constant changes for instance, shelves can be shifted from their expected position and dynamic obstacles can show up, but, the vehicle needs precise recognition of the coordinates that it will reach. Precise recognition allows for correctness of the picking or dropping processes, otherwise, the task can fail. Furthermore, the vehicle has to account for the new obstacles and clutter inside of the station and avoid collisioning with them when driving to the destination shelf.

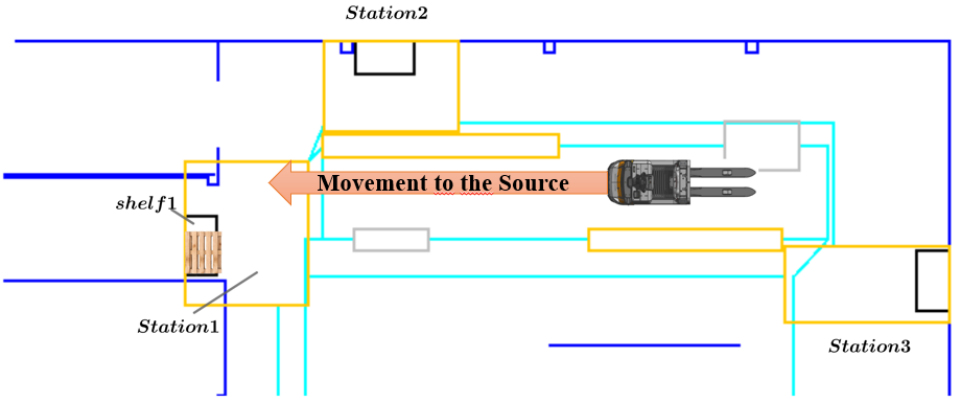

Figure 1.7: Vehicle Moving towards station 1

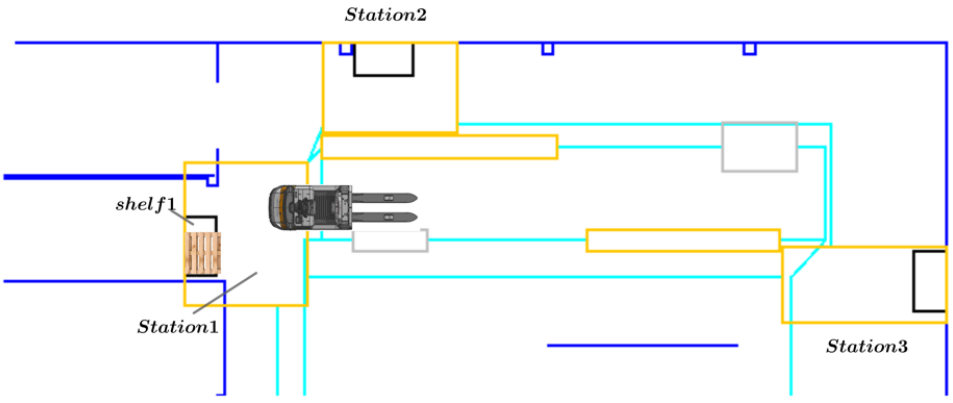

Figure 1.8: Vehicle arrives at station 1 and ready to execute the task

This thesis focuses on the subtask of an autonoumous online linking of the vehicle to shelf independantly from the task to be performed (pick up/drop). When it arrives at the station, the vehicle is fronted by these constraints:

- The vehicle's forks are not facing the destination shelf as shown in figure 1.8 but rather the opposite direction, so a driving direction change is needed.
- The vehicle is bulky in volume and mass (1200 to 4000 kg) with an overall length of 2500 to 4000 mm [5] which makes it challenging to change directions: turning on the spot or navigating in highly curved paths.
- The pallet docking process has to be very precise to avoid shifts and mistakes.

⇒ **The goal** is to autonomously and optimally transport the vehicle from the position of entering the station in figure 1.9 to the position of docking the shelf in figure 1.10 while accounting for obstacles, kinematics of the vehicle, and maximizing speed.

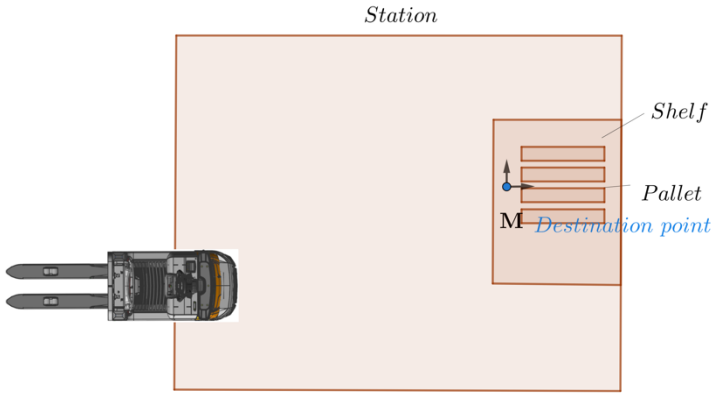

Figure 1.9: Vehicle entered the station waiting for docking the shelf

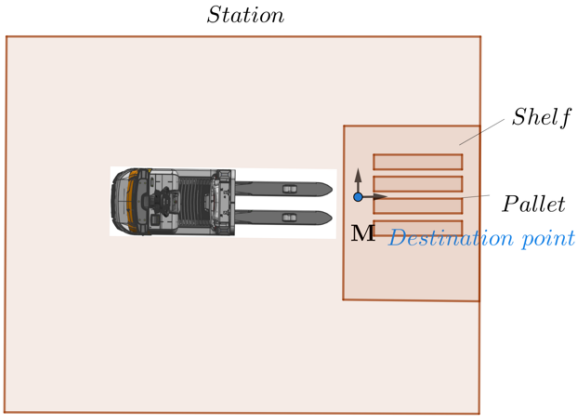

Figure 1.10: Vehicle docked the shelf

The next section focuses on the process followed during the development of the project.

3 Work Structure and Methodology

Our team adopts an Agile Scrum methodology showcased in Figure 1.11 to ensure efficient and flexible project management. Here's how we approach our work:

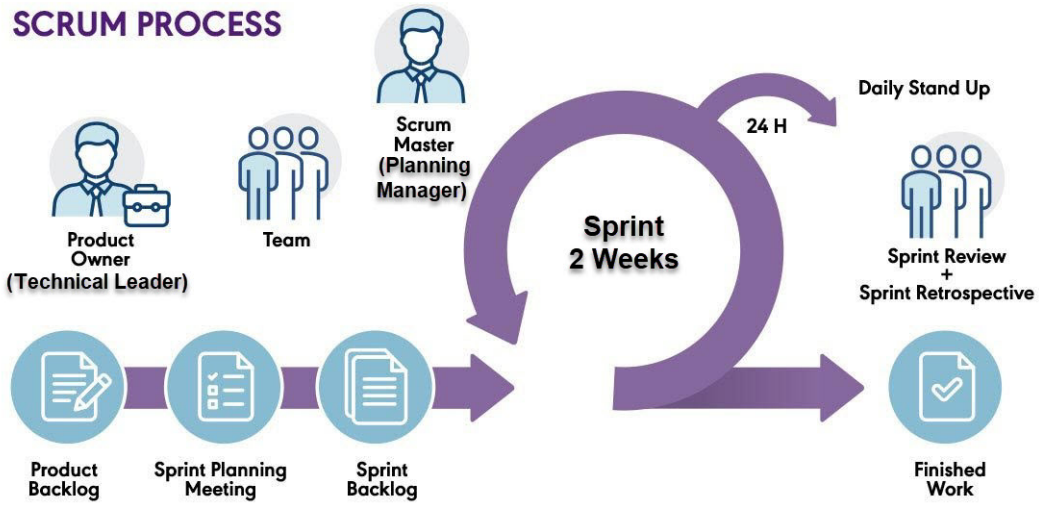

Figure 1.11: Agile Scrum Process [6]

3.1 Agile Scrum Framework

- **Jira:** We use Jira to organize and track our tasks and progress. Jira allows us to create and manage tickets, which are detailed records of tasks, bugs, or features that need attention. Each ticket is assigned to team members and tracked through its development stages until completion.
- **Sprints:** Our work is organized into 2-week sprints. Each sprint is a focused period where we aim to complete a set of predefined tasks. At the start of each sprint, we hold a meeting to review the previous sprint: every team member presents their completed tickets, and communicates the changes or blockers that appeared during the process and plan for the next sprint: decide which tasks will be tackled during the sprint. This helps us maintain a steady pace and regularly deliver increments of our project.
- **PI Planning:** Every quarter, we engage in Program Increment (PI) planning with the mobile automation teams. The PI happens in two phases: each team prepares their planning for the next 3 months, then it is discussed and tailored again in a

bigger round. This planning session helps us align our goals and strategies for the upcoming quarter. We review progress, set objectives, and coordinate with other teams to ensure that our work is aligned with broader project goals and company vision.

- **Daily Standups:** We hold 15 minutes long daily standup meetings to keep everyone on the same page. During these meetings, team members share updates on their progress, discuss any challenges they are facing, and outline their plans for the day. This practice promotes transparency, communication and quick problem-solving through collaboration.

3.2 Version Management

- **GitHub:** We use GitHub for version control and code management. GitHub allows us to collaborate on code, track changes, and manage different versions of our project. Each team member can contribute to the codebase, and we use pull requests to review and integrate new features.

3.3 Communication and Collaboration:

- **Microsoft Teams:** We use Microsoft Teams for real-time communication and collaboration. Teams provides a platform for chatting, video calls, and sharing files, facilitating smooth and efficient interactions among team members.
- **Microsoft Outlook:** Outlook is used for email communication and scheduling. It helps us manage meetings, track important messages, and coordinate tasks and deadlines.

By integrating these tools and practices, we ensure a structured yet flexible workflow, enabling us to adapt to changes, communicate effectively, and deliver high-quality results.

Chapter 2
State of the Art

Chapter 2

State of the Art: Optimal path planning for autonomous robots

Introduction

This chapter focuses on the research conducted around Optimal Near-field Path planning for robots in Intralogistics environments. It starts with an explanation of the fundamemental aspects that this work is based on like Automated Mobile Robots (AMRs), Near-field Path planning and Optimization techniques with a focus on Intralogistics. Then it focuses on the path planning approaches followed to rise to the existing challenges. Later, the topic of optimization is outlined, investigating the decision-making science that leads to the elimination of some path planning drawbacks. It concludes with a discussion around the relevance of the litertaure and ways it can be exploited for this work.

1 Intralogistics Environments

Intralogistics refers to the management, control, and optimization of internal material and information flow within a warehouse. It encompasses all physical and operational processes involved in the movement of materials and goods. This concept was introduced and defined by the German Engineering Association (VDMA). Figure 2.1 covers various aspects of intralogistics areas, including the storage, transportation, supply, Manufacturing, and disposal of production materials, as well as shipping. The scope of this work is focused around the warehouse area where storage: picking, and placing, takes place. In brownfield applications, introduced in the General Introduction, the warehouse environment is usually a crowded and cluttered atmosphere: The nature of activities like manufacturing storing install a high level of dynamics, whether it is operators or other

vehicles, and obstacles like boxes and pallets. The cluttered and highly dynamic nature of this environment makes it challenging to successfully implement robotic solutions.

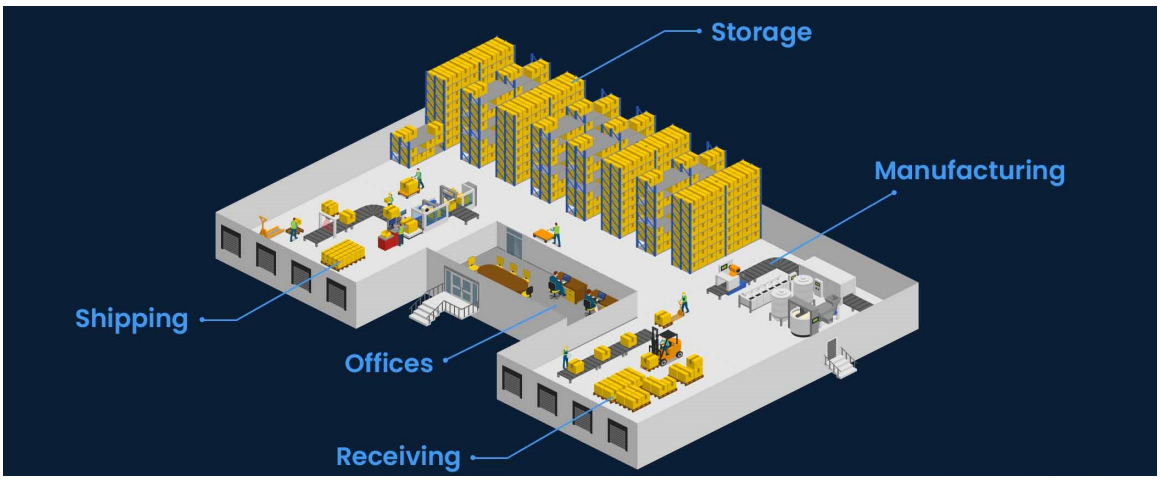

Figure 2.1: Common Intralogistics Environment Areas [46]

2 AMRs in Intralogistics Environments

Autonomous Mobile Robots (AMRs) are advanced robots designed to navigate and perform tasks independently in dynamic environments without the need for fixed infrastructure or human intervention. Equipped with sensors, cameras, and advanced software, AMRs can move around facilities like warehouses and factories, adjusting their paths based on real-time conditions. They are commonly used for material handling and transporting goods.

In the Intralogistics field, AMRs were introduced as a revolution to Automated Guided Vehicles (AGVs). First introduced in 1955, AGVs performed tasks like material handling. AGVs are managed by top-level software that handles task planning, providing the vehicles with intermediate waypoints to navigate from start to end points [7]. On the other hand, AMRs are automated in a way that makes them find the solution to unexpected problems. Figure 2.2, shows the difference in behavior between an AMR and an AGV in the case of a new obstacle. While AMRs are able to surpass the obstacle through sensors recognition and obstacle avoidance algorithms, AGVs require human intervention to eliminate the obstacle before restarting the navigation.

AMRs fit in the cluttered ares of warehouses. Their perception of the surrounding information allows for low commissioning efforts: they are able them to localize themselves, plan then navigate to the goal and reach their destinations autonomously. Their functioning does not require an expert or an engineer's intervention which limits the commissioning and utilization costs.

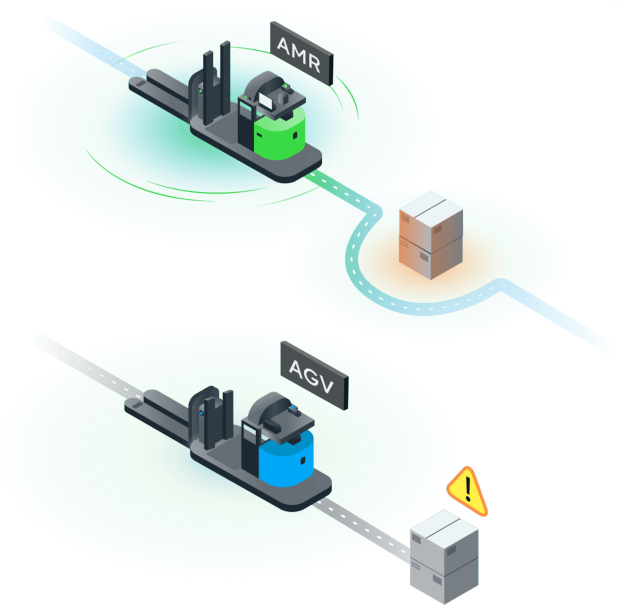

Figure 2.2: AMR and AGV behaviors at presence of an obstacle [9]

3 Path Planning

This section will dive into the path planning state of the art: presenting its promising opportuniities and current challenges, detailing the different approaches, and discussing their efficiency and compatibility with the problem statement.

3.1 Path Planning for autonomous robots: Opportunities and challenges

Path planning for mobile robots involves creating and generating efficient routes for the robot to travel from a starting position (A) to a target position (B), ensuring minimal time and travel distance while avoiding collisions with nearby objects [7]. Literature differentiates between two main types of path planning: global and local path planning. Global planning involves finding an optimal path from the start to the target position based on sensor input within a known, static environment, whereas local planning focuses on real-time path planning and obstacle avoidance, typically used online while moving to avoid dynamic obstacles [11].

In complex environments, movement and task fulfillment must be done carefully and accurately given the volume of the vehicle and the value of the handled material. With appropriate input from various sensors, such as laser scanners, cameras, and LiDAR providing robots can perceive of their surrounding environment and plan the right path accordingly. The innovation in hardware made navigation flexibility and autonomous recovery after failure possible[7]. The relevance of efficient path planning, mainly in the intralogistics sector, is derived from the constant need of optimizing material flow, productivity rates, and cost effectiveness. Path planning reduces travel time and distances when moving goods and thus , improving overall operational costs by allowing robots to take the most optimal routes when moving goods. By minimizing unnecessary detours and avoiding congested areas, robots can complete tasks more quickly, leading to faster material handling cycles. In those dynamic environments (see chapter 2, section 1), operators and employees are moving around, goods and pallets are being transported or stocked, and materials must be handled safely and carefully. Flexibility in routing and planning, enables the AMRs to always drive the optimal path based on the space settings. This flexibility offers several benefits, such as:

- Independence from Human intervention if an unexpected situation raises -AMRs do not require assistance, unike AGVs [7].
- Reduced energy consumption thanks to optimal: smooth and short paths that adhere to the vehicles' kinematics, allocated task and travel time.
- Robustness and responsiveness thanks to decentralized decison making: enables fast recovery and change of strategy after failure[7].

Efficient path planning is critical for ensuring safe and reliable handling of objects in dynamic environments. By maintaining safe distances from both static and moving obstacles, including people, and continuously detecting surroundings, robots can ensure safety at all times. An optimized path improves both the length and smoothness of travel, minimizing travel time and boosting productivity, as tasks are repeated frequently throughout the day. As a result, it enhances operational efficiency, contributing to overall system performance.

AMRs, as the name suggests, are standalone systems that must compile and process such input and generate, through algorithms, efficient paths. Path planning serves as the crucial link between the robot's sensor input and its motion control [10].

More that 60 years have elapsed since "Shaky" the first wheeled robot was running it first tests in Stanford University' labs. However, most of the robotic related topic are still being researched and improved. Dealing with all the aspects and challenges that robotics comes with can be very intricate. One of the major topics posing challenges to researchers is path planning. In a research by S. H. Tang et al. [20], the authors reviewed recent path planning approaches and challenges in dynamic unkown environments. Saftey in path planning was the main concern for 29 % of the reviewed studies. Navigating efficient paths while avoiding collisions is challenging to accomplish. Collision avoidance is tightly related to perception input through sensors, analysis and use of the data. While it may seem simple for the robots to correctfully analyze and recognize the objects around them, in reality, detailed understanding is not possible [21]. Issues are related to the accuracy of the sensors and the robustness of the algorithms. It is expected from the robot to preceive of the obstacles just like humans do, recognizing 3d shapes, dimensions, depth, direction and velocity, but from the robot's perspective, it is only able to recognize the outer surface that reflects the sensor's luminous signals. As shown in figure 2.3, the simulated robot (in gray) is surrounded by 3 obstacles (blue polygons). While the obstacles are polygon-shaped, the perception of the robot is limited to the surfaces that it can scan and does not see the hidden depth beyond these surfaces.

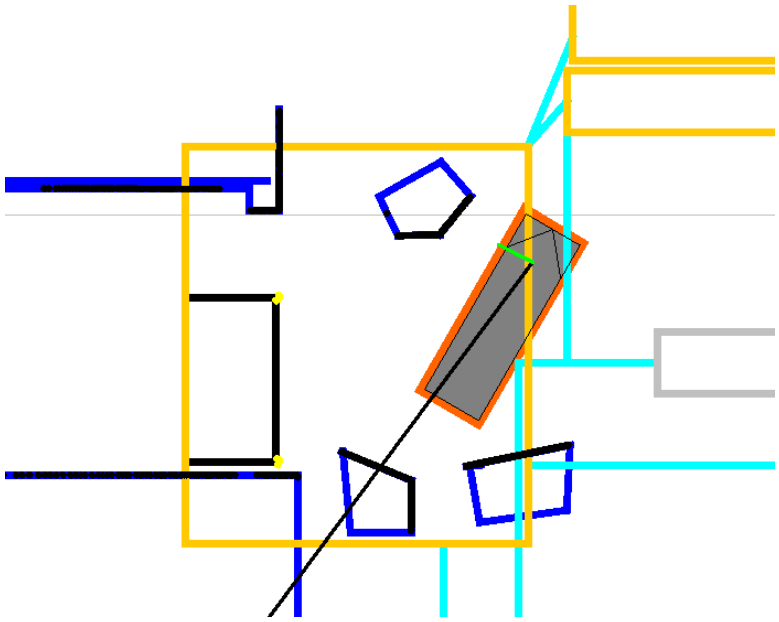

Figure 2.3: Simulation of the robot's perception of surrounding obstacles.
Yellow rectangle: Station limits
Black rectangle: Shelf limits: where to pick or to place pallets
In gray: Robot in simulation
Blue polygones: Simulated obstacles
Black lines: Perceived obstacle points

As a result, algorithms are to compensate the shadowed depth of obstacles by enforcing safety measures like keeping the vehicle at a safe distance from the obstacles and decceleratating at the proximity of static and dynamic objects to avoid collisions. In addition, this input contains noises caused by the sensors reflections of other objects or inaccuracies. This makes it challenging to interpret the input and create deliberative systems as the built algorithms have to deal with the given data in all cases and detect the inaccuracies and noises. Beyond that, it is complex to generate feasible solutions in all types of environments, some planners and algorithms risk stagnating in a local minima and not converging to the optimal routes due to the limited knowledge of the environment and the sensory limitations previously mentioned. Figure 2.4 shows a simplified example of a potential wrong decision. A vehicle encounters two possible driving paths, similar to what might happen in a racking system. Based solely on sensor data (grey shaded area), the correct path cannot be determined. Only with global path planning (green path) can a long-term route be selected (black arrow), avoiding the risk of ending up in a dead end (red arrow). Assuming the global trajectory leads to the mission goal, any local deviations from this path should be kept minimal while avoiding obstacles.

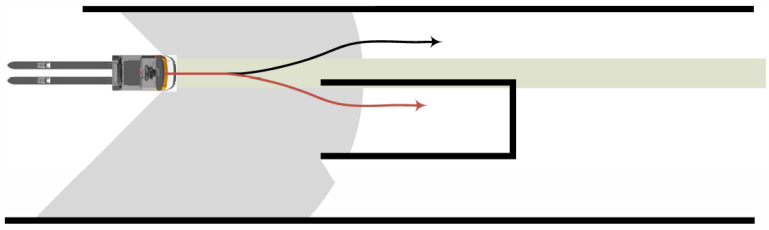

Figure 2.4: Model of a local minima problem due to limited sensor output [28]

Besides safety, computational cost is an interesting topic that researches are looking at. In a real time context, it is important to synchronize different tasks, analyze massive amounts of data generated by sensors and camera like point clouds and 2d/3d images, and compute the required decisions in a reliable and accurate way. As the complexity of the environment increases, so does the computational burden, often leading to longer processing times or the need for more powerful hardware [23].

While managing computational costs is crucial, it is equally important to ensure that the paths generated are not only computationally efficient but also smooth and short, as these factors significantly influence the robot's overall performance. Given the kinematics of a robot, the destination's location and the clutter in the environment, path planners usually render rough paths. Cusps, which are sudden and sharp direction changes, and high curvatures of the path are unusual path properties that are hard to drive, energy and time inefficient, and require continuous decelerations and accelerations. Long paths are also not favored. While they can be necessary to avoid obstacles or to create a smooth path, longer distances result in time consumption and extensive energy usage. Some path planner include post-smoothing methods that modify the paths after its creation and intervene by shortening and smoothing paths areas while obstacles. In [23], Heiden et al. present approaches to implement post-smoothing methods like B-Splines, Shortcuts, Simplify Max, and vertex optimization. They conclude that different methods deal with certain improvements areas differently. For example, While Splines are outperformed in the curvature and cusps areas, they are efficient when it comes to path-smoothning computation time.

In conclusion, tackling robot path planning requires looking at several improvement areas: computational intensivity, smoothing, and safety at the same time. Most path planning approaches are effective in the aspects that they focus on, but lack optimization in others. This further emphasizes that these challenges remain under research and are not yet fully addressed in the literature. While these challenges highlight areas needing further exploration, path planning approaches propose techniques to address them. Understanding these approaches can provide insights into potential solutions and advancements in overcoming the existing limitations.

3.2 Near-Field Path Planning

Path planning can be differentiated into Global Path Planning (GPP) and Local Path Planning (LPP) or Near-Field Path Planning (NFPP). The GPP generates a global trajectory based on static and known environment model, such as the current pose, target location and the static components of the environment like shelves and walls [28]. The LPP generates relative trajectories and allows deviation from the previously generated global trajectory if disturbances in the travel path are detected. However, LPP tend to converge into local minima as explained in the previous section .

Due to the sheer number of publications around LPP, the work of Krüger-Basjmeleh [28] clusters processes into 4 groups. Figure 2.5 illustrates 4 clusters. The classification of the methods is done by dividing them into direction, speed, and path generating methods and other methods based on AI for NFPP. All methods rely equally on immediate sensory input from the environment as the basis for future tracking strategies.

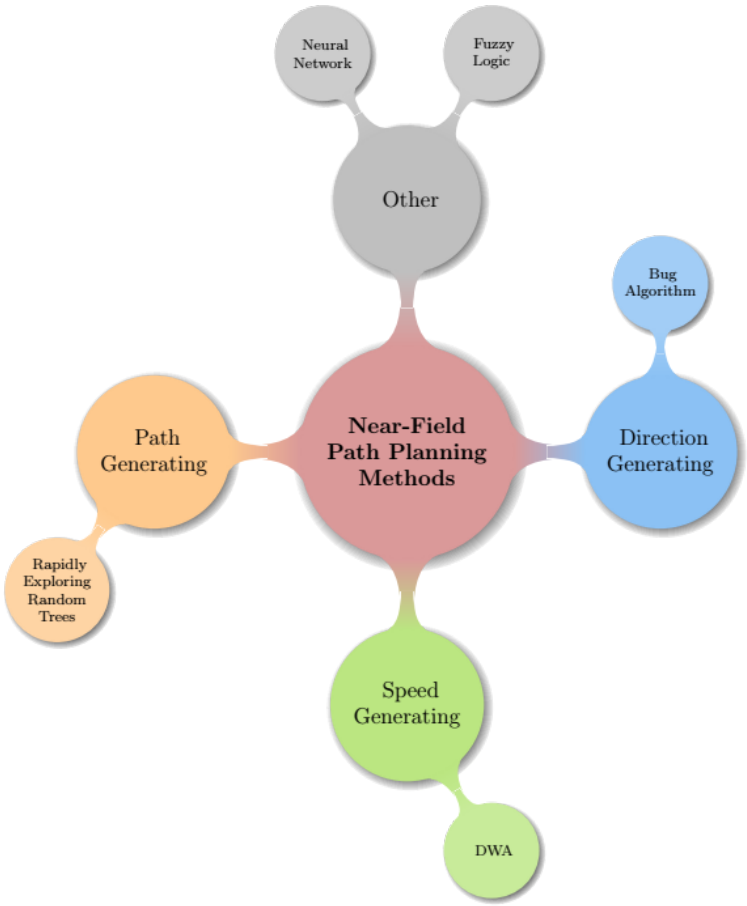

Figure 2.5: Clustering Of Near-Field Path Planning Approaches [28]

The process groups are then differentiated to identify their advantages, limitations and

suitability to the problem statement. For each cluster one or two methods are studied and explained, through which the discrimination is made.

Direction-generating methods compute feasible travel directions (directional state space) based on sensory input, which the AMR should follow in the next time step [28]. A very simple algorithm for obstacle avoidance is the Bug algorithm. In their research paper [25], Buniyamin, N. et al. propose the point to point Bug algorithm to navigate in unkown environments. Bug algorithms are based on range sensors input. As illustrated by 2.6, the robot at the starting point (Start green dot), plans to move directly to the Goal (Goal Red dot). Then it rotates scanning for obstacle points. If it encounters a sudden obstacle point (A), it navigates in its direction. The robot rotates in the target's direction (B then C) until it is able to resume its direct path to the target based on the constant search for the shortest distance from the current standpoint or find the next obstacle.

While this approach is simple from the computational point of view and the hardware used, it may not be optimal considering other metrics. First it can be significantly affected by sensory noises. Besides, its path length optimality depends on the nature of the environments and the number of obstacles and its effciency decreases if the complexity (obstacle density) of the test area increases: in figure 2.6, the path rich in vertices : start-A, A-B, B-C, and C-goal, which makes it long and lacks optimization.

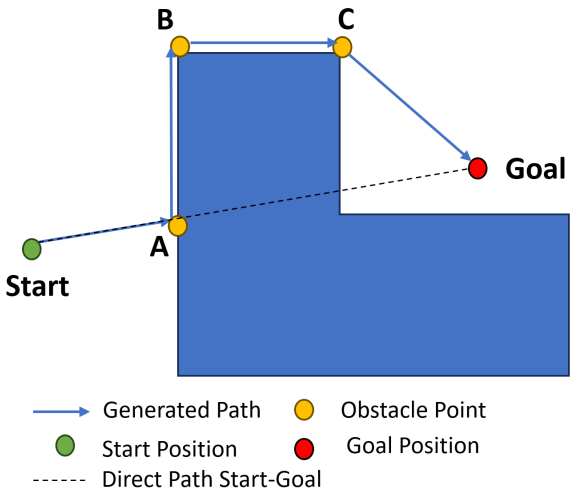

Figure 2.6: Bug Algorithm Path Solution to navigate from Start to Goal while avoiding the obstacle

Velocity-generating path planning methods are algorithms used in AMRs, to calculate both the speed and direction of movement. They predict the rotational and translational velocites creating the state space of velocities and taking into account the sensory inputs [28]. The Dynamic Window Approach (DWA) is a Velocity-generating path planning method.

In [19], Liu et al. used Djikstra ALgorithm (A path-generating Algorithm) for Global path planning and the DWA as the local path planner for sudden unkown obstacles that could appear for smart cars while following the global path. It works by evaluating different possible movements the robot could make within a short time frame and choosing the one that avoids obstacles while also moving towards the goal. The "window" refers to a limited set of possible velocities of the velocity space the robot can use based on its current speed and capabilities. Figure 2.13 details the flowchart of the DWA. The DWA algorithm starts by analyzing the car's position and base data. It then samples the speed and generates a range of possible trajectories. Each trajectory is evaluated to check if it will collide with obstacles. If a trajectory is collision-free, it's considered for optimal trajectory selection. This process continues until all possible trajectories have been evaluated or an optimal one is found. The combined algorithms were tested on 3 spaces with 3 stages of environment complexity ranging from simple to complex. However, the tests were effected with dynamic obstacles only inside the simulation and soit is subject for the reality gap problem. While these methods excel at avoiding obstacles at high speeds, they struggle with local minima. This is because they don't consider the overall structure of the free space when evaluating movements. Additionally, these velocity-generating methods have limitations in mapping the future movements of non-holonomic platforms. For example, a method might suggest a movement that is impossible for the vehicle to perform due to its non-holonomic nature [28].

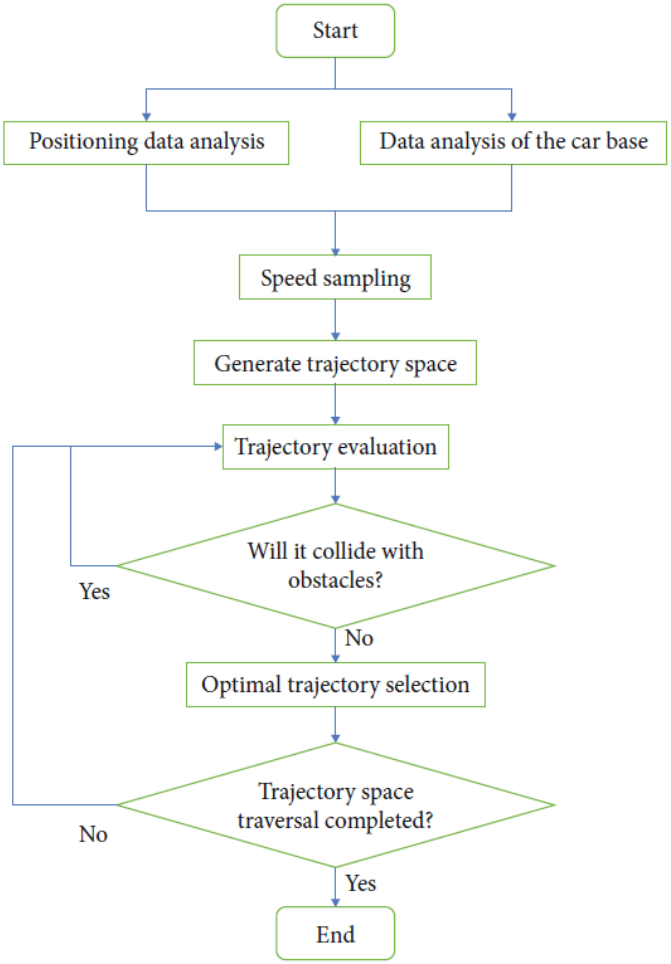

Figure 2.7: Flowchart of the DWA [19]

Path-generating path planning methods plan entire trajectories in advance, considering both the vehicle's position and its intended path over time. These planned trajectories are then combined with the previously calculated overall path. Currently, various path planning methods are used in both industry and research [28]. For example, A* Algorithm is a Path-generating Grid-based method where the environment is partitioned into a grid of discrete cells, where each cell represents a portion of the space. The algorithm then traverses these cells to identify suitable paths based on factors such as occupancy or cost. For example, in the A* algorithm, the fitness function is calculated from the start point to the reached cell, while in the Dijkstra algorithm, it's from the reached cell to the destination. While this approach offers a straightforward understanding and implementation, its computational cost can be significant due to the large number of cells to evaluate and the repetitive nature of the calculations. Additionally, grid-based methods are often constrained by the fixed directionality of the grid. The paths generated are restricted to the directions defined by the grid. For instance, if the grid is aligned along the x and y axes, the robot can only move horizontally or vertically. This can be inefficient or even impossible in environments where diagonal or more complex movements

are required. On the Other hand,

Lavalle created a method called Rapidly-exploring Random Trees (RRT), a randomized data structure used for various path planning tasks. It starts with a single node (usually the starting point) and iteratively grows a tree by adding new nodes. At each iteration, a random point is sampled from the search space. The nearest node in the tree is found, and a new node is added along the path to this random point. This process is repeated until a path is found or a maximum number of iterations is reached. RRT is particularly effective for exploring complex environments with obstacles. Figure 2.8, illustrates the test of RRT in an obstacle-rich environment (shown in blue) using 1, 4, then 32-core processors. The planned paths (branches) are shown in grey. The path calculated as optimal is shown in red and connects the start pose with the goal pose. These algorithms present advantages like handling big environments and complex obstacle situations, however, they can also be computationally intensive and demanding in complicated situations. Figure 2.8 shows the difference in the resulting optimal path in a cluttered environment where with 1 core the algorithm successfully finds a feasible path, and, as the cores double, the computed path improves its quality as the search tree expands. RRTs require collision checks for each new node, which can be computationally expensive and memory-intensive, especially in environments with numerous obstacles, heavy traffic, or complex paths.

As for artificial intelligence-based methods, also known as heuristic and metheuristic approaches, leverage the use of neural networks, machine learning and deep learning, and evolutionary algorithms. These methods learn from experience, adapt to changes, and optimize paths in complex environments. They are good at handling dynamic and unpredictable scenarios, making them suitable for tasks like autonomous driving, robot navigation, and logistics.

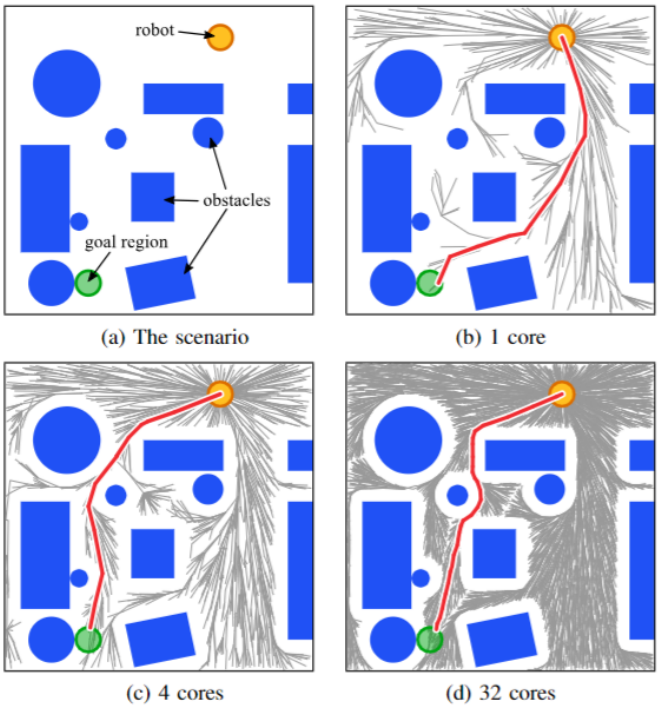

Figure 2.8: RRT ran on a non-holonomic robot using 1, 4, then 32-core processors [16]

To start with, Neural networks (NN) are inspired from the biological harmonious connection of the brain neurons that gives it the ability to process big amounts of data and generate ideas and decisions and solve problems. The NN is built in a way that enables it to get its input in a form of data and process it by learning, improving and adjusting the output to the desired results. NN are able to perform Parallel processing: the information is transmitted in two directions to the neuron in the case of Recurrent Neural Networks (RNN) that allows for learning fron current and past inputs to the NN. This approach improves the computation time and overall performance. The NN is then able to process complex solutions and create paths for difficult environments. It is well suited for unpredictable situations as it is built to adapt to the available input and the desired output. However, it presents a practicality challenge. Althugh the performance an results can be impressive, yet, in a real-time context, it is hard to rely on solutions that require extensive computational efforts and need long durations to process solutions. In addition, the number of neurons and layers have to be scalable and depends closely on the level of complexity that the vehicle is to deal with [12].

Fuzzy Logic (FL), on the other hand, is a way of thinking that mimics how humans make decisions, especially when things are unclear or uncertain. Instead of working with exact numbers, it uses "fuzzy" terms like "good", "average" or "bad" to make decisions. Input numbers are clustered following the fuzzy sets or intervals and assigned a value using membership functions. Later the values are interfered and defuzzified to genearte the ouput as a command value. In robotics and path planning, FL helps robots navigate

by allowing them to handle uncertain situations, like avoiding obstacles or finding the best route, even when the environment is not fully known. Instead of needing exact data, the robot can employ fuzzy terms like "close" or "far" to understand its surroundings. For example, if an obstacle is "somewhat close," the robot can smoothly adjust its path to avoid it. FL also helps the robot choose the best route by weighing various factors like distance and safety, even when the information is not perfect. This makes the robot better at handling unpredictable environments and making flexible and optimal decisions. However, one disadvantage is that it can be tricky to create the right rules for the robot to follow, and the system can become complicated as more rules are added. It may present a scalability problem because in unpredictable and dynamic environments it is not simple to decide about fixed fuzzy sets that would be practical in all of the cases [12].

While Neural Networks and Fuzzy Logic offer powerful approaches for handling uncertainty and complex decision-making, another key area in artificial intelligence for path planning is the use of meta-heuristic algorithms. Meta-Heuristoc algorithms are inspired from biological and natural processes for evolution and survival. Unlike NN and FL, that rely on data as input to generate decisions, meta-heuristic algorithms explore the solution space by evolving random solutions for the problem and optimizing them by rounds until a stopping criterion or set of criteria is satisfied (see more in section ...: Optimization Algorithms). Approaches like Genetic Algorithms (GA) have been used for path planning by Ahmed Elshamli et al. [17]. The The robusteness of their solution is its adaption to dynamic environments. They evolve their GA using variable size chromosomes, where each node represents a waypoint of the path, then they measure the quality of each path using an evaluation function. A modified Genetic Approach is then applied to each population: Crossover, mutation, **Repair** infeasible paths, **Shorten**, then **Smoothen** feasible paths while dynamically checking for new obstacles. The approach is tested in static and dynamic environments and has achieved proven efficiency when it comes to local optima challenges and survivig the best elements of the population. While the use of the GA itself is robust and successful, it can be challenging to recreate the algorithm and to add the improvements. In addition, the tests the ran are limited to a simple simulation with unrealistic situations and obstacle setting as displayed in figure 2.12.

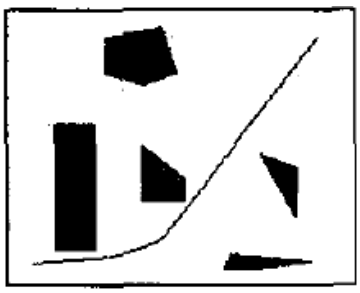

Figure 2.9: Simple obstacle environment

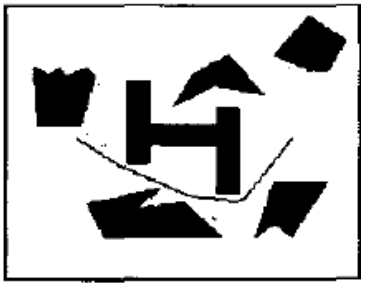

Figure 2.10: Intermediate obstacle environment

Figure 2.11: Complex obstacle environment

Figure 2.12: GA Test scenarios [17]

3.2.1 Local Path Planning

Local path planning has as the role to make real-time decisions about how to move safely and efficiently in their surroundings. Unlike global path planning, which focuses on finding the best overall route from a starting point to a destination based on a static map, local path planning deals with navigating the environment directly around the robot while considering real-time data input from sensors without prior knowledge about the surroundings[18]. This involves quickly responding to obstacles that appear or changes in the terrain, ensuring the robot can continue moving without collisions. Unlike Global Path planning, it functions without prior localization and mapping of the surrounding environment. Local path planning is crucial for the safe and effective operation of robots, especially in dynamic and unkown environments. For example, in busy spaces with moving people or vehicles, a robot needs to be able to make quick adjustments to avoid accidents. This type of planning allows the robot to react immediately to new information, making it essential for applications where unpredictability and quick responses are vital, such as in autonomous vehicles or service robots in warehouses and factories.

Common Techniques involve Reactive methods which are approaches in local path planning that focus on real-time obstacle avoidance and online adjustements on the path that are processed while the robot is navigating. Techniques like the Potential Fields, Dynamic Window Approach (DWA), and Bug Algorithms are commonly used. In their research paper [25], Buniyamin, N. et al. propose the point to point Bug algorithm to navigate in unkown environments. Bug algorithms are based on range sensors input. The robot at the starting point, plans to move directly to the target. Then it rotates scanning for obstacle points. If it encounters a sudden point, it navigates in its direction. The robot rotates in the target's direction until it is able to resume its direct path to the target based on the constant search for the shortest distance from the current standpoint or find the next obstacle. While this approach is simple from the computational point of view and the hardware used, it may not be optimal considering other metrics. Its path length optimality depends on the nature of the environments and its effciency decreases if the complexity of the test area increases. Local path planning approaches are effective in navigating through cluttered environments, but when it comes to long-range planning, a hybrid approach that integrates both local and global path planning strategies can provide more robust and efficient results.

3.2.2 Hybrid Path Planning

In recent years, path planning approaches that combine both global and local techniques have gained popularity. By integrating global and local path planning approaches, robots and autonomous vehicles can take advantage of the strengths of both methods. This hybrid approach is a promising area of research, with the potential to benefit a wide range of applications. Hybrid path planning approaches are becoming increasingly popular due to their ability to handle complex environments and dynamic obstacles.

In [19], Liu et al. used Djikstra ALgorithm for Global path planning and the DWA as the local path planner for sudden unkown obstacles that could appear for smart cars while following the global path. It works by evaluating different possible movements the robot could make within a short time frame and choosing the one that avoids obstacles while also moving towards the goal. The "window" refers to a limited set of possible velocities of the velocity space the robot can use based on its current speed and capabilities. Figure 2.13 details the flowchart of the DWA. The combined algorithms were tested on 3 spaces with 3 stages of environment complexity ranging from simple to complex. The tests showed that the hybrid path planning approach was efficient for navigating the robot and avoiding collisions. However, the tests were effected with dynamic obstacles only inside the simulation.

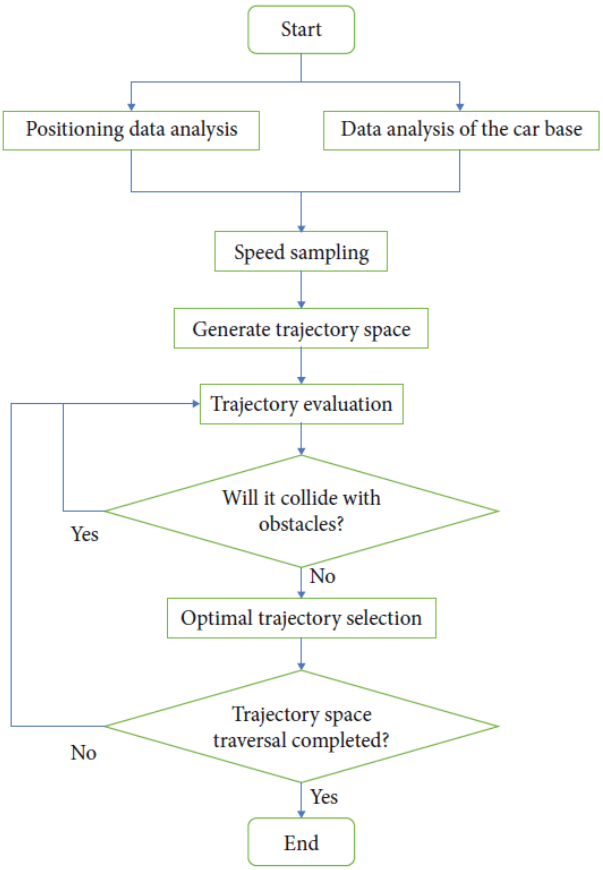

Figure 2.13: Flowchart of the DWA [19]

Sensor-based approaches, on the other hand, rely on data collected by the robot's sensors, such as LIDAR, sonar, or cameras, to make decisions about where to go next. These sensors help the robot create a map of its surroundings, which can then be used to identify obstacles and safe paths. Occupancy grids and Point cloud processing are techniques used to interpret the sensor data and guide the robot's movements. This approach allows the robot to have a detailed understanding of its immediate environment, making it more capable of avoiding obstacles and navigating complex spaces.

3.3 Discussion

While traditional path planning methods have proven effective across various applications and usecases, they often face challenges when applied to highly structured, repetitive environments like those found in industrial automation. To overcome these limitations, the research in the previous years became rather oriented to investigating and using heuristic approaches. In [26], Masehian et al. conducted a chronological review about the followe approaches in Motion Planning (MP). The results testify that in 30 years, The application of Heuristic approaches in MP went from 0% to 54% from 1977 to 2007 as shown on Figure 2.14.

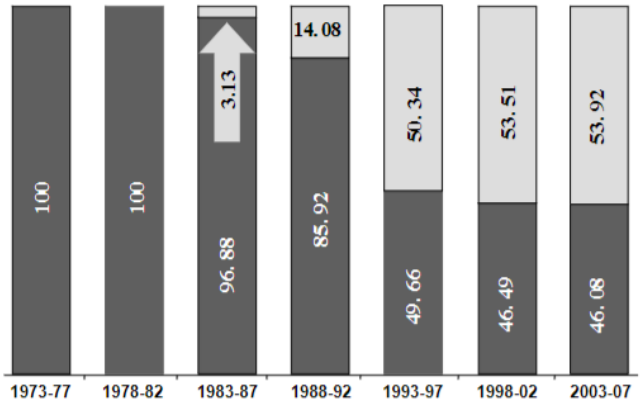

Figure 2.14: Application of classic and heuristic approaches in MP [26]
Dark gray: Classic approaches
Light gray: Heuristic approaches

The classic algorithms have the drawback of traps in local minima and increased complexity in sophisticated environments. In practice, the conditions where the robot operates are cluttered and complex than we can imagine which makes the classic approaches impractical. Consequently, it is necessary to tackle these issues in a different way. Although probabilistic methods proved high computational capabilities, scientists tend to combine 2 algorithms to benefit from the classic and heuristic advantages simultaneously. In [27], Chen et al. delve into a hybrid approach for online motion planning. Their approach is based on space exploration and heuristic search to determine optimal path linking a start point to an end goal. Tested on both low and high speed scenarios, the algorithm ensured a low planning time: it outperformed RRT and had similar results in terms of planning time to Hybrid A*.

To draw to a close, even though researches confirm that the heuristic approach prevails the classic one: [20] and [25], these approaches are not without their limitations. Heuristic methods, while good at navigating complex environments and avoiding local minima, sometimes fall short in finding the best possible solutions, especially in high-dimensional spaces. In contrast, classical methods, though computationally expensive, provide a more systematic and theoretically solid approach to path planning. To address these trade-offs, there is growing interest in hybrid methods that combine the strengths of both approaches. These hybrids use the precision and reliability of classical methods along with the flexibility and speed of heuristics. By doing so, they reduce the weaknesses of each approach, offering a more balanced and effective solution for path planning in complex environments. As a result, hybrid methods hold promise for future research, potentially leading to more efficient and dependable path planning algorithms.

4 Spline based Paths

This section will closely examine splines, their different types, their application in robotics and the importance of their integration and detail the challenges and limitations related to the use of Splines for robotic motion planning, and conclude by a discussion around their relevance to this work.

4.1 Definitions and Basic Concepts

In mathematics and computer graphics, a **curve** is a continuous and smooth flowing line without sharp angles. It can be defined parametrically or implicitly and represents a path that can be traced by a moving point. A curve can be defined using a parameter u, where $u$varies over an interval, and the coordinates of the curve are given by functions of:

$$C(u) = (x(u), y(u)) \quad \text{given} \quad a \leq u \leq b \tag{2.1}$$

The circle is an example of a curve where:

$$x(u) = \cos(u) \tag{2.2}$$

$$y(u) = \sin(u) \quad \text{given} \quad 0 \leq u \leq \frac{\pi}{2}[29] \tag{2.3}$$

whereas, the implicit form of the circle would be:

$$(x-h)^2 + (y-k)^2 = r^2 \tag{2.4}$$

where (h, k) is the center and r is the radius.

Parametric curves have a clear direction (from C(a) to C(b)), which implicit curves lack. This makes it simpler to create ordered sequences of points along a parametric curve. Additionally, the parametric form is more intuitive for designing and modeling shapes on a computer, as the coefficients in many parametric functions (such as Bezier and B-Splines) carry geometric significance. This results in user-friendly design techniques and algorithms that are numerically stable [28]. However, using one polynome-curves is inadequate as it is not possible to represent complex shapes, certain curve changes, and fitting the needed points. The solution is to use piecewise polynomial curves of degree $(n-1)$ for example if we need to integrate n data points [29].

Polynomials are a commonly used type of function in robotics because they can be easily differentiated and are less prone to numerical rounding errors (known as floating point errors). Although they cannot represent every geometric curve, polynomials can usually approximate them with sufficient accuracy.

Using classical polynomial functions or their derivatives in Bézier form is mathematically equivalent. This means that any curve described using standard polynomials can be transformed into a Bézier curve, and vice versa. However, when it comes to geometric modeling—especially in applications like computer graphics or robotics—the Bézier form is often preferred. This preference is due to the fact that the coefficients in the standard polynomial form (also known as the power basis form) don't offer much information about the curve's actual shape, making it harder to intuitively control and adjust the curve. In contrast, the coefficients in the Bézier form -the control points- directly influence the shape of the curve, in a visually meaningful way, making it easier to understand and manipulate as shown in equation 2.5 [28].

$$C(u) = \sum_{i=0}^{n} B_{i,n}(u)\mathbf{P}_i \quad \text{with } 0 \leq u \leq 1 \tag{2.5}$$

The basis function $B_{i,n}(u)$ is called the Bernstein polynomial of degree n and follows the equation

$$B_{i,n}(u) = \frac{n!}{i!(n-i)!} \cdot u^i \cdot (1-u)^{n-i} \tag{2.6}$$

In this context, the Bernstein polynomial is a fundamental component. The basis function $B_{i,n}(u)$ is used to determine how much influence each control point $\mathbf{P}_i$ has on the shape of the Bézier curve.

The geometric coefficients $\mathbf{P}_i$ are known as control points, and they define the shape of the polynomial curve. In many applications, such as mapping trajectories, a large number of control points is needed [28]. Furthermore, in the Bézier form, the continuity of the curve depends on the placement of control points. This means if you want to change the shape of part of the curve while keeping the rest smooth, you can't easily do so because changing one part affects the whole curve [29].

Instead, a more effective curve representation can be expressed as:

$$C(u) = \sum_{i=0}^{n} N_i(u)\mathbf{P}_i \tag{2.7}$$

where:

- $\mathbf{P}_i$: the control points
- $N_i(u)$: the piecewise polynomial functions

The continuity of the curve is determined by these basis functions, allowing for flexible modification of control points without affecting the curve's smoothness [29].

The basis functions for B-splines are defined recursively. For a given degree p and a set of non-decreasing knot values $\{u_i\}$, the basis functions $N_{i,p}(u)$ can be defined as:

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \leq u < u_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \tag{2.9}$$

With:

- u is the parameter along the curve.
- u_i are the knot values, which divide the parameter range into intervals.
- $N_{i,p}(u)$ are the B-spline basis functions of degree p.

In figure 2.15 stands an example of a NURBS spline generated through setting random control points using th NURBS generator [W1]

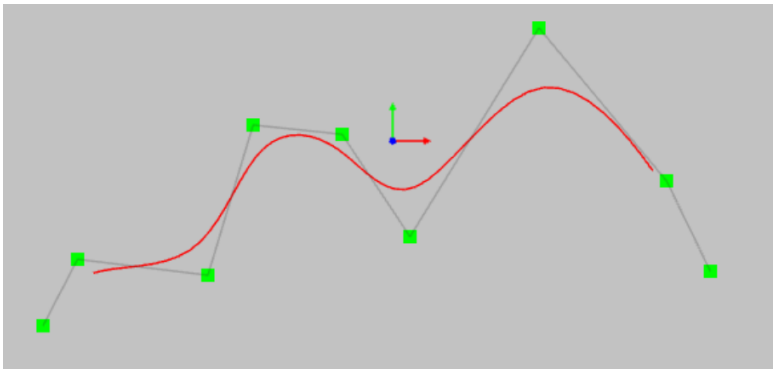

Figure 2.15: NURBS spline example.
Green: Control points
Red: NURBS spline

Understanding these principles is essential as we move forward to explore the applications of splines in robotics. In the next section, we will examine how these spline concepts are applied to solve real-world problems in robotic path planning and motion control.

4.2 Applications of Splines in Robotics

4.2.1 Trajectory Planning

Robots like AMRs in the intralogistics sector usually have mission to carry heavy loads. This property makes it risky to operate abrupt motion changes like stopping or turning. These vehicles' motion planning requires precise control and detailed alignment to the kinematic constraints that they present [30]. Given the mathematical nature of B-Splines, they ensure continuity of the first and second derivatives. This continuity translates to smooth transitions of the resulting velocities and accelerations of the path that the robot will follow. By having continuous velocity and acceleration, we make sure that the robot achieves smooth transitions and speed changes. Furthermore, tracking, precise following and correct control interventions are guaranteed allowing for precise navigation [30].

Although simple, the connected waypoints approach comes with drawbacks like sub-optimal travel time, unoptimized long paths, and mechanical wear because of abrupt direction changes unlike curved paths [30]. The morphological difference in both paths is distinguished in figure 2.16

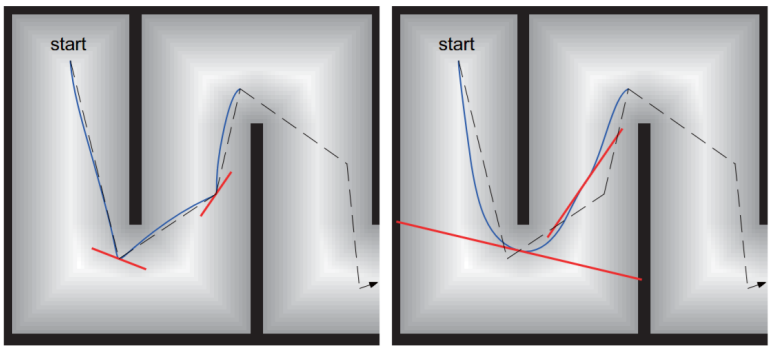

Figure 2.16: Morphological difference between a path connecting waypoints (left) and a curved path (right) [30]

Multiple approaches have been developed to introduce smooth transitions and turns into robot trajectories. One such method involves using Clothoid curves, which connect straight path segments with circular arcs to avoid abrupt changes in direction [31]. Originally introduced for designing highways and roads to ensure safe and comfortable driving for humans, Clothoids offer a gradual change in curvature.

However, when applied to robotics, Clothoid curves present certain drawbacks. These include the potential for discontinuities in curvature at the transitions, which can lead to jerky motions in robotic paths. Additionally, Clothoid-based paths can result in sub-optimal path lengths, as they may not be the shortest possible routes. Moreover, while circular arcs with constant radii are used in this approach, longer arcs lead to larger

curvatures, which can be impractical for mobile robots that require tighter turns and more precise control [31].

Migrating from a path with sharp angles to one with smooth curves can be effectively accomplished by using splines. The interpolation of the control points of the original path creates a smooth, continuous curve. By incorporating a convex hull around these control points, we ensure that the path does not deviate excessively from the desired points, thus avoiding potential collisions with obstacles [33]. The convex hull acts as a boundary that keeps the spline inside the limits of the control points that prametrize it (see figure 2.17) and keeps the robot from straying too far, ensuring a safe and accurate trajectory [29].

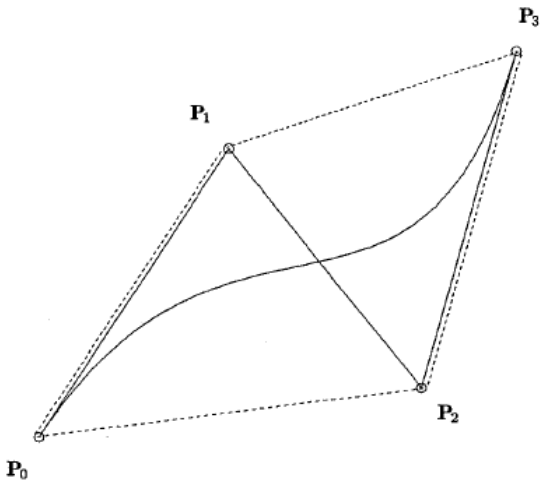

Figure 2.17: Convex hull contained Spline [29]

Splines are advantageous because they can accommodate various constraints such as curvature limits and minimum turning radii. These constraints are crucial in many applications, including robotics and vehicle navigation, where maintaining a smooth path is essential for stability and control. By adjusting the spline parameters, it is possible to design paths that meet specific requirements, ensuring that the resulting trajectory is not only smooth but also feasible given the physical and operational constraints [30].

In scenarios involving obstacle avoidance, splines can be blended or connected to create a seamless path while circumventing obstacles. This blending technique allows for the integration of different spline segments into a continuous trajectory, ensuring that the path remains clear of obstacles. By carefully adjusting the blend points and ensuring smooth transitions between splines, it is possible to navigate complex environments safely and efficiently. This approach combines the advantages of smooth path planning with effective obstacle avoidance.

A notable case study of splines integration in solving path planning problems, is a research conducted by B. Lau et al. [30] that aims to develop a time optimal solution while

considering the kinodynamic properties of a mobile robot. They used a global planner to generate straight-line paths and minimize time of travel. Then, they integrated splines to join the resulting path segments in a smooth and continuous way and replan in case of unprecedented collisions. The results show that our motion planning system works well in both real and simulated settings. It created smooth and accurate paths, with an average positional error of about 1 cm and a velocity error of less than 2 cm/s. The optimization process improved travel time by 31% compared to initial paths. The system also handled dynamic environments, like crowded trade shows, and adjusted smoothly when localization errors occurred by updating the path as needed. Overall, it performed reliably, navigating precisely and adapting effectively to changes and errors.

4.3 Discussion

In general, Splines are an effective tool that centralizes advantages related to path smoothness, simplicity of constraining the path according to the kinodynamic properties and mechanical limitations of the mobile robot like limiting the curvature, and allowing for safe and smooth speed and direction changes. These properties accommodate simple manipulation of splines for path planning and tracking of path properties. As an illustration, it is possible to measure the spline's features like curvature or approximate its length. Figure 2.18, shows an example plot of the curvatures of 3 random splines.

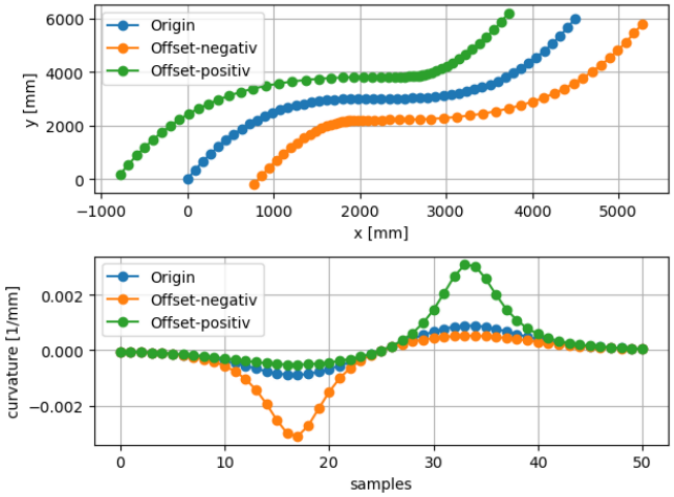

Figure 2.18: Curvature of 3 S-shaped splines

In the next section, we discuss the ways we can use such information about splines to measure paths quality.

5 | Evaluating Path Efficiency: Key Metrics

When planning paths for robots in the Intralogistics or in any field in general, it is important to have the ability to quantify the quality of the resulting path. There exists different ways to judge the efficiency of the solution and to which extent it accomodates the goal and optimizes the gaps that we are willing to improve. This section studies the strategies that the literature followed while quantifying path quality in the robotics field. It will look at the different indicators, why, and in which case they were used.

5.1 Core Metrics for Path Evaluation

The key metrics that researchers focus on and try to improve can vary depending on the specific areas of optimization they are working on. For example, if the goal is to make a robot move faster, then time and speed might be the main metrics they study. On the other hand, if the focus is on making the robot navigate more safely, then the proximity to obstacles and collision avoidance would become the primary metrics. Therefore, the choice of metrics is closely linked to the particular goals that researchers aim to achieve in their optimization efforts.

According to Tang, S. H. et al. [20], from 2011 to 2015, the second most common topic in the path planning research scope was path length. Path length is a key metric that helps us understand other important factors like travel time and energy consumption. By measuring the length of a path, we can get insights into how long it will take to travel and how much energy the robot might use. In other words, path length often reflects the overall efficiency and effectiveness of the route. Heiden, E et al. [23], employed path length as of the metrics used to evaluate the length of the paths generated by the algorithms that they compared against each other. Their planners were configured to minimize path length, therefore, they measured the path lengths that were computed in 4 scenarios with applied to 17 path planners and 4 steering algorithms. They used path length, along other metrics, to quantify the quality of paths after palnning and smoothing. Besides, they compared the algorithms using the above-mentioned metrics in order to classify them under the benchmark and to justify the choice of the best algorithm.

In addition to path length, Path smoothness is crucial for robotics in general and especially in the intralogistics sector because it directly affects the efficiency and safety of robot operations. Smooth paths reduce the need for sudden stops or sharp turns, which helps robots maintain a steady speed and avoid unnecessary wear on their components. In intralogistics, where robots often work in busy environments with other machines and human workers, smooth paths also minimize the risk of accidents and product damage. Overall, ensuring path smoothness helps robots operate more reliably and effectively, leading to faster and more accurate material handling. Path smoothness reflects less

cusps -sharp turns- and generally less curvature which allows for smooth driving and smaller scale steering. Curvature of a circle at a point $x(t)$ is defined according to Gary D. Knott, in his book "Interpolating Cubic Splines" [34], as the reciprocal of the radius at the same point as shown in equation 2.10.

$$\kappa = \frac{1}{R} \tag{2.10}$$

For a cubic spline $y = f(x)$, the curvature κ at a point x is given by:

$$\kappa = \frac{|f''(x)|}{\left(1 + (f'(x))^2\right)^{3/2}} \tag{2.11}$$

The unsigned curvature indicating whether the curve is concave or convexe is given by:

$$\kappa = \frac{f''(x)}{\left(1 + (f'(x))^2\right)^{3/2}} \tag{2.12}$$

Heiden, E et al. [23], also relied on curvature to determine path quality. They used it along path length and tracked it for the different scenarios and algorithms to measure their effectiveness. Liu, S. et al. integrated path smoothness metric to quantify path quality for their benchmark to compare motion planning algorithms[35]. Although the above-mentioned sources used the metrics to evaluate their optimized solutions, they did not mention the techniques followed to measure the metrics.

Path clearance is another significant metric used to evaluate how well a robot can safely navigate its environment. It measures the distance between the robot's position along the path and any obstacles it might encounter. A larger path clearance means the robot has more space around it, reducing the risk of collisions and making the path safer. This metric is especially useful when optimizing paths, as it helps ensure that the robot can move smoothly without bumping into objects or other robots. Elshamli, A. et al. [17] have used path clearance as one of the indicators of path feasibility. Clearance is measured as follows in 2.13:

$$\sum_{i=1}^{n-2} \exp\left(a \cdot (g_i - \tau)\right) \tag{2.13}$$

where g_i is the shortest distance from the i^{th} discrete position of the robot to the surrounding obstacles, and τ is the desired clearance distance that guarantees safety. By focusing on path clearance, we can design routes that not only get the robot from one point to another efficiently but also do so in a way that minimizes risks and improves overall safety.

5.2 Combinations of single metrics

Some applications focus solely on minimizing path length. They use this single value as the metric for evaluating and distinguishing path solutions or for optimization. However, other applications require the combination of several metrics. This combination invites us to look at possible ways to blend the measured value into one significant value that reflects path quality. Elshamli, A. et al. [17] used a weighted sum of the Distance between two nodes, or two waypoints, Path smoothness, and clearance as given by 2.14.

$$\text{eval}(p) = w_d \cdot \text{dist}(p) + w_s \cdot \text{smooth}(p) + w_c \cdot \text{clear}(p) \tag{2.14}$$

The outcome of this evaluation function of each path is used as the fitness function for that path. The fitness of each path serves later the evolution of the genetic algorithm.

Zhang B et al. [36], deployed a nonlinear optimization problem that integrates path length and curvature variation over a spline-based path. The problem is constrained by the curvature limits, the parametrized location of the spline's control points, and the distance to obstacles.

Combining metrics for path evaluation should be done with a simple approach. It is advisable to limit the number of combined metrics to 2 or 3. Using more metrics can make the optimization process complex and slow, as the optimizer has to handle more variables. Additionally, with too many metrics, it becomes challenging to understand how each one influences the results. By keeping the metrics to a manageable number, we ensure that the optimization remains efficient and the impact of each metric is clear and easy to interpret. Using a more metrics like computational time, number of cusps or direction changes, solvability percentage or predicted speed or travel time can be efficient for evaluating path planning algorithms and benchmarking some usecases as in [23].

6 Heuristic Approaches to Path Optimization in Robotics

This section will focus on the Application of heuristic algorithms for solving robotic path planning problematics.

6.1

Optimization refers to the decision making led by the machine with availability of certain circumstances and constraints thet either help or limit the outcomes of the optimizer [37]. In the case of path planning, optimization aims to modify some properties of the generated solutions to cope with the surrounding conditions. In stochastic environments, it is challenging to plan feasible paths. The increase in the complexity of the environment and the constant dynamics increase the number of variables involved in the mathematical representation of the problem. As a consequence, it raises the computation efforts that need to be deployed for the creation and optimization of the solution and decreases feasibility [7]. The integration of an optimizer that takes charge of the solution and fits it into the constraints that bound it is necssary. In order to get optimal results like smooth and short paths, finding a feasible path is not enough. The evaluation of the generated path and the optimization are helpful to ensure that the robot is driving the optimal path. There exists a range of planners and optimizers which are clustered as classic and Heuristic approaches [12]. Classic approaches include **Analytical** and **Enumerative** methods. The former rely on mathematical modeling, which can become increasingly complex as the navigation environment becomes more cluttered. Additionally, applying these models can be challenging in scenarios where the necessary models for different components are not readily available. The latter have the drawback of increased intricacy in bigger or more elaborate search spaces. On the other hand, Heuristic approaches can be sub-categorized into **Meta-Heuristic** and **Evolutionary algorithms**. Some of these methods can fall in the trap of local minima and sub-optimality [12] Alternatively, sebsequent to a review of the available planning and control approaches in intralogistics, Fragapane, G. et al. [7], concluded that nature-inspired algorithms can instill intelligence into planned paths. Elshamli, A. et al. explained in [17], that Meta-Heuristics are adaptive to the dynamics of the surroundings unlike classic approaches that proceed sequentially to generate solutions. The paths developed by classic approaches can thus become infeasible in a later stage if the environment changes. Metaheuristic approaches are based on parallel search mechanisms that can have updated input of the conditions and thus are more adaptive and practical in real-world scenarios.

6.2 Heuristic Optimization Algorithms : An Overview

In this section, we will dive into Meta-Heuristics which are defined by Bilal et al. as 'the optimization techniques mainly based on function evaluation and make little or no use of the properties of objective functions and constraints. Meta-heuristics are thus problem independent techniques not taking advantage of any specificity of the problem' in their review [37]. The authors later break down Meta-Heuristics into **Neighborhood-based Algorithms** and **Population-based Algorithms** as given by the tree graph in Figure 2.19

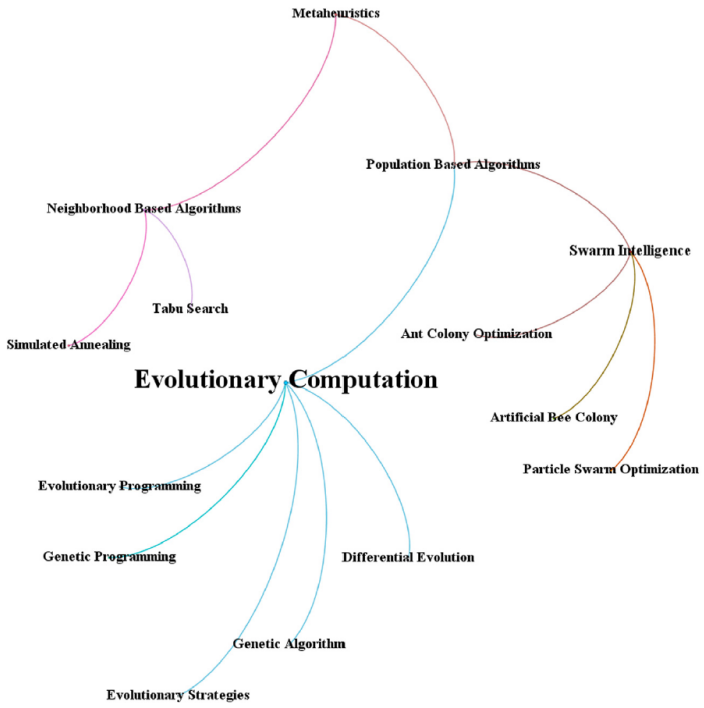

Figure 2.19: Tree of Meta-Heuristic Algorithms [37]

The Neighborhood-based algorithms benefit from the neighboring solutions by migrating from the current solution to the near surrounding solutions with the goal of improving the fitness function. The examples mentioned for this approach are Simulated Annealing (1979) and Tabu Search (1989). The Population-based algorithms are methods that start from a sample population and rely on their evolutions. They are inspired by the evolutionary processes observed in nature and are sub-categorized in this approach under Swarm intelligence and Evolutionary Algorithms. Swarm Intelligence approaches are analogous to the socio-cooperative practices of species like bees, ants, and birds. For instance, Ant Colony Optimization (1992)and Particle Swarm Optimization (1995) are examples of such algorithms. On the other hand, Evolutionary algorithms rather copy the species evolution theory, among which are Differential Evolution (1995) and Genetic Algorithm (1957) [37].

6.3 Application of Heuristic Optimization in Robotic Path Planning

These algorithms have been used multiple times for robotic path planning, as noted in the literature. Each algorithm has its own unique search methods and strengths, providing different benefits and areas for optimization.

Elshamli, A. et al. [17] have employed the Genetic Algorithm (See algorithm steps in figure 2.20) to solve the problem of path planning in dynamic environments. They represent their paths using chromosomes where each node of the chromosome represents a waypoint and develop a variable chromosome length approach that accommodates obstacle avoidance and efficiency in reaching the goal. The fitness of each chromosome is evaluated according to an objective function that includes path length, smoothness, and clearance to obstacles. A modified genetic Algorithm is applied where after crossover and mutation, paths are shortened and smoothed if it is possible. The implemented strategies guaranteed low convergence for the different complexity stages. Saving the best individuals generated by the GA was effective in delivering the overall best solutions. Tamilselvi, D. et al. also modified the GA by applying the elitism concept. They save the best solution S_{best} then they replace the worst element of the next population with S_{best}. This approach guarantees landing on a global optimum and was able to find feasible paths for environments with limited dynamic obstacles and integrate a motion prediction mechanism for dynamic obstacles based on the grid map [38].

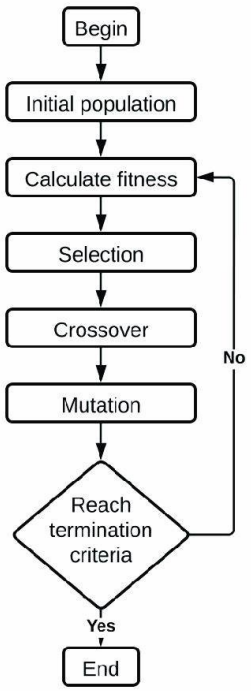

Figure 2.20: Flowchart of the GA [39]

Miao, H. et al. also worked on the path planning in dynamic environments problem. They crtic the GA to be computationally intensive and to require long planning time [40]. Instead they suggest the Simulated Annealing algorithm (SA) to solve the path finding problem.

Being a probabilistic meta-heuristic algorithm it is designed to guide the local optimum to a global optimum. The approach is based on the evaluation of the path length to discriminate path candidates: the shorter the path the better. It works by trying out different paths for the robot, beginning with a high "temperature" that lets the algorithm accept both good and bad paths. This helps it explore a wide range of options and avoid getting stuck in a local minima. As the process continues, the temperature slowly decreases, making the algorithm less likely to accept bad paths, and helping it to focus on finding the global minima as given by figure 2.21. algorithm was tested in four environments with different static and moving obstacles. The algorithm successfully found the best or nearly best paths and could quickly adjust to avoid collisions in real-time. Compared to a genetic algorithm, the SA method was 57% faster for complex environments and 74% faster for simple environments. However, the processing time increases exponentially for complex environments (13.57 seconds).

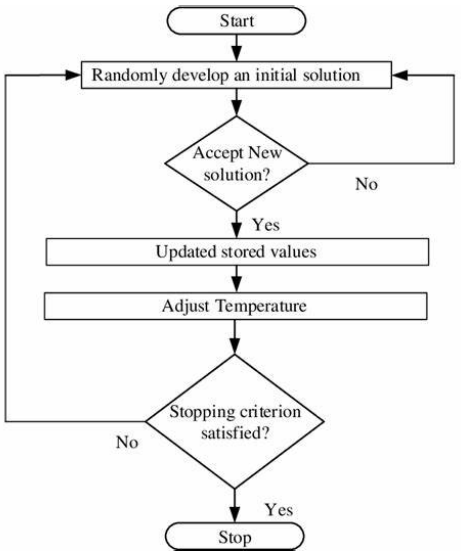

Figure 2.21: Flowchart of the SA [41]

Particle Swarm Optimization Algorithm (PSO), as well, is used for solving non-linear optimization problems like scheduling, power management and also robotic path planning. It is inspired from the cohesive behavior of bird and fish swarms traveling together.It works by having a group of particles (potential solutions) move around in the search space to find the best solution. Each particle in the PSO algorithm remembers the coordinates of the best solution it has found so far, known as its personal best (pbest). Additionally, the algorithm tracks the best solution found by any particle, referred to as the global best (gbest). The core idea of PSO is to guide each particle towards both its pbest and the gbest positions: from state a to c as given by figure 2.22, using a randomly weighted acceleration during each iteration. The algorithm iteratively updates the particles' positions and velocities until an optimal or near-optimal solution is found. Alaliyat, S. et al. have investigated dynamic environment navigation using PSO. They found out that the performance of the algorithm depends on parameters tuning. Overall, feasible paths were successfully generated. Planning time is rather high for simply organized environments (18.35s) and higher for the most complicated scenario (31.17s).

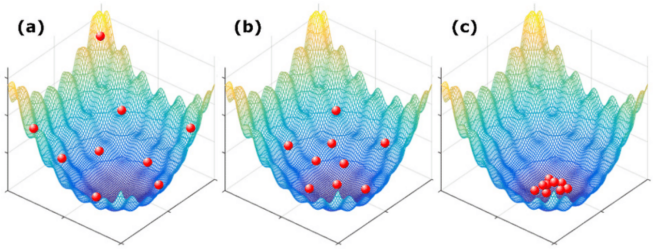

Figure 2.22: Particle Swarm Optimization [42]

Defferential Evolution (DE) has been used in a hybrid algorithm along with PSO by Tand, B. et al. [43] to leverage both of the algorithms benefits. DE is rarely solely applied to path planning problems. In their review [37] , of 20 years of research about DE, Bilal et al. did not mention robotics among the fields of application of DE. DE starts with a set of potential solutions. Each solution, or individual, is modified through mutation, crossover, and selection to improve over time. Mutation introduces random changes to create new candidates and avoid getting stuck in local optima. Crossover mixes parts of the new candidates with the original solutions to explore new possibilities. The selection process keeps only the better solutions for the next generation. This method effectively balances exploring new options and refining existing ones, making DE well-suited for complex optimization tasks. By integrating improved PSO and DE algorithms for enhancing particle diversity, Tand, B. et al. achieved better path quality compared to other methods. The results demonstrate that their approach outperforms traditional evolutionary algorithms in path optimality an optimize computation times by 1.38% [43].

Ant Colony Optimization (ACO) is known for being robust and good at parallel processing, but it can be slow and sometimes gets stuck in local optima solutions. To fix this, improvements like Ant Colony System (ACS) and Max-Min Ant System (MMAS) have been developed, which make ACO better at finding solutions but still have issues like being too rigid and converging too early. In mobile robot path planning, new strategies have been created to address these problems, such as adaptive methods that improve how pheromones are updated and balance exploration with finding the best path. Recent research has also introduced advanced versions like polymorphic ACO and multi-objective optimization, making ACO more effective for real-time and complex situations. However, these improvements also make the algorithm more complex and take more time to compute.

Meta-heuristic algorithms, such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Differential Evolution (DE), have become increasingly popular in robotic path planning. These algorithms are attractive because they can find good solutions to complex problems by mimicking natural processes like evolution or swarming behavior. However, despite their growing use in research and their potential efficiency in practical usecases, the application of meta-heuristic algorithms in real-world robotic path planning like the intralogistics field remains quite limited. One of the major issues is that most studies focus on simulations rather than testing in real environments. While

simulations can help develop and adjust these algorithms, they do not fully cover the challenges robots face in the real world, such as varying conditions, surface types, or unexpected noises. Another major issue is how obstacles are represented in these studies. Often, obstacles are either unrealistically large compared to the robot or have overly simple shapes that don't accurately reflect real-world conditions. This lack of realism can result in algorithms that work well in simulations but fail when used in real environments. Additionally, because these algorithms are mostly tested in controlled, simulated settings, they may be too finely tuned to those specific conditions. This can make them less reliable when applied to the unpredictable conditions of the real world. In summary, while meta-heuristic algorithms show promise for robotic path planning, more real-world testing and realistic obstacle modeling are needed to ensure they work effectively in practical situations. Addressing these issues is key to advancing the use of robotics in intralogistics and other fields.

7 Summary and Discussion

To synthesize the research conducted, it is paramount to summarize the key concepts comprehensively. This State of the Art report focuses on mobile robot path planning, a critical component of the overall robotic planning process. Effective path planning not only organizes and optimizes a robot's navigation but also significantly enhances its productivity by enabling it to execute missions more efficiently. In dynamic environments, a well-designed path planner is essential for ensuring safe operations and maximizing the robot's resource utilization and operational capacities. Moreover, Spline-based path planning represents a cutting-edge approach to achieving smooth and safe robot motion. This technique is particularly vital for managing heavy autonomous robots, such as forklifts, that handle sensitive materials and must operate with precision and stability. By utilizing constrained splines, it is possible to control various motion properties, including steering and the handling of abrupt turns, thereby reducing wear and tear on the robot and extending its operational lifespan. Nevertheless, there has to be a mean of quantifying the path qualities in order to test the performance of the used approach and to compare different approaches. This tool must be based on the properties of the path itself like length, direction changes, and clearance to obstacles. These information are useful in a later stage to optimize the paths by optimizing the supervised properties of the paths.

Meta-heuristic optimization algorithms are highly effective for refining mobile robot paths, particularly when building routes from scratch while considering both static and dynamic obstacles. They also have the capability to solve problems without information about their mathematical model. This has the great advantage of reducing the burden of expliciting the situation in order to solve it especially in complex cases where the effort to model the problem and the contsraints is higher and, later, the computation is harder. Instead, meta-heuristic algorithms focus on optimizing the fitness function related to the specific usecase. However, these algorithms are also computationally demanding due to their iterative nature, which involves repeatedly constructing and refining the path segments from the starting point to the end goal. Previously reviewed literature reveals that these algorithms often require significant computation time, which is acceptable in simulation environments but problematic in real-world applications. In a dynamic warehouse setting, blocking the area for extended periods to perform path planning is impractical. Such delays can disrupt warehouse operations, interfere with other vehicles and personnel, decrease overall productivity, and lead to increased time and energy costs. In addition the applications of meta-heuristic optimization approaches to robotics in the intralogistics field is limited to mission scheduling and rarely tackles path planning problems.

Furthermore, the classical and the heuristic approaches are capable of creating new different paths every time the algorithms are ran even in the same environment setting. While this phenomenon proves of the intelligence of the algorithms and the availability of several optimal paths, it makes the fully automated robot's behavior non-repeatable and unexplainable. For humans, it is difficult to foresee the behavior if many options are available. However, it is important for those working closely to robots to have an expectation

of their behavior. It enables them to avoid collision risks and injuries and to plan their actions accordingly. If a predictable vehicle behavior is available, it promises a structured approach to path generation, reducing computational complexity and improving path quality.

Combining an optimal path planning approach that accomodates the kinematic constraints of Autonomous forklifts in the intralogistics environment and generates smooth paths, is behaviorally explainable and deploys meta-heuristic Algorithms is a new approach that would have an added value to mobile robot path planning in Intralogistics of reducing computation time and travel distance and improving AI driven solutions explainability.

Chapter 3
Methodology & Implementation

Chapter 3

Discussion of the problem and the proposed solution

Introduction

This chapter details the methodology and implementation of the proposed path planning system for logistics forklifts, focusing on guiding the robot to a pallet in the tight spaces of a warehouse. The system uses pattern-based paths with splines, optimized by meta-heuristic algorithms, to navigate efficiently in these confined areas. The chapter begins with an overview of the design structure, followed by a clear explanation of the steps and techniques used to ensure smooth and accurate movement toward the pallet. Implementation details are then covered, showing how the system components are integrated into a cohesive solution for precise pallet access within the warehouse environment.

1 Conceptual Framework

1.1 Choice of Methodology

Based on the State of the Art (Chapter 2), we discussed that classic and heuristic approaches have some drawbacks, such as complexity and long planning times. These approaches are typically designed to solve path planning problems in various environments and have been used in multiple applications.However, in our case, the robot operates in a consistent environment—the station. Although stations may differ in position, size, and layout, the components within them remain the same: shelves containing pallets. This consistency suggests that we should develop a solution specifically tailored to the station environment, which would help minimize potential problems and reduce the effort

required to solve them. A path planning module has already been developed for the autonomous forklift trucks, utilizing classic sampling-based path planning methods. This module is based on the Open Motion Planning Library (OMPL), an open-source library for motion planning in robotics. OMPL is designed for complex use cases and includes several algorithms, such as probabilistic roadmaps (PRM) and rapidly-exploring random trees (RRT), along with their variants. OMPL performs well in clean environments without obstacles, with a minimum processing time of 150 ms, but in complex environments, processing time can increase to 2 seconds. While it can be very fast in the abscence of obstacles this approach presents some limitations:

- Not Station-Specific: OMPL is built to fit many different environments, so it is not naturally designed for the station-related pickup/drop operations.
- Too Complex for Simple Tasks: OMPL includes advanced algorithms that are more powerful than necessary for straightforward tasks like picking up and dropping off pallets.
- Longer Processing Times: OMPL can take longer to process in complex situations.

Considering these limitations of OMPL, it is important to develop a simpler, station-based approach. This new method should be specifically tailored to the consistent environment of the station, allowing for quicker and more efficient path planning. By focusing on the typical tasks of pallet pickup and drop-off, we can create a solution that works effectively in simple and moderately complex situations, reducing processing time and resource usage. This approach would serve as the primary option for linking the robot in the station to the pallet. In case of very complex environments and failure, the OMPL based approach becomes the secondary option.

For optimizing paths, various approaches exist. Path optimization can be approached mathematically using Jacobians, which map control inputs, like position or wheel speeds, to the desired outcomes. The objective is to minimize a cost function—such as path length, energy use, or time—by adjusting the robot's control inputs to achieve the desired position. A common technique is gradient descent, where control inputs are iteratively updated to reduce the cost. The Jacobian matrix, which links these inputs to the robot's position, guides these adjustments until the most efficient path is found. However, the Jacobian-based optimization method poses several challenges for this use case. For wheeled mobile robots with multiple control inputs, the Jacobian matrix can become complex and large, leading to increased computational costs and making real-time optimization challenging. Frequent updates in dynamic environments further add to this load, potentially leading to delays or suboptimal paths. Mukherjee et al. [44] noted that their approach to optimal trajectory planning does not guarantee a global optimum but rather an isolated local optimum close to the global one. Additionally, this method relies on precise kinematic and dynamic models, so any inaccuracies, such as sensor errors, can result in poor path optimization. In this context, the kinematic model would need to be refined for the specific truck using it. Managing complex inequality constraints, like obstacle avoidance, can also complicate the optimization process and reduce efficiency.

On the other hand, Reinforcement Learning (RL) can be used to learn optimal paths by interacting with the environment. The robot learns a policy that maximizes cumulative rewards, where rewards are given for reaching the goal, avoiding obstacles, and minimizing path length. The policy is often represented by a neural network, and the path is optimized as the robot learns from its experiences. Reinforcement learning (RL) methods for path planning in wheeled mobile robots face significant limitations due to high computational costs and long training times. These methods require extensive data to cover various scenarios, making the process time-consuming and resource-intensive. Training an RL agent, particularly in complex environments, can take millions of interactions, which is impractical for real-time applications. Additionally, as the environment's complexity increases, the state and action spaces expand, leading to scalability issues and longer training times. When combined with deep learning, RL becomes even more computationally demanding, requiring substantial resources and careful tuning of hyperparameters.

As highlighted in Section 2.4, meta-heuristic approaches, though powerful for solving complex problems, often require significant computational time. However, this can be greatly reduced by optimizing how these methods are applied. For instance, instead of generating a path entirely from scratch, starting with a pre-built or partially constructed path provides a solid foundation. Additionally, reducing the number of waypoints can streamline computation, allowing the algorithm to focus on refining an already good solution rather than calculating every detail from the ground up. This approach preserves the strengths of meta-heuristics while significantly speeding up the planning process, making it more practical for real-time or near-real-time applications.

1.2 Goal Setting and Work Structure

Considered the problems detailed in chapter 1 and the previous conclusions, the following goals were identified based on the company's vision and the general requirements for applications to autonomous mobile robots in intralogistics:

- **Predictability and Robustness of Mobile Robot Behavior:** The solution utilizes an intelligent approach to connect the robot to its destination. The developed algorithm generates an explainable navigation process that enables humans to understand and trust the technology. Additionally, secure decision-making for autonomous trucks is ensured by the developers.
- **Accuracy of the solution and station relation:** The developed solution is stable, repeatable and flexible: this means that it would work for almost all stations in warehouses where it operates. The linking to the destination is accurate and precise to avoid failures. Precision and accuracy are guaranteed by the fact that the planning is done in the limited station space: a narrow field that allows short distance perception and planning. The solution is then tailored to each environment's specificities.

- **Optimization of paths:** The algorithm optimizes the path by minimizing travel distance, reducing sudden turns, and avoiding challenging maneuvers that could strain the truck. This optimization ensures the most efficient route, balancing speed and safety. The approach is designed to be computationally efficient, allowing for real-time adjustments and smooth behavior of the autonomous truck.
- **Obstacle avoidance:** The vehicle avoids collisions and drives safely to its destination. The system is designed to prioritize safety while maintaining an efficient and effective route to the target location.

The following sections will detail the chosen methodological approach. Then, the implementation steps are explicited and demonstrated from the simulation point of view. Finally, the tests results are explained and discussed and the outlooks are elaborated.

2 | Methodology and Design of the Solution

Starting with the operating environment, the stations, the focus was on the problems inside. The issue of linking the forklifts in the near-field to the pallet is nothing but new to the intralogistics sector. The process is repeated dozens of times daily inside warehouses and factories where forklift drivers take control over resolving the problem. Taking a deeper look at their approaches to resolve the same issue, the autonomous vehicles team noticed that forklift drivers have certain driving styles that are repeated according to the current situation. Similarily to car parking, depending on the type of the parking slot, there are some patterns to reproduce in order to fit the car inside its assigned spot as given by figure 3.1. In case of a backward parking, the driver needs enough space for direction changing. To do so, he drives further to the opposite direction to allow the car to comfortably and smoothly drive to its parking place. Parking assistance feature is embedded in human-driven and driverless cars to instill safety and order.

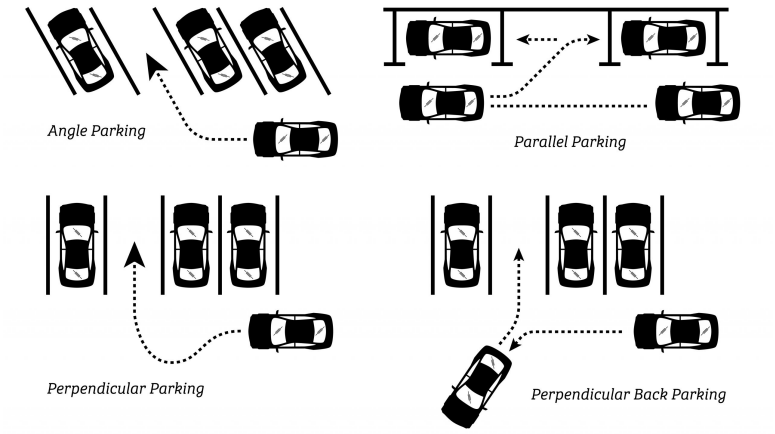

Figure 3.1: Parking Styles

Analogically, after the truck arrives in the station, the drivers would drive and steer in the opposite direction of the pallet, until the forks are in a position to allow the truck to easily and correctfully drive to the pallet in fork direction. The goal here is to develop a pallet-linking algorithm to automate the driving to the pallet pickup location for autonomous forklifts as given by figure 3.2. The truck would autonomously plan the path to its destination based on the pattern and on the environment settings that it would navigate in. The pattern enhances the explainability in the forklifts behavior. Explainability allows for more order in the warehouse: it simplifies the coordination of safe simultaneous tasks around the truck. This is achieved through the design of an algorithm that aims to create a transparent navigation process, making it easier for humans to understand and trust the technology. Additionally, the approach is intended to ensure secure and reliable decision-making for autonomous trucks.

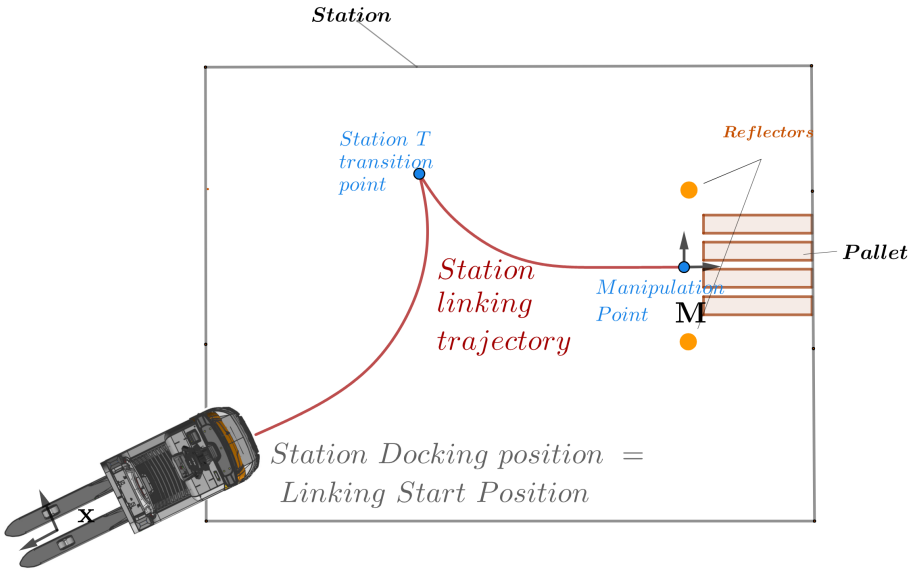

Figure 3.2: Linking the robot to goal destination [28]

From now on, the following definitions of driving directions will be used as gven by figure 3.3:

- Main Driving Direction: Driving in fork direction.
- Opposite Driving Direction: Driving in vehicle chassis direction.

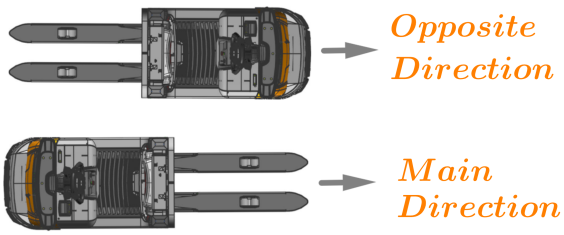

Figure 3.3: Truck Driving Directions

The following transition method is suggested: once the truck docks the station, it has the option to transition and change direction on the sides of the shelf placed inside the station. First, it drives to the transition position, then, to the pallet or the drop-off position as given by figure 3.4. Having two possibilities for transition zones allows for more flexibility: in case one area is unreachable or presents obstacles, the second one can be used. They present free areas inside the stations where less obstacles and dynamics are expected and smooth maneuvering of the truck can be managed. In addition, the

geometric solution is scalable to any station that is recognized and whose properties are available to the truck. This makes the overall solution and the autonomous forklifts simple to commission in new warehouses.

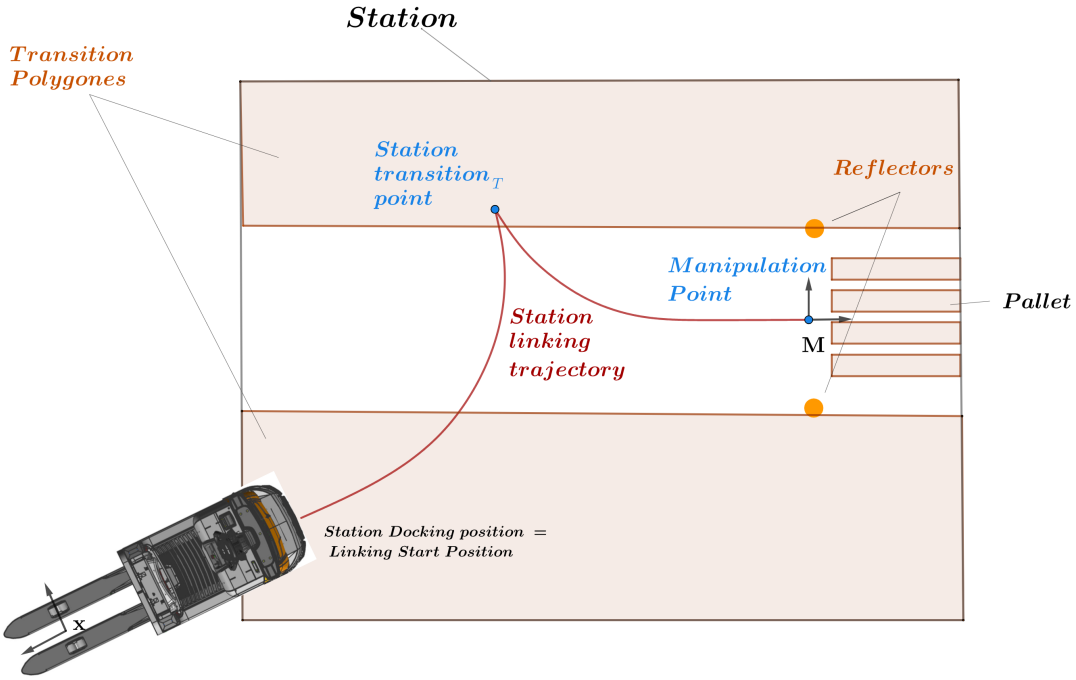

Figure 3.4: Linking the robot to goal pallet [28]

Smooth and continuous paths are generated through spline interpolation of constructed waypoints. This technique ensures that the resulting paths are smooth, operationally seamless, and reduce abrupt changes in direction and speed. By interpolating between waypoints, the method enables the creation of curves that are optimized for driving efficiency and stability.

Finally, an optimizer generates an optimal path. generates multiple path candidates and rigorously evaluates each one based on key factors. The evaluation considers the path's length, smoothness, and potential collisions with surrounding objects to ensure the chosen path is both efficient and safe. After testing all options, the path that best meets these criteria—providing the shortest, smoothest, and safest route with minimal collision risk—is selected as the best solution.

This approach ensures the final path is not only theoretically ideal but also practical and reliable for real-world use. The process is designed to be flexible and effective, even in obstacle environments. The designed methodology is summarized in the following flowchart 3.5:

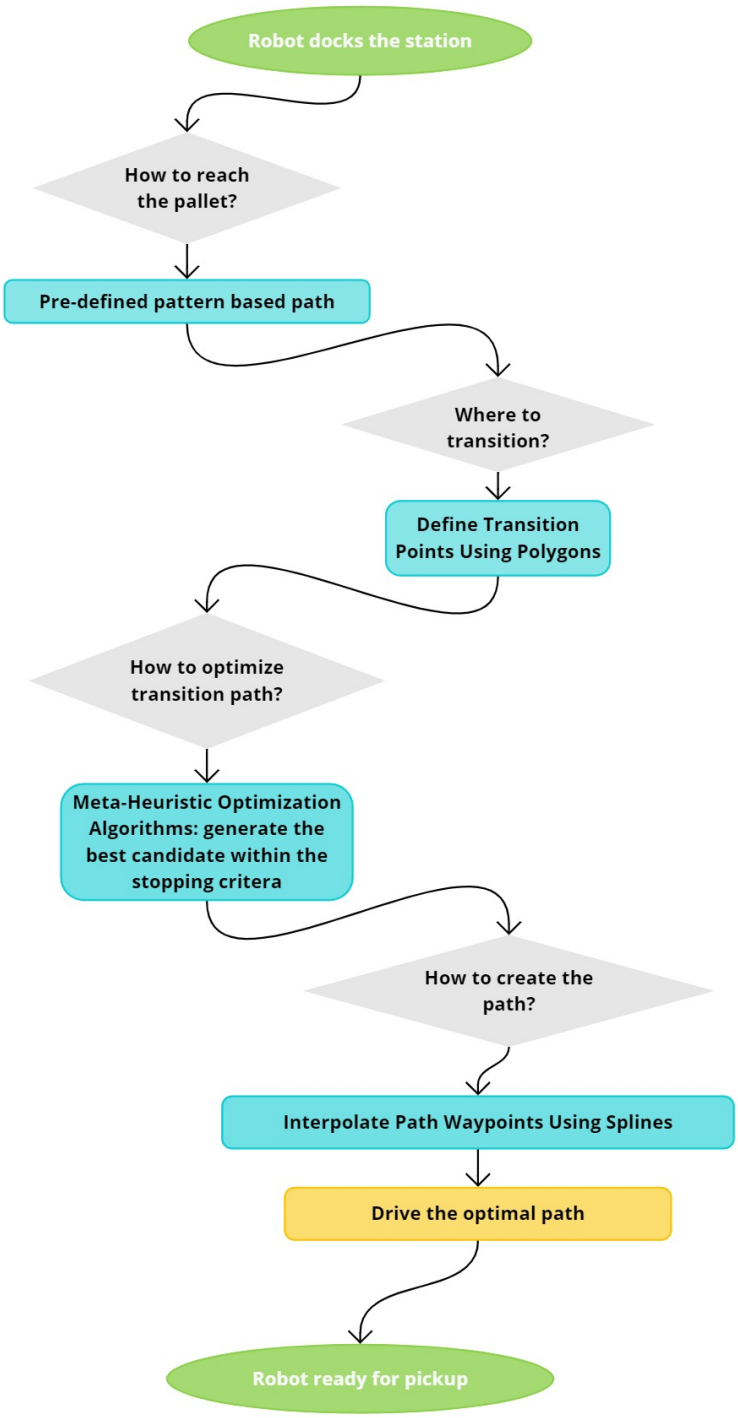

Figure 3.5: Path planning approach to pallet docking

3 Development phases and Implementation

This section details the rollout of the module development, providing a comprehensive overview of the process. It is structured to include both the development steps and the corresponding simulation results at each phase. Every phase started with a deep investigation of the existing relevant solutions detailed in Chapter 2.

3.1 Geometric Partitioning of the Station

This section explains the development of the geometric Partitioning of the station into 2 polygons used later for transitioning inside the station. **The input of this step is the station information and the output are the polygons**.

3.1.1 Utility

As explained previously in Section 3.2, in order to carry out the transitioning of the vehicle inside the station to change the driving direction from the opposite to the main direction, Transition subpolygons are needed. The subpolygons are created according to the station shape and dimensions and the position of the shelf inside of it. As given by figure 3.6, the stations can be found at certain positions of the warehouse.

3.1.2 Implementation

The vehicle initially plans its missions: it constructs a plan of the stations that will be visited and the operations to do on each station. While the path driven from one station to the other is pre-defined by the global planner, the path linking the vehicle to the pallet is to be planned in real-time. Once the vehicle arrives at the docking position in the stations, it starts the path planning process. The station-related information are stored in a file accessible to the vehicle. These information include, but are not limited to, it position and dimensions and the coordinates of the shelf that it contains. For each station represented like figure 3.7 the module creates the subpolygons on each side of the shelf, limited by the station and shelf shapes.

The process that creates the subpolygons is detailed in the following Algorithm 1. The algorithm begins with an initialization step, then the corners of the station and shelf areas are retrieved and transformation matrices are prepared to convert coordinates between global and shelf-specific coordinate systems. The need to transform the used coordinates to the shelf frame is derived from the scalability needs. As seen on figure 3.6, stations can be placed in different positions and have different orientations. If global coordinates

are used, it becomes computationally challenging to develop an exhaustive algorithm that fits all the possibilities of station orientation and shelf configurations. On the other hand, using the shelf coordinate frame shifts the point of view to the inside of the station only. It enables the shelf to orchestrate the geometric partitioning of the station based on its position in the station. The algorithm then iteratively transforms the station and shelf corners from global coordinates to shelf coordinates, storing the transformed corners in dedicated lists. After the transformation, two subpolygons are created: one representing the negative Y subpolygon (NegYsub_polygon) and the other representing the positive Y subpolygon (PosYsub_polygon). For special cases like station3, where one of the subpolygons can be very narrow (lower than a certain threshhold), the algorithm omits it and generates only one subpolygon. The algorithm concludes by returning the created subpolygon(s), effectively segmenting the input areas into distinct, usable geometric entities.

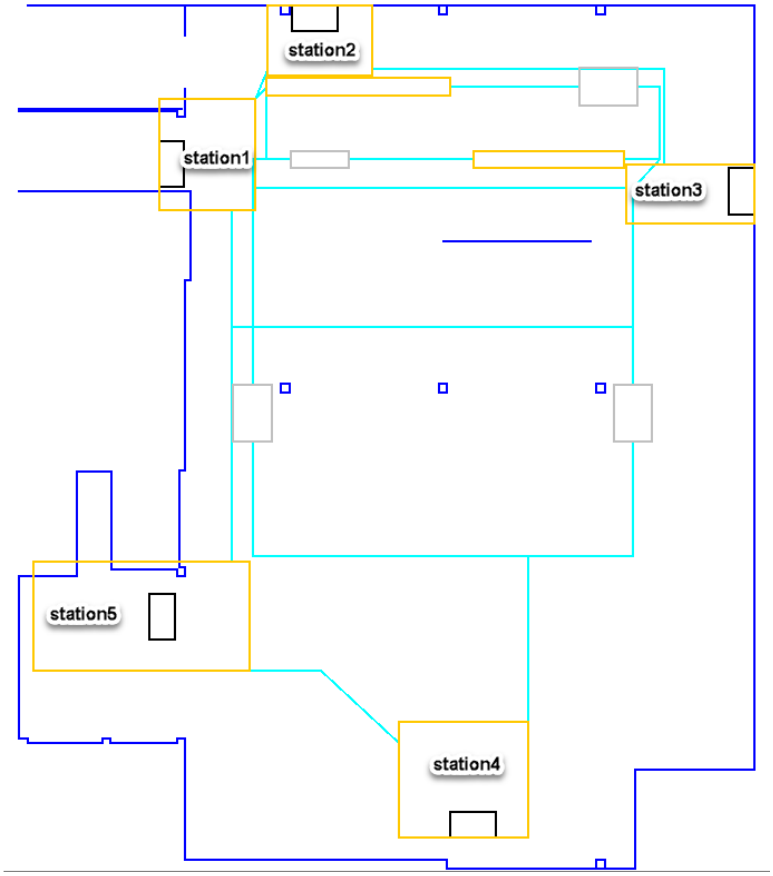

Figure 3.6: Simulation of the stations in a warehouse

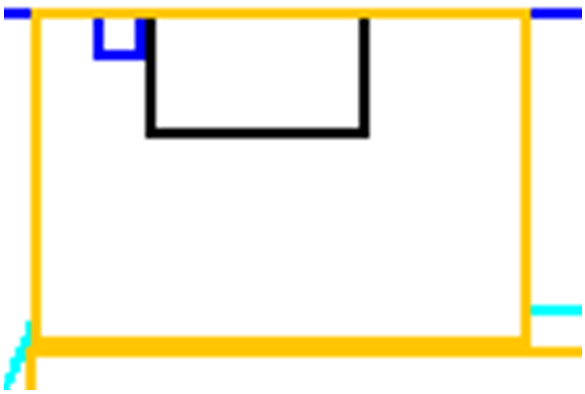

Figure 3.7: Station Simulation

Algorithm 1: Creation of Subpolygons

Data: station_area, shelf_area
Result: created two subpolygons

Initialization;
station_corners, shelf_corners ← GetCorners(station_area, shelf_area);
tm_station_corners, tm_shelf_corners ← empty lists;
tm_shelf2global ← GetTransMatrix(shelf_area);
tm_global2shelf ← InvertMatrix(tm_shelf2global);
foreach corner ∈ station_corners **do**
    corner_shelf ← TransformCorner(tm_global2shelf, corner);
    tm_station_corners.append(corner_shelf);
station_corners_shelf.append(tm_station_corners[0]);
foreach corner ∈ shelf_corners **do**
    corner_shelf ← TransformCorner(tm_global2shelf, corner);
    tm_shelf_corners.append(corner_shelf);
NegYsub_subpolygon ← CreateNegYsubpolygon(tm_station_corners, tm_shelf_corners, tm_shelf2global);
PosYsub_subpolygon ← CreatePosYsubpolygon(tm_station_corners, tm_shelf_corners, tm_shelf2global);
return created two subpolygons;

3.1.3 Results

In figure 3.8 stands the simulated result processed using real station data: In purple and orange are the polygons, and in green the outer polygon represenst the station.

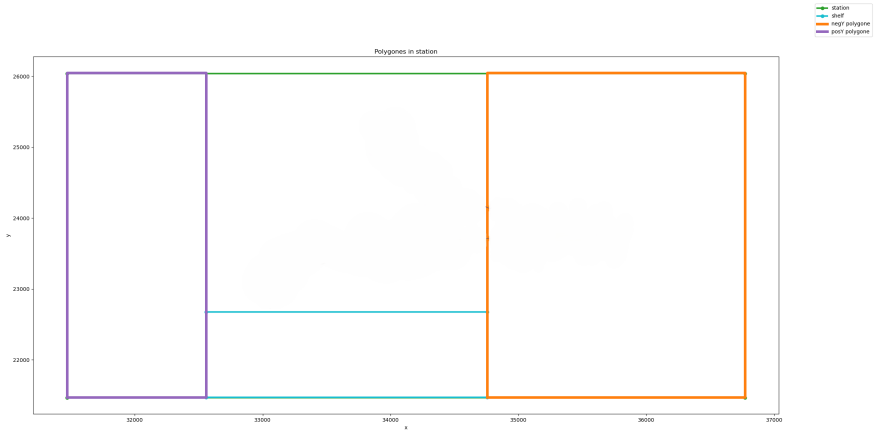

Figure 3.8: Station with created Polygons

In figure 3.9, stands the result of the module Create subpolygon in a different station:

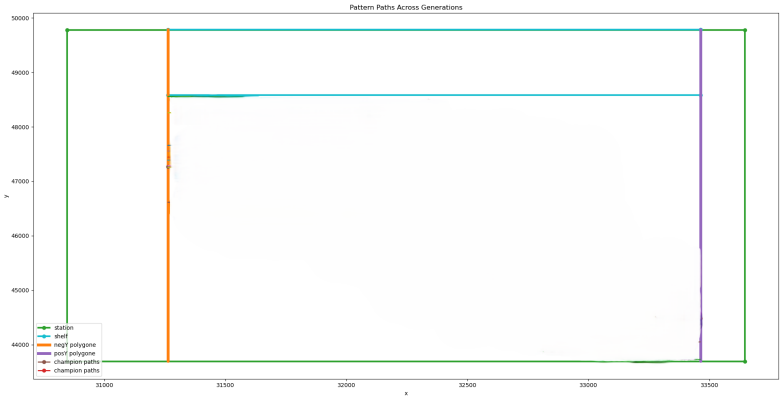

Figure 3.9: Station with created Polygons

3.2 Path creation

This section details the methodology for path creation using splines. **The input of this section are the polygons besides the robot's position and the destination. The output is a spline-based path linking the robot to the destination with a driving direction change**.

3.2.1 Utility

Spline-based paths provide several key benefits for robotic navigation. They ensure smooth and continuous movement by creating curved paths that connect waypoints seamlessly. Unlike abrupt turns and sharply angled paths, this smoothness reduces wear on the robot and makes its movements more energy-efficient. Additionally, spline-based paths make it easier to control the robot in real-time, allowing for quick adjustments if the environment changes and preventing oscillations as a response to control corrections. Using mathematical equations to define these paths also makes it simple to calculate important properties like path length and curvature, helping the robot navigate efficiently and safely.

3.2.2 Implementation

Path creation comes after the creation of the subpolygons. After a transitioning point is chosen (see section Path optimization), the waypoints are created. The waypoints depicted on the simulation figure 3.10 are created relating to the Robot, the destination and the transition positions. In a simple scenario where no obstacles are present, orientation points are strategically added in front of the three key positions of the robot. These orientation points serve as reference markers that guide the robot's movement, particularly during the initial phase of acceleration or deceleration. When the robot starts to move, it follows a precise trajectory, either in a straightforward or straight-backward direction, depending on the intended driving direction. This straight-line motion ensures that the robot maintains stability and accuracy as it transitions from a stationary position to motion or vice versa, minimizing any unnecessary deviations or turns that could disrupt its path. The alignment with the orientation points helps the robot to consistently maintain a straight trajectory, which is crucial for precise navigation and control.

Then, the orientaion points are interpolated to create the spline-based path. Given the need to create the driving direction change as a straight section of the path, two different splines will be interpolated using 4 waypoints each:

- The first spline interpolates the starting points then the transition points.
- The second spline interpolates the transition points then the destination points.

By doing so, two connected splines are obtained. Then they are merged together in the same path data structure to form one path. The first spline is driven in opposite direction driving, the second in main direction driving. The transition section at the polygon enables the robot to have an orientation that facilitates changing the driving direction as given by figure 3.12.

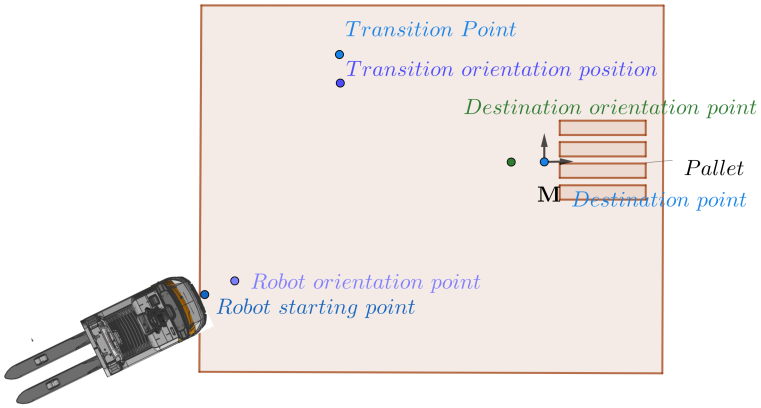

Figure 3.10: Orientation Points

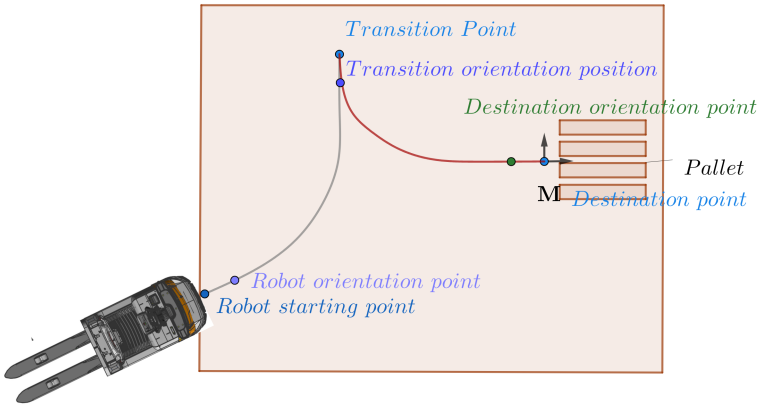

Figure 3.11: Interpolated Splines of Orientation points

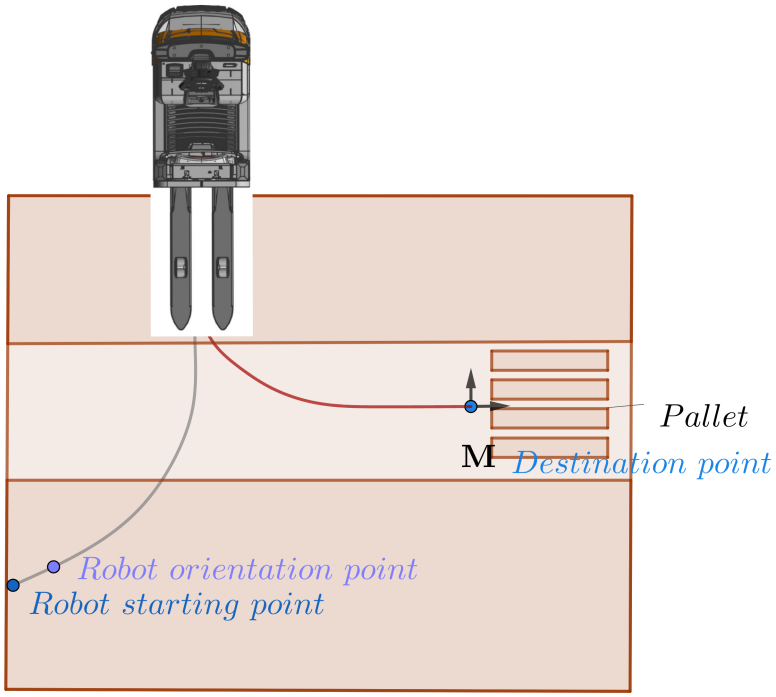

Figure 3.12: Robot at the transition position

The interpolation of the waypoints and the manipulation of splines is through The Spline Library integarted in the RACK framework (See RACK section). The library provides tools to construct splines through several methods like interpolating points or blending two or more splines together and retrieve their features by segmeting the splines and calculating length and curvature for instance. The result is a structure of coordinates through which passes the spline. The Algorithm of the approach to create a spline is explained in Pseudo code 2. The algorithm first acquires the waypoints to be interpolated. Then it creates the curve and measures the minimu and maximum parameter values from the knot vector which are used to evaluate the curve but also to retrieve relevant data about the splines like positions. It guides the exploration of the spline. Besides, the spline length and cumulated curvature change are retrieved. The curvature change supervises the abrupt changes in curvature in segment curves of the whole spline. Finally a number of positions belonging to the spline is retrieved and transfered to the path data structure, this is where the resulting spline is transformed into a path that the robot can follow.

Algorithm 2: Creation of a Spline by Points Interpolation

```
Data: points
Result: createdNewSplineFromPoints
Initialize Spline;
SetInterrogationPoints(points);
CreateCurveByInterpolation();
max_par ← Spline.GetMaxParameterVal();
min_par ← Spline.GetMinParameterVal();
length ← Spline.GetFullCordLength();
curvature_change ← Spline.GetCurvatureChange();
samples ← round(length / kSegmentLength);
step ← (max_par - min_par) / samples;
for f ← min_par + step to max_par by step do
    Spline.GetPosition(f, position);
    PathData.Spline.Position ← position;
end
return createdNewSplineFromPoints;
```

3.2.3 Results

The obtained results are of smooth splines adhering to the pattern and linking the truck to the destination pallet while passing by a transition location. The simulated results of a cloned test environment show the pattern spline-based path as given by figure 3.13. The resulting path on the RACK simulation environment which is the real testing environment show as well the creation and plotting of the pattern path on the simulated station as given by 3.14 and 3.15.

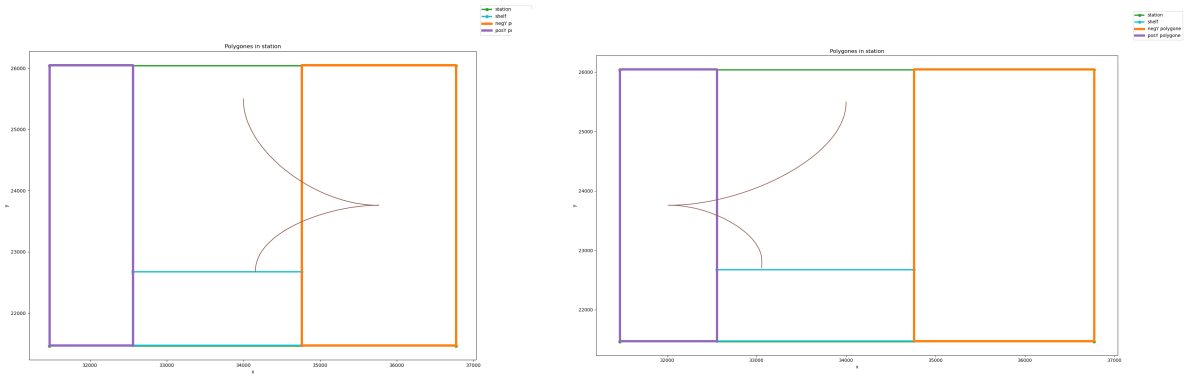

Figure 3.13: Test Results on Cloned Test Environment

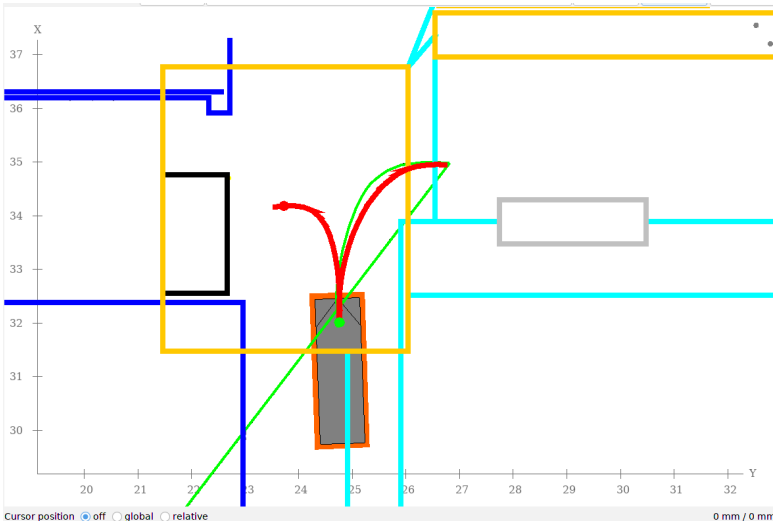

Figure 3.14: Test Results on the Simulated Environment: Truck driving the Spline-based Pattern path

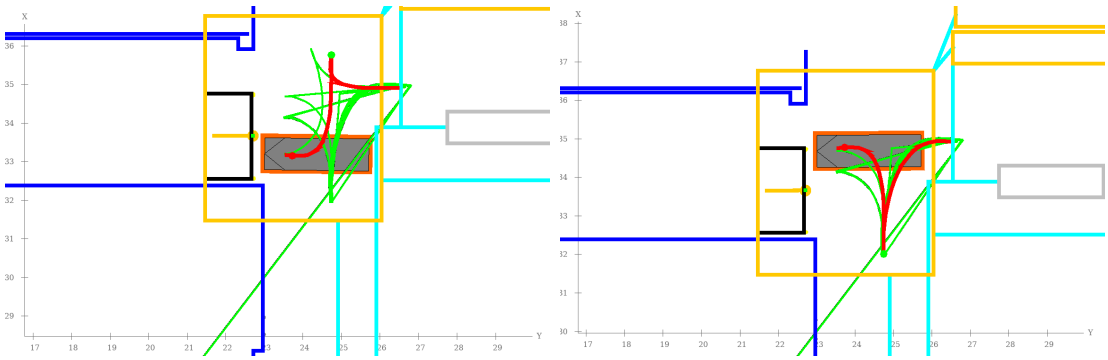

Figure 3.15: Test Results on the Simulated Environment: Truck at the Destination

3.3 Path Evaluation

Once the path has been created and the relevant information has been retrieved, the foundation is set to evaluate the quality of the generated paths. As discussed in Section 2.3, the literature proposes several approaches to path evaluation that have been tested and assessed in this work. **The input for this section is a spline-based path pattern, and the output is a scalar value that reflects the quality of the path**.

3.3.1 Utility

Path evaluation allows to assess the quality of a path. From the vehicle's perspective, it enables the prediction of whether the path is mechanically and strategically feasible in

terms of curves, sharp turns, direction changes, and other factors. This evaluation prevents the vehicle from following low-quality paths and paves the way towards optimization and selection of the best possible routes.

3.3.2 Implementation

By analyzing the spline, key characteristics of the path can be determined, such as its length, curvature, and the time required to traverse it. The following requirements are essential for developing a robust path evaluation formula:

- The formula used to evaluate the path must be **simple** enough to be easily integrated into an optimization process. During path planning, optimization algorithms are often employed to find the best possible path according to certain criteria. These algorithms require objective functions that can be computed efficiently. A complex or computationally expensive formula may slow down the optimization process or make it impractical for real-time applications. Therefore, the path evaluation formula should be simple, allowing for quick calculations that can be repeatedly applied within an optimization loop. This simplicity also facilitates its integration with existing optimization frameworks.
- The formula should include a **normalization** step to ensure that different paths can be compared on a common scale. Paths can vary significantly in terms of length, curvature, and other characteristics. To make meaningful comparisons between paths, it's essential to normalize these metrics. Normalization involves scaling the values of each metric (e.g., path length, curvature change) relative to a reference value, such as the characteristics of a potential ideal path. By normalizing the path metrics, it is ensured that the evaluation formula is dimensionless and that the different aspects of the path (e.g., length versus curvature) are balanced appropriately. This step is crucial when combining multiple metrics into a single evaluation score, as it prevents any one metric from disproportionately influencing the final result.
- The evaluation should **prioritize paths that minimize length and curvature**. In most path planning scenarios, especially for vehicles or robots, the goal is to find the most efficient path from point A to point B. Efficiency can be measured in terms of:
  - Path Length: Shorter paths are generally preferred because they reduce the distance the vehicle or robot needs to travel, therefore saving energy and time.
  - Curvature: Paths with less curvature are preferred because sharp turns can be difficult for vehicles to navigate, especially at higher speeds. Minimizing curvature reduces the risk of instability of the vehicle.

The combination of these requirements leads to a path evaluation formula that is both practical and effective.

To implement this path evaluation effectively, it was essential to accurately measure the path's length and curvature change, as these are critical components of the evaluation formula.

The path length was calculated by summing the distances between consecutive points along the spline, ensuring that even small variations in the path's trajectory were captured as given by equation 3.1.

$$L = \sum_{i=\text{min par}}^{\text{max par}} s(i) \tag{3.1}$$

where $s(i)$ denotes the length of the i-th curve segment of the spline measured between two points of the spline using Euclidean distance.

The curvature change of a spline path is quantified by evaluating how the curvature of the path varies along its Length. The curvature change (cc) calculated by:

$$cc = \int \frac{d(k)}{d(s)}\, ds \tag{3.2}$$

represents the total variation in curvature along the path, where $\frac{d(k)}{d(s)}$ measures how curvature itself changes as a function of arc length, and integrating this derivative over the path provides a comprehensive metric of curvature variation.

As this equation involves an integral, it has to be approximated in order to compute it in a programming environment. The approximation is conducted through a discretization of the path into a series of small segments. For each segment, the curvature change and segment length are approximated. Let $\Delta s(i)$ denote the length of the i-th segment and $\Delta k(i)$ the change in curvature over this segment. Then, the trapezoidal rule is used to approximate the integral. This is achieved by summing up the ratio of curvature change to segment length over all segments as given by equation 3.3:

$$cc = \sum_{i=\text{min par}}^{\text{max par}-\text{step}} \frac{\Delta k(i)}{\Delta s(i)} \tag{3.3}$$

Mathematically, if $k(i)$ is the curvature at the i-th segment, then $\Delta k(i)$ can be expressed as:

$$\Delta k(i) = |k(i) - k(i-1)| \tag{3.4}$$

where $k(i-1)$ is the curvature of the previous segment. The absolute value is used to prevent cancellation of terms: curvature can be negative or positive according to the convexity or concavity of the curve. When summed together, a neutralization could happen resulting in an insignificant value. This could be solved by introducing the squarred difference between $k(i)$ and $k(i-1)$. However, it will minimize the result to a negligible value for the overall analysis as the curvature values are around the magnitude of 10^{-2} or less.

The summation provides a cumulative measure of curvature changes relative to the segment lengths along the entire path, offering insight into the overall curvature dynamics of the spline. The formula for *cc* evaluates the total effect of curvature changes normalized by segment lengths over the path. This metric is useful for understanding how the path's curvature varies and helps in optimizing paths to ensure smooth and efficient navigation.

To develop a cost function that integrates both path length and curvature, we explore two distinct approaches: Exponential Weighted Path Evaluation and Normalized Weighted Path Evaluation. These approaches use weights to prioritize the significance of each component in the evaluation process.

3.3.2.1 First approach: Exponential Weighted Path Evaluation

The Exponential Path Evaluation is based on A. Elshamli, et al. method [17]. As explained in section 2.3, they consider path length, smoothness and clearance combined in a weighted cost function. Their approach of measuring path length and smoothness is considered here with some changes to fit it into the use case.

Path length term is measured in [17] as follows:

$$d = \sum_{i=1}^{n-1} d(s_i) \tag{3.5}$$

Whereas, Path smoothness is measured as follows:

$$d = \sum_{i=2}^{n-1} \exp(a(\theta_i - \alpha)) \tag{3.6}$$

where θ_i represent the steering angle at the i-th segment, and α denote the desired steering angle.

Besides path clearance, those values are weighted and summed. This summation is not normalized given that it adds the distance term which has a unit to the smoothness and clearance terms that are exponential functions results. In this approach of the work, the exponential function is used for normalization of distance and curvature change terms. The length term is given by:

$$\text{Length Term} = e^{\beta \cdot (\text{current}_L - \text{ref}_L)} \tag{3.7}$$

The curvature change term is given by:

$$\text{Curvature Change Term} = e^{\alpha \cdot (\text{current}_{cc} - \text{ref}_{cc})} \tag{3.8}$$

where:
- *current* refers to the path being evaluated and *ref* refers to the reference path.
- α and β are factors that rationalize the exponential terms because the length is in the range of thousands of millimeters while the curvature is around the magnitude of 10^{-3}. Without these factors, the two terms would differ significantly, leading to incorrect analysis of the impact of each term.

The cost function of the weighted terms is given by equation 3.9:

$$Z^* = \arg\min \left(\omega_c \cdot e^{\alpha \cdot (\text{current}_{cc} - \text{ref}_{cc})} + \omega_L \cdot e^{\beta \cdot (\text{current}_L - \text{ref}_L)} \right) \tag{3.9}$$

with constraints:

$$K(i) \ < \ K_{\max}$$

and

$$\text{RobotFootprint}(\mathbf{p}_i) = 0 \quad \text{for all } i$$

where

- ω_c and ω_L are the weights assigned to the curvature change term and the length term, respectively. These weights determine the relative importance of curvature change and path length in the overall evaluation.
- $K_{\max}$ the maximum tolerated curvature
- $\mathbf{p}_i$ represent the i-th scan point in space.
- RobotFootprint($\mathbf{p}_i$) be a boolean function that returns 1 if the point $\mathbf{p}_i$ lies within the robot's footprint (polygon) and 0 otherwise.

3.3.2.2 Second approach: Normalized Weighted Path Evaluation

The Normalized Weighted Path Evaluation is based on the division of the path's Length and Curvature Change measured respectively using the equations **??** and 3.3 respectively by the reference path's Length and Curvature Change. The length term is given by:

$$\text{Length Term} = \frac{\sum_{i=\text{min par}}^{\text{max par}} s(i)_j}{\sum_{i=\text{min par}}^{\text{max par}} s(i)_{\text{ref}}} \tag{3.10}$$

The curvature change term is given by:

$$\text{Curvature Change Term} = \frac{\sum_{i=j\,\text{min par}}^{j\,\text{max par}} \frac{\Delta k(i)_j}{\Delta s(i)_j}}{\sum_{i=\text{ref}\,\text{min par}}^{\text{ref}\,\text{max par}} \frac{\Delta k(i)_{\text{ref}}}{\Delta s(i)_{\text{ref}}}} \tag{3.11}$$

where j denotes the current path being evaluated.

The cost function of the weighted terms is given by equation 3.12:

$$Z^* = \arg\min \left(\omega_C \cdot \frac{\sum_{i=\text{j min par}}^{\text{j max par}} \frac{\Delta k(i)_j}{\Delta s(i)_j}}{\sum_{i=\text{ref min par}}^{\text{ref max par}} \frac{\Delta k(i)_{\text{ref}}}{\Delta s(i)_{\text{ref}}}} + \omega_L \cdot \frac{\sum_{i=\text{min par}}^{\text{max par}} s(i)_j}{\sum_{i=\text{min par}}^{\text{max par}} s(i)_{\text{ref}}} \right) \tag{3.12}$$

with constraints:

$$K(i) \ < \ K_{\max}$$

and

$$\text{RobotFootprint}(\mathbf{p}_i) = 0 \quad \text{for all } i$$

where

- ω_c and ω_L are the weights assigned to the curvature change term and the length term, respectively. These weights determine the relative importance of curvature change and path length in the overall evaluation.
- $K_{\max}$ the maximum tolerated curvature
- $\mathbf{p}_i$ represent the i-th scan point in space.
- RobotFootprint($\mathbf{p}_i$) be a boolean function that returns 1 if the point $\mathbf{p}_i$ lies within the robot's footprint (polygon) and 0 otherwise.

The algorithm that measures the path quality is detailed through Pseudo Code 3.

3.3.3 Results

Both evaluation approaches were tested in the cloned simulation environment. Tests were ran on 10 different spline paths that were generated with transition points scattered in the station 3.16. Different locations were used to stress the metric outcomes by creating good and bad paths. The Exponential Weighted Path Evaluation was tested with α = 2, β = 0.0007, ω_c= 0.7 and ω_L=0.3. The results are shown in figure 3.17: The bar chart reflects the Evaluation score of each spline path from figure 3.16. Each bar color reflects the same color spline. The Normalized Weighted Path Evaluation was ran for two tests on the same splines set with ω_c= 0.7 and ω_L=0.3 as well as ω_c= 0.5 and ω_L=0.5. The results are shown in figures 3.18 and 3.19

For the Exponential Weighted Path Evaluation, the results are correct. It is easy to differentiate poor-quality paths. For example, the Blue spline was used as the reference path, while the Orange, Gray, and Purple paths were intentionally created as bad paths. The Orange and Gray paths generate significant curvature near the destination, making

it challenging for the vehicle to navigate and arrive in the correct position and orientation given the narrow aisle that it has to turn and navigate in as shown by figure 3.20. Given the high curvature and the proximity to the destination, the truck decelerates and moves very slowly. Changes that it stops at the correct orientation that allows it to pickup the pallet are very low. Such path have to be avoided at all costs. The Purple path is longer and curved at the starting area. Furthermore, the Brown path demonstrates the best overall fitness. Although it is short, it introduces high curvatures at the transition and destination areas. Compared to the Blue and Red paths, the Brown path is shorter but less smooth. The Exponential Evaluation also favors the Dark Green path to the Pink one, even though the concentrated curvature at the end of the green path is high. In conclusion, this evaluation method is very sensitive to path length and less sensitive to high concentrated curvatures. This is due to the high factor of the length term, in the range of thousands of millimeters, and the low factor of the curvature term which is around the magnitude of 10^{-3}. It is inconvenient to change the factors ω_c and ω_L by increasing ω_c and decreasing ω_L as the difference becomes huge while it is mandatory to attribute importance to both factors. Besides, this Metric is very sensitive to the change of the α and β factors. These factors must be carefully tuned to fit all use cases and account for varying station sizes and configurations. However, finding factors that are scalable across different use cases is not straightforward.

On the other hand, The Normalized Weighted Path Evaluation, chose the Red path as the best fitness, outperforming the Blue pattern path. This is due to the remarkable less curvature of the Red path at the start and transition locations due to its proximity to the transition polygon and distance to the destination. These factors allow for smooth driving along the path. Comparing figures 3.18 and 3.19, the approach of outweighing the curvature over the length outperforms the equal weights approach. The Pink path is visibly better than the Brown due to better smoothness and the first weights approach discriminates them better. Given the challenges associated with driving along smooth paths, it is a strategic decision to prioritize the curvature term over the length term. Navigating a slightly longer path is far less difficult than handling a path with sharp curves. This approach is not affected by the difference of range between length and curvature metrics given that they are each divided respectively by the reference path's length and curvature. This not only balances out the two terms but also makes it fit all use cases and account for varying station sizes and configurations given that the reference path is always relevnat to the present station.

As a final point, the champion approach is the Normalized Weighted Path Evaluation with ω_c= 0.7 and ω_L=0.3 given by equation 3.12.

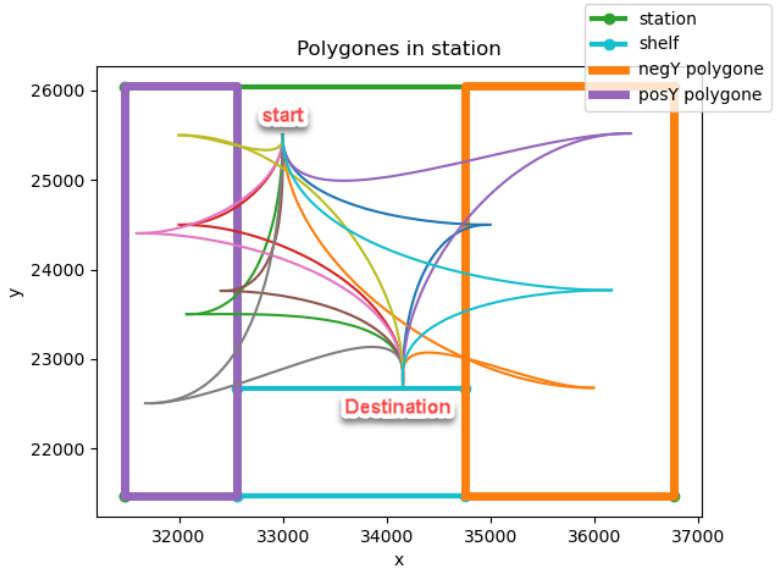

Figure 3.16: Test Results on the Simulated Environment: Multiple Splines Visualization

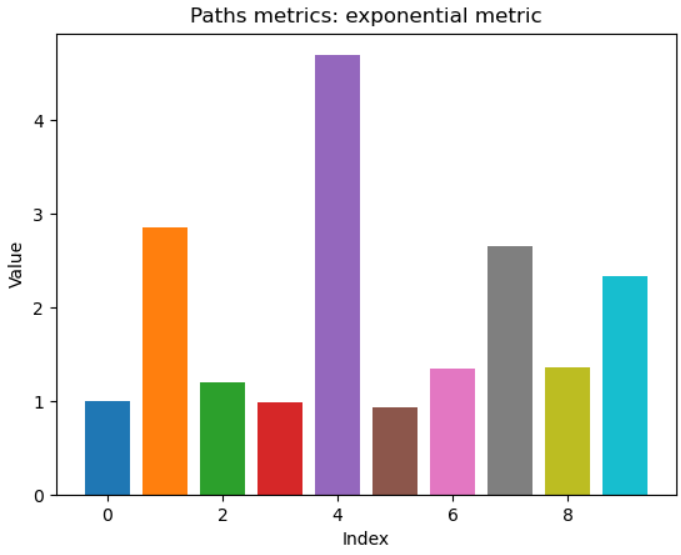

Figure 3.17: Test Results on the Simulated Environment: Evaluation results of the Exponential Approach

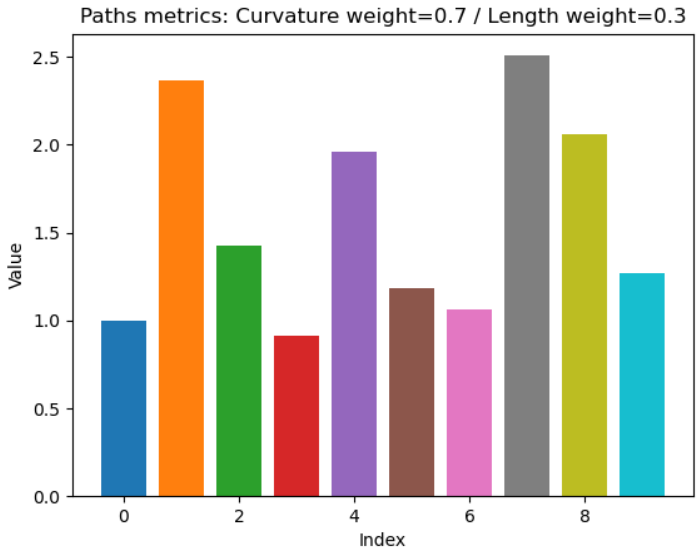

Figure 3.18: Test Results on the Simulated Environment: Evaluation results of the Normalized Approach with weighing out the curvature

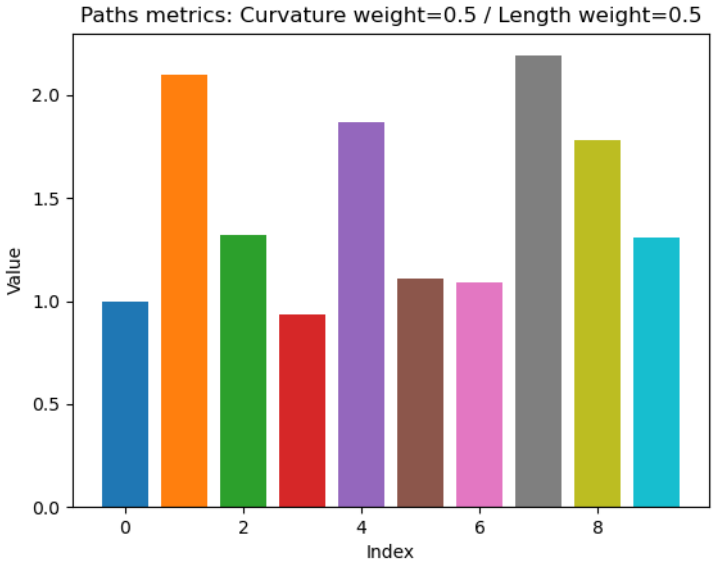

Figure 3.19: Test Results on the Simulated Environment: Evaluation results of the Normalized Approach with equal weights

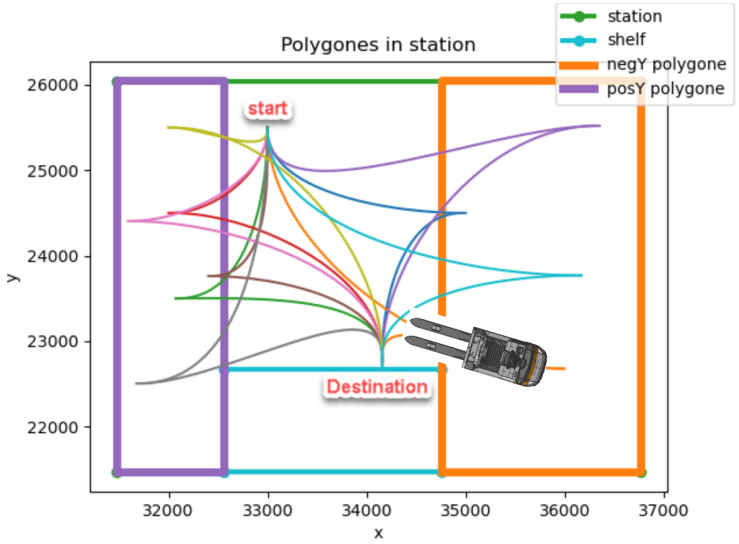

Figure 3.20: Navigating high curvatures in narrow areas

Algorithm 3: Path Evaluation Algorithm

Data: Spline Path, set of scan points P
Result: Evaluation result

```
Initialize Metric;
if local_max_curvature > maximum_curvature_ then
    metric.curvature_change_ ← curvature_KnockOut_factor ·
      local_curv_change;
end
else
    metric.curvature_change_ ← local_curv_change;
end
metric.length_ ← metric.calculate_length_metric(gen);
for each point p_i in P do
    test ← RobotFootprint(p_i);
    if test == 0 then
        result ← obstacle_KnockOut;
    end
    else
        result ← EvaluationFunction(curvature, length);
    end
end
return result;
```

3.4 Path Optimization

After successfully developing the tools to partition the stations, create spline-based pattern paths, and evaluate their quality in line with the trucks' properties and target outcomes, the paths are ready for the optimization phase. **The input of this section is a range of paths along with their evaluation scores and the output is the optimal path to be driven.**

3.4.1 Utility

The utility of the path optimization process lies in its ability to identify the most efficient and effective path for the vehicle to follow, based on various criteria. By considering key factors such as path length, curvature, and smoothness, the optimization algorithm ensures that the selected path not only adheres to the vehicle's operational constraints but also minimizes risks associated with challenging maneuvers. This process plays a crucial role in improving overall system performance, as it helps reduce fuel consumption, wear on vehicle components, and driving complexity. Additionally, the optimized path leads to better time efficiency and safer operations, particularly in complex environments where tight turns or narrow spaces can hinder navigation.

3.4.2 Implementation

As discussed in section 3.1, The meta-heuristic approach is used to optimize the spline-based pattern paths. Based on the Literature review the choice was made around testing 5 meta-heuristic optimization algorithms then assessing each algorithm's performance. The algorithms are: Particle Swarm Optimization (PSO), Differential Evolution (DE), Genetic Algorithms (GA), Simulated Annealing (SA), and Ant Colony Optimization (ACO). Those algorithms were selected due to their capabilities of solving the local optima problematic and their applications in robotics path planning although considered recent for some of the algorithms.

The optimization process is detailed through Algorithm 4. After initializing the optimization problem-related parameters, The algorithm starts the evolution of the generations or the iterations. In each iteration it generates a fixed number of individuals. **The decision vector of each individual is the transition point. The optimization algorithm starts by placing random transition points inside the subpolygons**. First the initial individuals are generated. For Population based algorithms like GA and DE, generate an initial population of candidate points (x, y), for swarm-based algorithms like ACO and PSO, initialize particles or ants randomly, For Simulated Annealing (SA), start with a single solution point (x, y).
Each transition point is used to generate a path as explained in section 3.3.2 then evaluated through the method in section 3.3.3. The evaluation result is the fitness of the path that the optimization algorithm bases its results on. By now the initial generation is com-

pleted. Then, the algorithm iterates until the maximum number of iterations is met. The reason behind choosing a maximum number of iterations is that the fitness function is not tangible. It can vary according to many factors like station configuration and dimensions and navigation environments. Tuning a goal fitness that fits all cases can be limiting if it set higher than the possible optimal fitness and causes sinking in suboptimal solutions or can constrain the optimizer if set too low: the optimizer can quit without solutions. Then the algorithm proceeds to generate new points following the meta-heuristic algorithm:

- GA: Apply selection, crossover, and mutation to generate new points (x, y).
- DE: Create new trial points by combining the difference between vectors of points.
- PSO: Update positions of particles based on their velocities, influenced by the personal and global best.
- SA: Randomly perturb the current point (x, y) and generate a new path.
- ACO: Ants select the next points (x, y) based on pheromone trails and heuristic information.

Afterwards, it generates Corresponding Paths, evaluates Fitness of New Paths and updates the Population or Solution Set in relevance to the meta-heuristic algorithm:

- GA: Select the next generation of points based on fitness, using techniques like elitism or tournament selection.
- DE: Retain points with better fitness, replacing weaker ones.
- PSO: Update the personal best positions and global best positions of particles.
- SA: Accept or reject new points based on a probabilistic function that includes temperature.
- ACO: Update pheromone trails based on the quality of the paths found, and reduce pheromone levels globally (evaporation).

The last step of each iteration is to update the algorithm parameters:

- GA/DE/PSO: Adjust crossover rates, mutation rates, or particle velocities.
- SA: Reduce temperature according to the cooling schedule to reduce acceptance of worse solutions over time.
- ACO: Evaporate pheromone globally and reinforce the best paths with additional pheromone.

Once the stopping criteria is met, the algorithm returns the best path that it found.

For the application of these algorithms, Pagmo library was used. Pagmo is an open-source scientific library designed for C++ that specializes in optimizing massively parallel problems. It was developed between the Max Planck Institute for Astronomy and the Advanced Concepts Team, at the European Space Research and Technology Center [45]. Pagmo provides an exhaustive toolset of meta-heuristic optimization techniques including the original Algorithms and the improved versions.

Algorithm 4: Generic Optimization Algorithm with Path Evaluation

Input: Problem parameters, Population size, Max iterations, Algorithm-specific parameters

Output: GeneratedOptimalPath

Initialize population size, max iterations,algorithm-specific parameters;
Initialize Population of Points
Generate an initial population of candidate points (x, y).
Evaluate Fitness for Initial Paths
CreateEvaluationPath(x, y);
MeasureNormalizedMetric(path); **Repeat until Termination Condition Met (max iterations):**
begin
    Generate New Points
    Generate Corresponding Paths
    Evaluate Fitness of New Paths
    Update Population or Solution Set
    Algorithm-Specific Update Step
Return Best Point and its Path:
Return the best point (x, y) and its corresponding path found during the search process.

Algorithm	**Planning Time (Simple Environment) (ms)**	**Planning Time (Complicated Environment) (ms)**
Genetic Algorithm	57	73
PSO	59	67
ACO	56	65
Simulated Annealing	80	78
Differential Evolution	27	44

Table 3.1: Comparison of Planning Time for Different Algorithms in Simple and Complicated Environments (in milliseconds)

Algorithm	**Simple Environment**		**Complicated Environment**	
	Negative Fitness	**Positive Fitness**	**Negative Fitness**	**Positive Fitness**
Genetic Algorithm	X1	X2	Y1	Y2
PSO	X1	X2	Y1	Y2
ACO	X1	X2	Y1	Y2
Simulated Annealing	X1	X2	Y1	Y2
Differential Evolution	X1	X2	Y1	Y2

Table 3.2: Comparison of Fitness Values for Different Algorithms in Simple and Complicated Environments

Chapter 4

Chapter 4

chapter name

Introduction

General conclusion

Bibliography

[1] KION group website. URL: https://www.kiongroup.com/en/About-us/Management/ consulted on 19/07/2024

[2] KION group website. URL: https://www.kiongroup.com/en/About-us/KION-at-a-glance/ consulted on 19/08/2024

[3] STILL website, iGo Neo page. URL: https://www.still.de/en-DE/trucks/new-trucks/order-pickers/opx-20-25-igo-neo.html

[4] STILL website, iGo Neo page. URL: https://www.still.de/en-DE/trucks/new-trucks.html

[5] iGo Neo datasheet. URL: data.still.de/assets/products/Vehicles/Order_Pickers/OPX_iGo_neo/pdfs/OPX_EN_TD.pdf?mod=1681891889&s=baf72412ac0d268f13b017603ca95ea2

[6] Agile Scrum article. URL: https://top20review.com/phuong-phap-agile/

[7] Giuseppe Fragapane, René de Koster, Fabio Sgarbossa, Jan Ola Strandhagen, Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda, European Journal of Operational Research, Volume 294, Issue 2, 2021, Pages 405-426, ISSN 0377-2217, https://doi.org/10.1016/j.ejor.2021.01.019.

[8] Moshayedi, Ata Jahangir & Liao, Liefa & Kolahdooz, Amin. (2021). Gentle Survey on MIR Industrial Service Robots: Review & Design. 10.

[9] AMR vs. AGV Figure, Vecna Robotics Website. URL: https://www.vecnarobotics.com/amr-vs-agv/, consulted on 28.07.2024

[10] Lixing Liu, Xu Wang, Xin Yang, Hongjie Liu, Jianping Li, Pengfei Wang, Path planning techniques for mobile robots: Review and prospect, Expert Systems with Applications, Volume 227, 2023, 120254, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2023.120254.

[11] Marín, Pablo & Hussein, Ahmed & Martín Gómez, David & de la Escalera, Arturo. (2018). Global and Local Path Planning Study in a ROS-Based Research Platform for Autonomous Vehicles. Journal of Advanced Transportation. 2018. 1-10. 10.1155/2018/6392697.

[12] Mohd. Nayab Zafar, J.C. Mohanta, Methodology for Path Planning and Optimization of Mobile Robots: A Review, Procedia Computer Science, Volume 133, 2018, Pages 141-152, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2018.07.018.

[13] Zhang B, Zhu D. A new method on motion planning for mobile robots using jump point search and Bezier curves. International Journal of Advanced Robotic Systems. 2021;18(4). doi:10.1177/17298814211019220

[14] H. -T. Lee, H. -M. Choi, J. -S. Lee, H. Yang and I. -S. Cho, "Generation of Ship's Passage Plan Using Data-Driven Shortest Path Algorithms," in IEEE Access, vol. 10, pp. 126217-126231, 2022, doi: 10.1109/ACCESS.2022.3225571.

[15] Sertac Karaman and Emilio Frazzoli. 2011. Sampling-based algorithms for optimal motion planning. Int. J. Rob. Res. 30, 7 (June 2011), 846–894. https://doi.org/10.1177/0278364911406761

[16] c. Ichnowski and c. Alterovitz, "Parallel sampling-based motion planning with superlinear speedup," 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura-Algarve, Portugal, 2012, pp. 1206-1212, doi: 10.1109/IROS.2012.6386194.

[17] A. Elshamli, H. A. Abdullah and S. Areibi, "Genetic algorithm for dynamic path planning," Canadian Conference on Electrical and Computer Engineering 2004 (IEEE Cat. No.04CH37513), Niagara Falls, ON, Canada, 2004, pp. 677-680 Vol.2, doi: 10.1109/CCECE.2004.1345203.

[18] Q. Jin, C. Tang and W. Cai, "Research on Dynamic Path Planning Based on the Fusion Algorithm of Improved Ant Colony Optimization and Rolling Window Method," in IEEE Access, vol. 10, pp. 28322-28332, 2022, doi: 10.1109/ACCESS.2021.3064831

[19] Liu, Li-sang, Lin, Jia-feng, Yao, Jin-xin, He, Dong-wei, Zheng, Ji-shi, Huang, Jing, Shi, Peng, Path Planning for Smart Car Based on Dijkstra Algorithm and Dynamic Window Approach, Wireless Communications and Mobile Computing, 2021, 8881684, 12 pages, 2021. https://doi.org/10.1155/2021/8881684

[20] S. H. Tang, F. Kamil, W. Khaksar, N. Zulkifli and S. A. Ahmad, "Robotic motion planning in unknown dynamic environments: Existing approaches and challenges," 2015 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS), Langkawi, Malaysia, 2015, pp. 288-294, doi: 10.1109/IRIS.2015.7451627.

[21] Alterovitz, R., Koenig, S., & Likhachev, M. (2016). Robot Planning in the Real World: Research Challenges and Opportunities. AI Magazine, 37(2), 76-84. https://doi.org/10.1609/aimag.v37i2.2651

[22] Ohki, T., Nagatani, K., & Yoshida, K. (2012). Local Path Planner for Mobile Robot in Dynamic Environment based on Distance Time Transform Method. Advanced Robotics, 26(14), 1623–1647. https://doi.org/10.1080/01691864.2012.694648

[23] Heiden, E., Palmieri, L., Arras, K.O., Sukhatme, G.S., & Koenig, S. (2020). Experimental Comparison of Global Motion Planning Algorithms for Wheeled Mobile Robots. ArXiv, abs/2003.03543.

[24] Harris-Birtill , D & Harris-Birtill , R 2021 , Understanding computation time : a critical discussion of time as a computational performance metric . in A Misztal , P A Harris & J A Parker (eds) , Time in variance . The study of time , vol. 17 , Brill , Leiden , pp. 220-248 , The 17th triennial conference of the International Society for the Study of Time , California , California , United States, 2019. https://doi.org/10.1163/9789004470170_014

[25] Buniyamin, N., Ngah, W. W., Sariff, N., & Mohamad, Z. (2011). A simple local path planning algorithm for autonomous mobile robots. International journal of systems applications, Engineering & development, 5(2), 151-159.

[26] Masehian, Ellips & Sedighizadeh, D.. (2007). Classic and Heuristic Approaches in Robot Motion Planning - A Chronological Review. World Academy of Science, Engineering and Technology. 29.

[27] C. Chen, M. Rickert and A. Knoll, "Combining space exploration and heuristic search in online motion planning for nonholonomic vehicles," 2013 IEEE Intelligent Vehicles Symposium (IV), Gold Coast, QLD, Australia, 2013, pp. 1307-1312, doi: 10.1109/IVS.2013.6629647.

[28] Krüger-Basjmeleh, T. (2024). Befähigung autonomer, mobiler roboter zum one-shot-imitationslernen für Den vollständigen transport von paletten in lager- und produktionsbereichen Tino Krüger-Basjmeleh. Verlag Dr. Hut.

[29] Piegl, Les , and Wayne Tiller. The NURBS Book. 2nd ed., Springer Berlin, Heidelberg, 1995, https://doi.org/10.1007/978-3-642-59223-2

[30] B. Lau, C. Sprunk and W. Burgard, "Kinodynamic motion planning for mobile robots using splines," 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, MO, USA, 2009, pp. 2427-2433, doi: 10.1109/IROS.2009.5354805.

[31] Bui, Xuan-Nam, et al. "Shortest path synthesis for Dubins non-holonomic robot." Proceedings of the 1994 IEEE International Conference on Robotics and Automation. IEEE, 1994.

[W1] http://nurbscalculator.in/

[33] Elbanhawi, M., Simic, M. & Jazar, R.N. Continuous Path Smoothing for Car-Like Robots Using B-Spline Curves. J Intell Robot Syst 80 (Suppl 1), 23–56 (2015). https://doi.org/10.1007/s10846-014-0172-0

[34] Knott, G. D. (2000). Interpolating cubic splines. Birkhauser.

[35] Liu, S., Liu, P. Benchmarking and optimization of robot motion planning with motion planning pipeline. Int J Adv Manuf Technol 118, 949–961 (2022). https://doi.org/10.1007/s00170-021-07985-5

[36] Zhang B, Zhu D. A new method on motion planning for mobile robots using jump point search and Bezier curves. International Journal of Advanced Robotic Systems. 2021;18(4). doi:10.1177/17298814211019220

[37] Bilal, Millie Pant, Hira Zaheer, Laura Garcia-Hernandez, Ajith Abraham, Differential Evolution: A review of more than two decades of research, Engineering Applications of Artificial Intelligence, Volume 90, 2020, 103479, ISSN 0952-1976, https://doi.org/10.1016/j.engappai.2020.103479.

[38] D. Tamilselvi, S.M. Shalinie, A.F. Thasneem, S.G. Sundari, Optimal Path Selection for Mobile Robot Navigation Using Genetic Algorithm in an Indoor Environment, In: P.S. Thilagam, A.R. Pais, K. Chandrasekaran, N. Balakrishnan (eds), Advanced Computing, Networking and Security. ADCONS 2011, Lecture Notes in Computer Science, vol 7135, Springer, Berlin, Heidelberg, 2012, https://doi.org/10.1007/978-3-642-29280-4_31.

[39] Ziadia, Abdelhamid & Mohamed, Habibi & Kelouwani, Sousso. (2023). Machine Learning Study of the Effect of Process Parameters on Tensile Strength of FFF PLA and PLA-CF. Eng. 4. 2741-2763. 10.3390/eng4040156.

[40] H. Miao and Y. -C. Tian, "Robot path planning in dynamic environments using a simulated annealing based approach," 2008 10th International Conference on Control, Automation, Robotics and Vision, Hanoi, Vietnam, 2008, pp. 1253-1258, doi: 10.1109/ICARCV.2008.4795701.

[41] Lone, Tahir & Wadood, Abdul & Gholami Farkoush, Saeid & Yu, Jiangtao & Kim, Changhwan & Bong, Rhee. (2019). An Improved Optimal Solution for the Directional Overcurrent Relays Coordination Using Hybridized Whale Optimization Algorithm in Complex Power Systems. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2925822.

[42] Septfons, Baptiste , Chehri, Abdellah , Chaibi, Hasna , Rachid, Saadane , Tigani, Smail. (2022). Swarm Robotics: Moving from Concept to Application. 10.1007/978-981-19-3455-1_14.

[43] Tang B, Zhu Z, Luo J. Hybridizing Particle Swarm Optimization and Differential Evolution for the Mobile Robot Global Path Planning. International Journal of Advanced Robotic Systems. 2016;13(3). doi:10.5772/63812

[44] Mukherjee R, Emond BR, Junkins JL. Optimal Trajectory Planning for Mobile Robots using Jacobian Elliptic Functions. The International Journal of Robotics Research. 1997;16(6):826-839. doi:10.1177/027836499701600607

[45] Biscani, F., & Izzo, D. (2020). A parallel global multiobjective framework for optimization: pagmo. Journal of Open Source Software, 5(53), 2338. doi: 10.21105/joss.02338.

[46] AMSC USA website. URL: `https://www.amsc-usa.com/blog/warehouse-layout-design/` consulted on 31/08/2024

[47] StevenM. LaValle. Rapidly-Exploring Random Trees: A New Tool for Path Planning. Ames, IA 50011 USA, 1999.

Annexes