People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
A. Mira University of Bejaia
Faculty of Exact Sciences
Department of Computer Science

Final Study Thesis

In view of obtaining the Master of Research in Computer Science
Specialization: Advanced Information Systems

E-COMMERCE BASED ON ROBOT OF DELIVERY

Prepared by:
MR. Badaoui Khalil Brahim

In front of the jury composed of:
President:Dr MOHAMEDI Mohamed.
Examiner:Dr BOUADEM Nassima.
Supervisor:Dr CHIBANI Samia.
Co-supervisor:Dr EL BOUHISSI Houda.
Representative of the Incubator:Dr MOHAMEDI Mohamed.
Representative economic : Mr KHIREDDINE Rahim.

Promotion 2024-2025

Acknowledgments

As we conclude this work, I would like to express my deep gratitude and sincere appreciation.

First and foremost, I thank Almighty God for granting me the strength, determination, and perseverance needed to complete this endeavor. My heartfelt thanks go to my parents, whose unwavering support has been a constant source of inspiration and motivation throughout this journey.

I am deeply grateful to my advisors, Mrs. Houda El Bouhissi and Mrs. Chibani Samia, for their continuous support, insightful guidance, and trust in me. Their invaluable collaboration and encouragement throughout the writing of this thesis have been instrumental in its completion.

I extend my appreciation to the members of the jury for taking the time to evaluate my work. Finally, I express my sincere gratitude to all the faculty and staff of the Computer Science Department at the University of Béjaïa for their invaluable support and contributions to my academic journey and professional development.

Dedication

In grateful recognition of Almighty Allah's guidance and blessings, and with heartfelt appreciation for the unwavering support of my beloved family and friends, I dedicate this thesis:

To my dear mother and father, whose endless love and sacrifices have been the foundation of my journey.

To my beloved sisters and my brother Houssam the hero.

To my friends: Laid, Soultan, Mahdi, Haron, Addel, Noufel, Youba, Zaki, Sabrina, Chabiba Zina Ihhedaden. I am forever indebted to you. Your strength and resilience inspire me every day, and your encouragement fuels my ambition and drives me to strive for excellence.

Last but not least, I want to thank ME, for believing in myself, for doing all this hard work, and for never quitting.

Contents

List of Figures

List of acronyms

APF : Artificial Potential Field
BFO : Bacterial Foraging Optimization
CS : Cuckoo Search
DC : Direct Current
DWA : Dynamic Window Approach
FL : Fuzzy Logic
IMOA* : Improved Multi-Objective A* Algorithm
GPS : Global Positioning System
GSA : Gravitational Search Algorithm
LSTM-FTR : Long Short-Term Memory with Feature-Temporal Representation
MGWO : Modified Grey Wolf Optimizer
NN : Neural Network
OSM : Open Street Map **PSO** : Particle Swarm Optimization
PDDs : Personal Delivery Devices
RL : Reinforcement Learning
RBF : Radial Basis Function
SBM : Stochastic Block Model
YOLO : You Only Look Once

Abstract

Safe and efficient navigation is among the key challenges for sidewalk delivery robots operating in urban environments. This paper proposes a hybrid path-planning approach using the global planning capability of the A* algorithm combined with local obstacle avoidance using the Artificial Potential Field (APF) method.

Performance tests were carried out in simulated urban scenarios with a wide range of algorithms. The two most promising ones were A* and APF. A* brought computation time down to 1.2 seconds, only 8% of the total simulation time for the computation of a 12.4 km path. It outperforms Dijkstra, Greedy, BFO, PSO, ACO, and GWO in both path efficiency and cost. APF managed to avoid obstacles in 5.2 seconds for a 30-minute simulation video, while other approaches required between 40 and 100 seconds.

Keywords: sidewalk robot, autonomous navigation, path planning, A*, APF, hybrid approach, last mile delivery robot.

Chapter 1

Introduction

1.1 Introduction

Sidewalk delivery robots, formally known as Personal Delivery Devices (PDDs) [3], are small, autonomous vehicles designed to execute the "last mile" of delivery, transporting goods like meals and groceries along sidewalks at walking speed. Their navigation relies on advanced technology, including GPS, cameras, and sensors, which not only enable the robots to avoid obstacles but also possess potential applications in broader assistive systems, such as helping blind individuals navigate home safely. The primary motivation for their development is to offer a cleaner, more efficient, and economical alternative to traditional vehicle delivery [2], they reduce congestion and pollution as they are battery powered, lower long-term operational costs, and increase delivery speed and reliability by avoiding street traffic. Furthermore, they provide a safe, contactless option and assist individuals with mobility issues [1, 2]. However, their integration faces challenges concerning ensuring sidewalk safety and accessibility for all pedestrians, especially those with disabilities, alongside concerns about potential congestion, poorly maintained infrastructure, and privacy issues related to data collection. Despite these hurdles, the technology continues to expand, and these robots are increasingly common sights in cities and on university campuses [2].

1.2 Thesis organization

This work is structured into five chapters, with each chapter presenting essential information related to the research.

Chapter tow: Background and State of the Art This chapter provides the theoretical foundation of the project by reviewing the system's software and robotic components. It surveys previous work on both global and local path planning methods. The chapter concludes with a comparative analysis and critical discussion of existing path planning approaches.

Chapter three: Proposed Approach This chapter presents the main contribution of the project: the development of a new pathfinding solution. The focus is on selecting effective algorithms for shortest path planning and obstacle avoidance, and then integrating them into a single robust method.

Chapter four: Evaluation and Results This chapter describes the experimental workflow, including the collection of real-world city data for training and testing the pathfinding models. It also explains the creation of a video-

based dataset using camera feeds for detection modules. The results section presents system simulations, performance visualizations, and quantitative metrics such as path accuracy, collision avoidance rate, and processing time, followed by an evaluation of strengths and weaknesses,and showing interfaces of the web application.

Chapter five: Conclusion This chapter summarizes the project's achievements in pathfinding and intelligent navigation. It highlights the limitations encountered, such as hardware constraints, data variability, and real-time processing challenges. Finally, it outlines future work directions, including recommendations for hardware improvements, better datasets, and algorithmic enhancements to move the prototype toward real-world deployment.

Chapter 2

Background And Stat Of Art

2.1 Introduction

Sidewalk delivery robots represent a pivotal solution to the expensive and environmentally taxing challenge of last-mile delivery. These small, autonomous vehicles, resembling rolling coolers, navigate urban sidewalks at walking speed using sophisticated sensor suites including cameras, radar, ultrasonic sensors, and high precision GPS to safely perceive and interact with their surroundings [2, 4]. While commercial implementations by companies like Starship.

Technologies and Amazon have successfully proven the viability of this technology [5], achieving widespread deployment requires tackling significant technical hurdles identified in the State of the Art.

2.2 Robot and Web

Getting your groceries is becoming much easier thanks to websites and small sidewalk robots working together. First, you use a simple website to pick out everything you need, just like shopping online. After you pay, a message is sent to the grocery store or a special warehouse where your items are packed up. Then, a small, self-driving robot is loaded with your order and travels along the sidewalks to your home, and you can often track its journey on your phone so you know exactly when it will arrive.

2.2.1 Hardware List for Delivery Robot

To construct a complete autonomous delivery robot, you must begin with a dual-processor core control system, featuring a high-level processor like an NVIDIA Jetson Orin Nano Developer Kit running off a high-speed microSD card (128GB or more, U3/A2 rated) to handle the Robot Operating System (ROS), sensor fusion, and AI-driven object detection, which delegates real-time actions to a microcontroller such as an Arduino Mega 2560 or Teensy 4.1 for precise PWM motor control and reading encoder data. This brain perceives the world through a comprehensive sensor suite led by a LIDAR (like a Slamtec RPLIDAR S1) for mapping, a stereo camera (such as a Stereolabs ZED 2i or Intel RealSense D435i) for object classification and depth perception, a high-precision RTK GPS module (requiring both a base station and a rover unit) for centimeter-level sidewalk positioning, a 9-DOF IMU (like a

Bosch BNO085-based unit) for stable orientation, and an array of 4 to 8 JSN-SR04T waterproof ultrasonic sensors for close-range obstacle avoidance. This hardware is mounted on a rigid chassis built from aluminum extrusion (e.g., 2020 or 3030 series) and CNC-milled plates, featuring a compliant suspension system like a double bogie design to ensure its all-terrain rubber wheels (130mm+ diameter) maintain traction. Propulsion is driven by high-torque DC geared motors with integrated quadrature encoders, which are managed by high-current H-bridge motor drivers (like the BTS7960) to handle stall currents safely. The entire system is energized by a high-capacity Li-Po or Li-Ion battery pack (typically 4S to 6S, 10,000mAh to 22,000mAh), with power regulated through multiple DC-DC buck converters and distributed via heavy-gauge silicone wire (12-16 AWG) using robust XT60 or XT90 connectors. Finally, for safe human-robot interaction and remote operation, the robot must include physical E-Stop buttons, addressable RGB LED strips (like NeoPixel WS2812B) for visual signaling, a weatherproof speaker with an amplifier module for audible alerts, and a 4G/5G LTE modem for remote telemetry and receiving missions [30].

2.2.2 Software List for Delivery Robot

The software involved in this project includes path planning, artificial intelligence, computer vision, and development. This merges algorithms from global navigation together with obstacle avoidance and detection and visual processing frameworks-aided by libraries.

Bio-inspired algorithms

In general, bio-inspired algorithms are problem-solving methods that mimic nature's successful strategies to tackle incredibly complex optimization and search problems. Instead of using rigid mathematical logic to find a single, perfect answer which is often impossible or would take too long they use processes like evolution or swarm intelligence to efficiently find a "good enough" or near-perfect solution. The typical approach involves starting with a population of many random potential answers, evaluating how "fit" each one is, and then repeatedly applying a nature-inspired rule to guide the population toward better results.

Famous examples include Genetic Algorithms, which copy evolution's "survival of the fittest"; Ant Colony Optimization, which mimics how ants

use pheromone trails to find the shortest path; and Particle Swarm Optimization, which models the coordinated flocking of birds to find the best spot in a landscape. Ultimately, they are powerful, intelligent shortcuts for navigating vast problem spaces where traditional methods would fail [6].

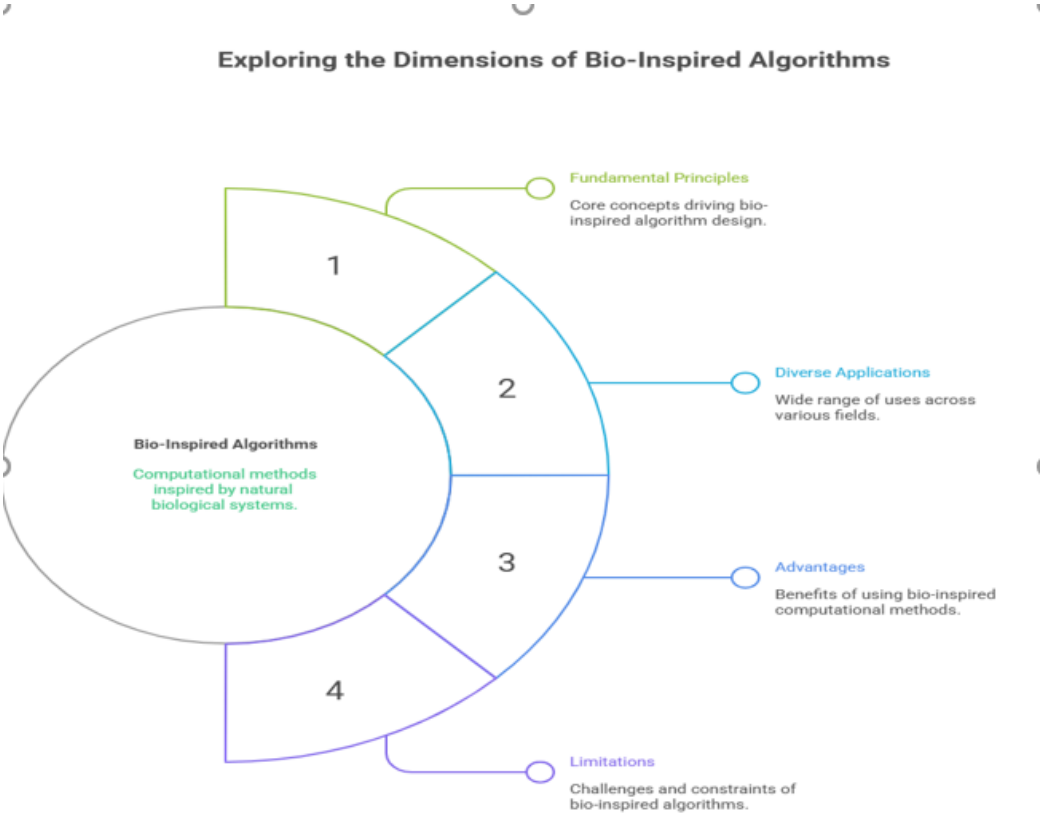


Figure 2.1: Bio-inspired Algorithm

Classic pathfinding algorithm

Classic pathfinding algorithms are the fundamental, logical methods computers use to find the absolute best route between a starting point and a destination within a network, like a road map or a game level. Unlike creative or random approaches, they are deterministic and guarantee an optimal solution.

One well-known approach in this category is the Greedy algorithm, which makes decisions step-by-step by always choosing the option that seems best at the moment. In pathfinding, a greedy algorithm moves toward the goal using only the "locally best" choice — usually by selecting the node that appears closest to the destination according to a heuristic. While greedy methods can be extremely fast, they do not guarantee the optimal path, because they focus only on short-term gain without considering the overall cost.

The most famous deterministic optimal solver is Dijkstra algorithm, which acts like an expanding circle of light, reliably finding the shortest path from a start point to all other points by always exploring the next closest location. However, it can be inefficient as it explores in all directions blindly. The more advanced and widely used A-star algorithm improves on this by adding a sense of direction; it not only considers the cost of the path traveled so far but also uses an educated guess to estimate the distance remaining to the goal, allowing it to intelligently focus its search and find the same shortest path much more quickly[7].

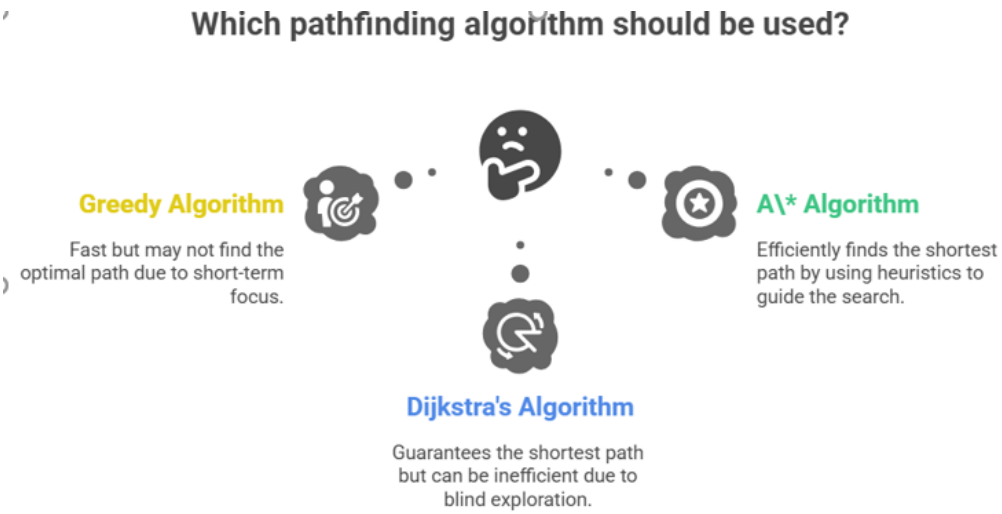


Figure 2.2: classic algorithm

2.3 Related Works

studies and systems that deal with problems similar to that proposed by the current research will be reviewed. It looks at the various existing approaches, pointing out their strengths, weaknesses, gaps, or limitations, therefore showing what is being done and what is still lacking, thus contextualizing and justifying the new contribution, The work falls into global and local path planning for balancing efficiency and safety. Global planning is intended to determine in general the route from start to destination using a map, ensuring that the robot takes the shortest or most efficient path. The local path planning allows the robot to make necessary changes and adjustments in view of real-time obstacles like pedestrians or unexpected objects so the robot can react and navigate safely. This division ensures the robot can reach its destination efficiently while avoiding collisions and adapting to changes in its environment.

2.3.1 Global path planning

Path planning examines methods for efficient navigation, combining global route planning with local obstacle avoidance. It includes classical, and intelligent algorithms, we study different types of algorithms presented in various articles, aiming to gather comprehensive information on global path planning.

Abdi et al 2020 [8] The prime objective of this work was to enhance the navigation capabilities of robots by developing the robust bacterial foraging (RBF) algorithm, which effectively targets various obstacle shapes, improving upon the traditional bacterial foraging optimization (BFO) algorithm that primarily manages circular obstacles. By utilizing data from multiple ultrasonic sensors, the RBF algorithm identifies the environmental layout and the obstacles' positions, allowing the robot to collaboratively determine its next move based on shared information from surrounding entities. This approach not only optimizes trajectory planning to minimize movement time and errors but also demonstrates high efficiency and performance across a range of unknown environmental scenarios, showcasing its potential for practical applications in robotic navigation.

M,Rhifky 2021 [9] The research indicates that while the Greedy algorithm can quickly identify a shortest path, it may fail to provide an optimal solution in some instances. In contrast, the A-Star algorithm generally outperforms

Greedy by finding better paths, though it too can miss the target under certain circumstances. Dijkstra algorithm consistently delivers the best results among the three, reliably finding optimal solutions, but it is slower due to its comprehensive cost comparisons. The Greedy algorithm's drawback lies in its short-sighted decision-making, while A-Star requires complex node data, and Dijkstra efficiency is hampered by its detailed evaluation process.

Alkhlidi et al 2021 [10] the Cuckoo Search algorithm effectively navigates obstacles in a complex environment. The results demonstrate the algorithm's ability to maintain efficient pathfinding while adapting to various obstacles. The study highlights the advantages of both MPSO and CS algorithms in robotic path planning, especially in intricate scenarios involving multiple robots. It underscores the need for robust algorithms capable of overcoming limitations encountered by traditional methods like the potential field algorithm, which struggles with irregular obstacles. The findings suggest that incorporating both MPSO and CS in future research may yield a hybrid approach that combines their strengths, optimizing robotic navigation and improving overall performance in more challenging environments.

Chen et Al 2021b [11] Path planning algorithms are broadly classified into three categories: traditional path planning algorithms, heuristic algorithms, and intelligent bionic path planning algorithms. Heuristic algorithms, known for their strong path search capabilities, are well-suited for discrete path topologies and include popular methods such as the A* algorithm, Dijkstra algorithm, and Floyd's algorithm. In contrast, the ant colony algorithm is an intelligent model characterized by its adaptability and cooperative behavior; it continuously optimizes paths by accumulating pheromones, guiding the search towards optimal solutions. Given that the robot path planning problem is a typical combinatorial optimization issue, leveraging the ant colony algorithm has proven valuable for improving effectiveness and efficiency in finding optimal paths. This adaptability and ability to learn from the environment make the ant colony algorithm a compelling choice for complex path planning situations.

Lui et Li 2022 [12] This article introduces a mobile robot path planning scheme that merges a modified gray wolf optimization (MGWO) algorithm with a situation assessment mechanism. The MGWO algorithm builds on the strengths of the original gray wolf optimization algorithm, enhancing population diversity through logistic chaotic mapping and achieving a balance between search and development via an adaptive control parameter adjustment strategy. Additionally, a static weighting average strategy is utilized to

update position and speed, thereby accelerating convergence. By integrating both global and local path planning, the situation assessment method facilitates local planning along the globally established path. Simulation results demonstrate that the proposed scheme offers superior optimization accuracy and stability compared to existing methods, effectively reducing path length and ensuring smoother trajectories. This hybrid approach prioritizes optimal pathfinding while accommodating dynamic obstacle avoidance, presenting clear advantages over using a single algorithm. Future work may involve task clustering based on the robot's task scale and battery capacity, enabling the autonomous conversion of large-scale tasks into manageable, time-sequenced actions, thereby enhancing the applicability of path planning in extensive environments.

Martins et al 2022 [13] This paper discusses the design and implementation of the IMOA-star algorithm for mobile robot path planning within a large workspace, implemented The IMOA-star method emerges as an effective alternative to the traditional A-star algorithm, as evidenced by performance assessments across four test cases. Key findings reveal that the IMOA-star algorithm drastically reduces average process time by approximately 99.98% compared to A-star, even with variable start and target points and new obstacles; it also enhances path smoothness by producing segment angles closer to 180 degrees, yielding a 24.75% improvement in smoothness metrics. Furthermore, IMOA-star reduces path length by an average of 1.58% and decreases the number of points by 83.45%, resulting in fewer turns and less torque on DC motors, ultimately improving battery efficiency. Collectively, these advantages indicate that IMOA-star significantly outperforms traditional A-star, establishing it as a promising option for efficient mobile robot path planning in expansive environments.

2.3.2 Short path planning

Local path planning or Obstacle avoidance focuses on enabling robots to safely navigate around dynamic and static objects in real time. It includes classical and intelligent algorithms, and we study different types of approaches presented in various articles, aiming to gather comprehensive information on local path planning and obstacle avoidance.

Guo et al 2021 [14] innovative local path planning fusion method for mobile robots that integrates LSTM neural networks, fuzzy logic control, and reinforcement learning, culminating in a network model designated as

LSTM_FTR. The proposed model exhibits significant improvements in path planning efficiency by approximately 81.88% compared to traditional fuzzy control algorithms thereby making it more suitable for real-time decision-making systems. Additionally, the reinforcement learning component facilitates independent exploration, enabling the model to learn new rules that enhance obstacle avoidance and facilitate shorter path planning in complex environments. The adaptability of LSTM_FTR allows it to rapidly optimize paths based on real-time sensor data, resulting in an increased success rate in path planning.

However, the approach currently faces limitations, such as being confined to simulation settings without incorporating practical application challenges. Real-world deployment will necessitate adjustments in parameter settings and enhancements in dynamic obstacle handling. Furthermore, relying solely on lidar sensors for environmental input may not provide a comprehensive understanding of real-world conditions; thus, future work will focus on enriching state inputs by integrating more environmental data and employing computer vision to better differentiate between obstacle types, ultimately refining the model's adaptability and performance.

Szczepanski et al 2022 [15] The proposed algorithm significantly enhances local path planning for Automated Guided Vehicles (AGVs) by reducing energy consumption by over 20%, minimizing goal-reaching time and path length, and enabling smoother movement while avoiding local minima, which improves overall operational efficiency. By integrating the original Artificial Potential Field (APF) approach with innovative modifications, the algorithm effectively addresses the challenges posed by heavy payloads, such as 300 kg pallets, ensuring collision-free navigation and maintaining battery capacity through smoother trajectories. As a result, this advancement is particularly relevant for industrial environments where energy efficiency and task execution speed are critical, with future research aiming to further optimize obstacles avoidance dynamically.

Hongwie et al [16] Path planning is a pivotal area in the field of mobile robotics, crucial for enhancing the autonomy of robots in dynamic and complex environments. Classical algorithms like A* and Dijkstra, although widely used, demonstrate critical weaknesses, particularly in adapting to varying conditions and complex scenarios involving human-robot and multi-robot interactions. These traditional methods can struggle with issues such as infinite loops and redundant searches, resulting in inefficiencies and reduced effectiveness in practical applications. Therefore, current research trends in-

dicate a shift towards hybrid approaches that combine various algorithms to leverage their strengths and mitigate individual shortcomings, as no single algorithm can tackle the diverse challenges presented by real-world environments.

Reinforcement learning (RL) has emerged as a promising avenue for future path planning research, with its ability to learn optimal behaviors through trial and error. Key areas of exploration include the design of effective reward functions that guide the learning process, the integration of RL with conventional path planning methods, and the application of RL in multi-agent settings to facilitate cooperative behaviors among robots. Despite its potential, RL is resource-intensive, prompting interest in brain-like intelligent systems that can learn efficiently with fewer resources, thus addressing the limitations of traditional RL approaches.

Moreover, as robotic applications expand, particularly in industrial automation and emergency scenarios, multi-robot cooperative path planning is gaining significance. Research is expected to focus on developing algorithms that enable multiple robots to work collaboratively, optimizing their positional communication and control mechanics to execute complex tasks seamlessly.

Future directions will likely emphasize improving the performance of path planning algorithms by reducing computation time and enhancing efficiency in finding global optimal solutions. Additionally, innovations in sensor technologies offer new opportunities for data-driven path planning; effectively fusing data from multiple sensors can provide richer environmental insights, enabling more accurate and reliable navigation for mobile robots. In summary, the convergence of algorithmic innovation, reinforcement learning applications, and advanced sensor integration will be integral to advancing the state of mobile robot path planning.

2.3.3 Local and global path planning (hybrid)

This stat of art study the Hybrid path planning combines global routing with local obstacle avoidance to guarantee efficiency and safety. In this, both classical and intelligent algorithms are integrated. We study various types of approaches presented in different articles, aiming at the complete gathering of information on hybrid path planning strategies. Lixing Liu , Xu Wang et al [17] The analysis of 105 research papers on environment modeling presented in this paper reveals significant trends in path planning algorithms,

particularly highlighting the dominance of the grid method, which accounts for 85.71% of the studied models. further illustrate that while classical algorithms primarily contribute to global path planning, dynamic algorithms like DWA excel in changing environments. The research also indicates that bionics-based algorithms are better suited for localized path planning due to their quick response times, whereas AI algorithms such as Neural Networks (NN) and Fuzzy Logic (FL) show equal interest but are more often applied in local scenarios. Notably, FL's usage in three-dimensional environments is limited due to complexity and memory requirements, but it frequently aids in optimizing other algorithms. Overall, advancements in AI and bionics-based methods suggest a gradual shift towards overcoming classical algorithms' limitations, such as local optima and computational inefficiency, particularly for real-time applications.

2.4 Analysis and Comparison

Examine the strengths, weaknesses, and limitations of existing algorithms and approaches. It includes classical and intelligent methods, and we study different types of approaches presented in various articles, aiming to gather comprehensive insights to identify research gaps and guide the development of improved path planning solutions.

2.4.1 Comparison criteria

We use these criteria to systematically and fairly compare different path planning approaches, ensuring that all important aspects of robot navigation are considered. They provide a structured framework to assess efficiency, safety, adaptability, and overall performance. By applying these criteria, we can identify the strengths and weaknesses of each method, validate results in controlled and real environments, and make informed decisions for selecting or improving algorithms that achieve reliable, practical, and optimized robotic navigation.

- Path Planning (local/global): Global path planning determines the overall route, while local path planning makes immediate adjustments to avoid obstacles.
- Algorithm: The computational method or set of rules the robot uses for decision-making, such as for navigation and obstacle avoidance.

• Path Smoothing: The process of refining a calculated path to ensure the robot's movement is fluid and mechanically efficient.

• Test (Simulation/actual): The method of validating robot performance, either within a virtual, computer-generated environment or in a physical, real-world setting.

• Data: The sensory information the robot collects and processes from its surroundings to understand its environment and location.

• Advantage: The inherent benefits or positive aspects of using this technology, such as increased efficiency or reduced environmental impact.

• Disadvantage: The inherent drawbacks or challenges associated with the technology, including technical limitations and operational hurdles.

• Evaluation: The systematic measurement and assessment of the robot's effectiveness and reliability against defined success criteria.

Authors	Path Planning		Dimension	Algorithm	Path Smooth	Test		Data	Adventage	Disadventag	Performance evaluation
	Global	Local				Simulation	Actual				
Lui et Li 2022	y	y	2	MGWO	n	y	n	/	Improved Optimization Accuracy. Faster Convergence . Dynamic Obstacle Avoidance.	Increased System Complexity. Higher Computational Cost. Performance May Vary Across Environments	shortest path
Martins et al 2022	y	n	2	IMOA	n	y	n	/	processing time . path smoothness. path length. the number of random points	More computationally intensive.	time /coste
M,Rhifky 2021	y	y	2	A*	y	n	n	/	good in various environments. Flexible.	Memory usage. Can be slow	optimal solustion
M,Rhifky 2021	n	y	2	Dijkstra	y	n	n	/	Easy to implement. One of best approach in solving the simple shortest path proble.	Slow on large graphs.	optimal solustion
M,Rhifky 2021	n	y	2	Greedy	y	n	n	/	can reduce the computational complexity. making the best choices	generates additional time costs.	optimal solustion
Szczepanski et al 2022	y	y	2	APF	y	n	y	/	minimizing goal-reaching time and path length, and enabling smoother movement ,	Local minima problem. Oscillations near narrow spaces or goal. Needs careful tuning of parameters.	time
Chai et al 2021	y	y	2	PSO	n	y	n	/	Simple to implement. Doesn't require much information about the problem. Effective at finding good approximate solutions to complex problems.	Discrete encoding PSO is continuous by default; paths must be encoded properly Validity Some generated paths might be invalid (collision or loops) Speed May be slower than A* or Dijkstra for small maps Parameter tuning	/

Chen et Al 2021b	y	y	2	ACO/APF	n	y	n	/	Strong for combinatorial problems . Flexible . Positive feedback loop.	Many parameters to tune. Computation can be heavy. Can get stuck in local optima if pheromone trails become too strong too quickly (called premature convergence).	get all solustion
Guo et al 2021	y	y	2	FL/NN(LSTM)	n	y	n	/	path planning efficiency . Can avoid obstacles in time and plan a shorter path in complex environments. e proposed model is adaptable, can self-learn.	e research is still in the simulation stage, without considering various problems in practical application. e environment input only depends on the lidar ranging sensor, which cannot accurately understand the real environment information	coste
Abdi et al 2020	y	n	2	BFO	n	y	n	/	The algorithm is easy to work with and optimizated.	The RBF algorithm struggles to escape local minima, causing the robot to halt near such points, highlighting a need for improved navigation. The accumulation of virtual obstacles around closed-form barriers hinders the robot's motion, revealing a flaw in the obstacle avoidance strategy.	time/path
Alkhlidi et al 2021	y	n	2	CS&PSO	n	y	n	/	Simple to Implement. Excellent performance in nonlinear optimization problems.	Can Be Slow. Not Always the Best Choice	coste

Figure 2.3: table of analyse and comparion

2.5 Discussion

The comparative analysis of path-planning algorithms between the years 2020 and 2025 indicates a clear trend from classical to meta-heuristics, hybrid, and AI-based methods. The classical algorithms, like A* and Dijkstra, are optimal but have speed and scalability issues . Meanwhile, meta-heuristic approaches such as MGWO, PSO, BFO, and CS offer faster convergence and adaptability at the cost of increased complexity and possible local optima. Hybrid and AI methods include combinations of ACO, APF, fuzzy logic, and neural networks, offering better adaptability and learning, although introducing higher computational burdens. While the majority of the studies address both global and local planning with the intent of making the routing efficient along with dynamic obstacle avoidance, a few focused on either global or local planning, which might restrict real-time applicability. Simulations dominate the testing aspects, with a very few implementations in actual environments, reflecting gaps between theoretical performance and practical deployment. Smoothness of the path is still a challenge for meta-heuristic methods, and

extra processing is required for feasible robot navigation. The overall analysis represents the trade-off between optimality, speed, and complexity and the practical feasibility issue arising from these, underlining the priority for future research to be integrating intelligent and adaptive algorithms with real-world operational constraint [19, 20, 26].

2.6 Conclusion

Sidewalk robot navigation has decidedly shifted towards complex, hybrid systems that combine AI and bio-inspired algorithms in both global and local path planning. While these advanced techniques show great promise in simulation, exhibiting such improvements as more than 81% increased efficiency and more than 20% energy savings due to smoother paths, the most critical conclusion drawn is the wide chasm between these theoretical achievements and demonstrated real-world capability. The biggest issue facing the field today is no longer solely one of algorithmic refinement but bridging this gap between simulation and reality via robust hardware integration, multi sensor fusion, and extensive testing in the unpredictable physical environments these robots are designed to master.

Chapter 3

The approach

3.1 Introduction

The central focus of the work is the finding of the best combination of algorithms to navigate a mobile robot in an environment with both static and dynamic elements. This is done in two parts: first, by analyzing and comparing several path-finding algorithms such as A* and Dijkstra to find the most efficient method to compute the shortest global route on a map that is already known [18, 19, 23]. Second, we study and then implement reactive obstacle avoidance algorithms based on real-time sensor data to navigate around unforeseen obstructions [24, 18]. At the center of our effort in seamlessly integrating these two algorithmic layers lies the ability of the robot to dynamically step out from its pre-planned optimum course, safely maneuver around obstacles, and then intelligently replan to the destination. This involves extensive simulation and real testing to analyze the performance of different algorithmic combinations in terms of efficiency of path, computational burden, and robustness in complicated scenarios [17].

3.2 Contribution

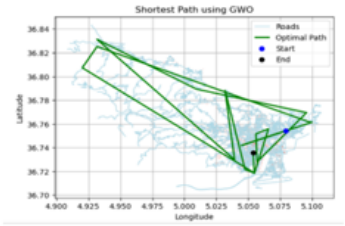
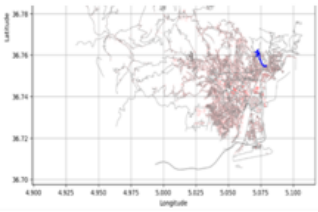
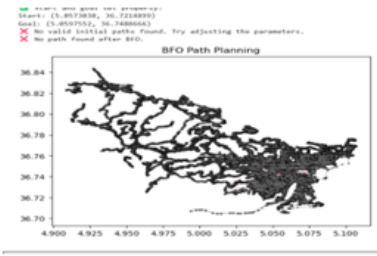
Our contribution focuses on the practical advancement of autonomous robot navigation by systematically evaluating and integrating established path finding and obstacle avoidance algorithms. We provide a comprehensive analysis of how global planners like A* and Dijkstra can be effectively combined with reactive methods such as the Dynamic Window Approach or Artificial Potential Fields to create a robust, hybrid system. Through rigorous testing in both simulated and real-world scenarios featuring dynamic obstacles, our work offers crucial insights into optimizing this integration, leading to more reliable, efficient, and adaptable navigation solutions for mobile robots operating in complex, unpredictable environments [27].

3.3 Best Algorithm For Global Path

we have worked with common methods like A* and Dijkstra, and also with algorithms that are based on nature, like ones that copy how ants find food. My work involved testing all of these on different kinds of maps to see how they performed in various situations. By doing this, I compared how:

.fast each algorithm was. .how much the total weight of edges as cost. .and how well it found a good route path length.

This process helped me learn the pros and cons of each method so I can choose the right one for challenges.

algorithem	result	evaluation
GWO		long time,not the shortest path ,high cost
ACO		long time, not the shortest path ,high cost
PSO		long time ,the shortest path, high cost
BFO		long time, didnt fond real path ,high cost

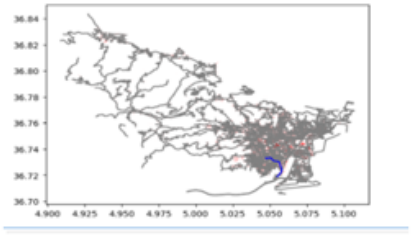
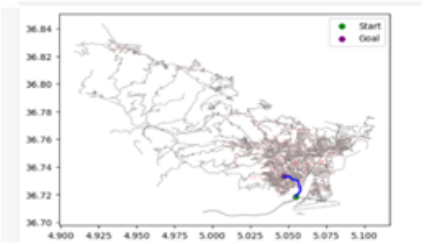
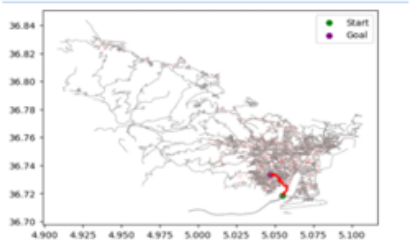
A*		short time , the shortest path, high cost
DJEKSTRA		short time , the shortest path, low cost
GREDY		short time ,the shortest path, low cost

Figure 3.1: Performance Comparison of Global Path Planning Algorithms on the Map

from the figure 3.1 and in the algorithm we applied the implementation available at github [31],A* algorithm and GREDY algorithm is the best methods for finding the shortest path on a map made of points and the paths between them based on time path and cost. Think of it like a ripple spreading out in a pond: it starts at one point and explores outwards step-by-step, always choosing the next closest location it hasn't visited. It keeps doing this until it has found the best route from the start to every other point on the map. The main reason it's considered the best is because it is guaranteed to find the absolute shortest path, not just a good guess. While other methods might be quicker for a single trip, A* is incredibly reliable and a perfect

choice when you need to know the definite best route from one place to all others.

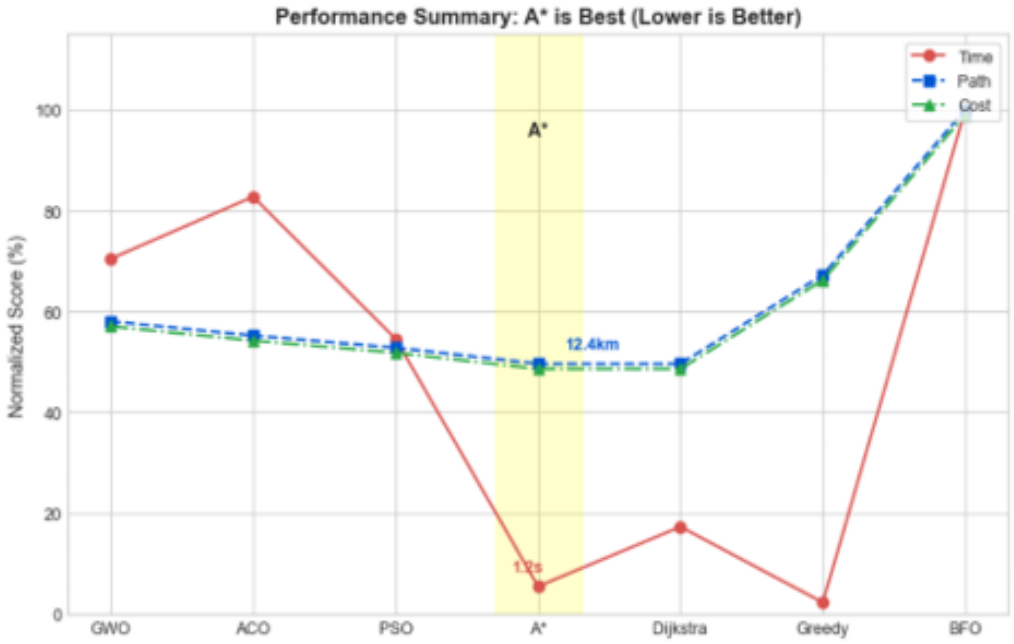


Figure 3.2: Evaluation of Algorithms Based on Time, Cost, and Path Length

The A* yielded the lowest computation time at about 1.2 seconds, far faster than other algorithms like ACO by 82%, GWO by 70%, or BFO by 100%, The A* path length is around 12.4 km, which is shorter results and GWO, ACO, PSO it is slightly better when compared to Dijkstra, Cost, which here represents the overall path efficiency, is also the lowest for A*, at about 50%, while for BFO and Greedy, it is the highest at close to 100%, indicating much less optimal paths, Dijkstra performs reasonably well, with path and cost scores quite similar to A*, but its computation time is higher by about 17% compared to A*, Greedy is very fast-actually, it is close to 0% of the time required but the length and the cost of the path are both very high, showing its short-sightedness, Overall, the numbers show that A* provides the best trade-off between speed, path length, and cost, making it the most efficient algorithm for global path planning in this case .

3.4 Best Algorithm For Avoiding Obstacles

Using a simple graph as simulation for voiding obstacles using some of bio-inspired algorithm combine from start point (green circle) goal (red circle) and 3 obstacles (blue circles) to evaluate the algorithm .

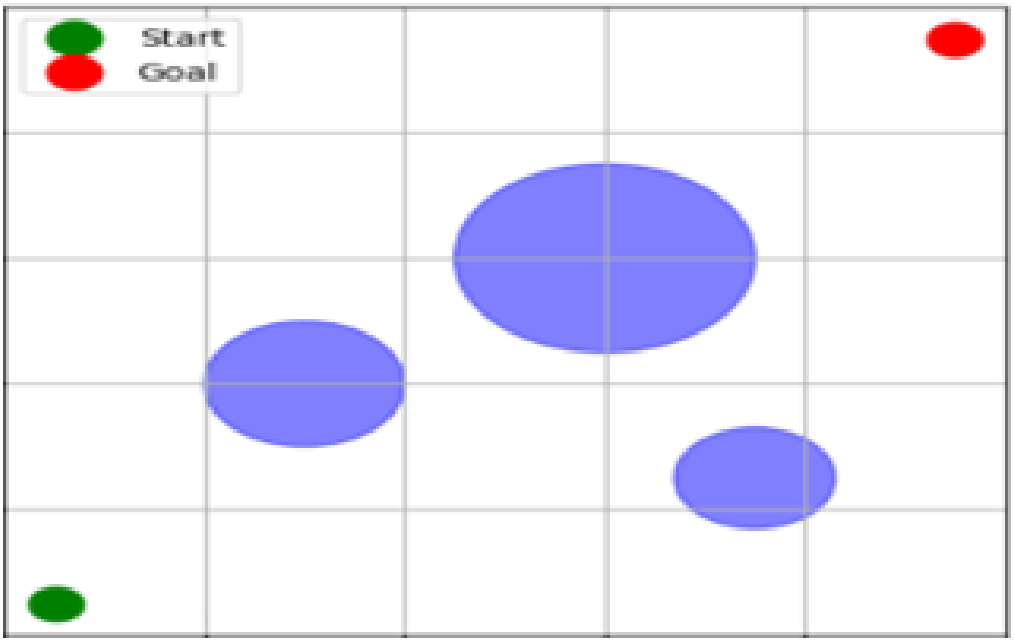


Figure 3.3: graph simulation for avoiding obstacles used

We evaluate this algorithms using this three tools:

- Time is how long the computer takes to find the answer.
- Shortest Path is how direct the route is.
- Cost is the final numerical score of that path's length, where a lower score is better.

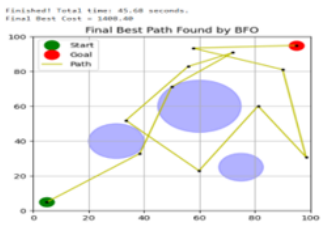
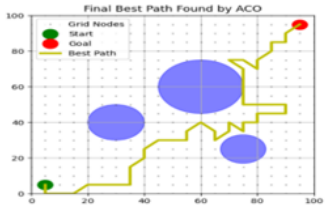
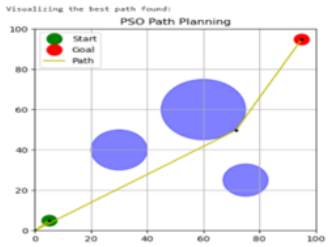
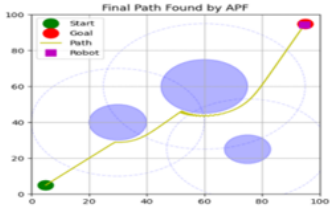
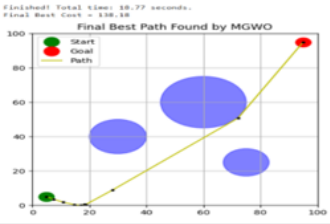
algorithem	result	evaluation
BFO		long time,not the shortest path ,high cost
ACO		long time, not the shortest path ,high cost
PSO		long time ,a short path, low cost
APF		short time , the shortest path, low cost
MGWO		short time , a short path, low cost

Figure 3.4: Result of algorithm for avoiding obstacle and finding shortest path

As discussed in the algorithm analysis, we applied the implementation available at github [31], PSO provided one of the best path among the bio-inspired methods, but we chose the APF algorithm because it was the best solution for our application. In the evaluation, although PSO found a low-cost and relatively direct path, it had a critical weakness in common with BFO and ACO a very long computation time. APF had the same primary advantage as PSO-predicting a smooth and direct trajectory but with much lower computational demand. APF treats the goal as an attractive force and obstacles as repulsive forces and thus can dynamically and reactively calculate a safe and efficient path, making it the best choice for a robot that needs to decide immediately about its actions in a constantly changing environment.

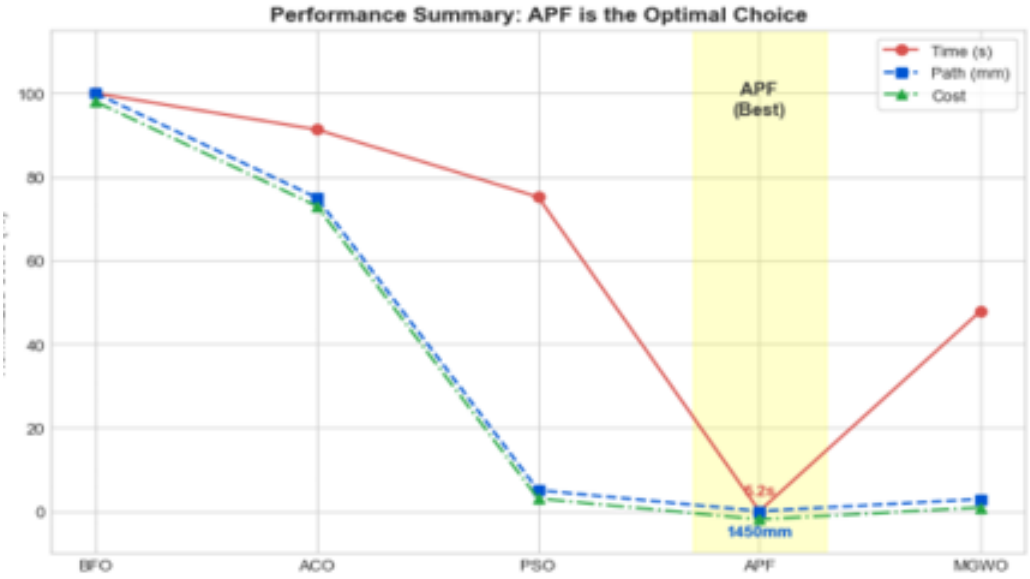


Figure 3.5: Graph of evaluation the algorithms based on time, cost ,path

This clearly indicates that APF is the best local path-planning technique among BFO, ACO, PSO, APF, and MGWO. The performance graph indeed shows the BFO and ACO starting with high normalized scores close to 100%, while PSO gradually slows down for each of the metrics. APF then leads to a dramatic improvement, with a computation time of only 2.2 s and a path length of about 1450 mm, driving the cost score to close to zero, the best

overall result in the chart. This dominant performance is highlighted in the yellow color shown. While MGWO does recover somewhat after APF, its results are far from matching the efficiency and stability of APF.

3.5 The Hybrid Algorithm Of (A* and APF)

The algorithm developed integrates two complementary path-planning techniques.

• Inputs: The system requires road network data defining potential routing paths, building or obstacle data, the coordinates of the start and goal positions, and control parameters specifying the strength of attractive and repulsive forces for local smoothing.

• A* for Path Planning: A* graph of valid connections is constructed from the road data, and any edges intersecting buildings are removed to ensure obstacle avoidance. The A* algorithm is then applied to generate a coarse, obstacle-free route represented by a sequence of waypoints.

• APF for voiding obstacles: The coarse A* waypoints are refined using Artificial Potential Fields. Attractive forces pull the robot toward each successive waypoint, while repulsive forces push it away from nearby obstacles, creating a continuous and natural trajectory.

• Outputs: The final output is a smooth, collision-free path from start to destination that is suitable for practical robotic navigation and autonomous delivery applications.

3.6 Conclusion

This study presented a hybrid navigation framework for mobile robots operating in environments with static and dynamic obstacles. Through a comparative evaluation of global path-planning algorithms, A* was identified as the most efficient method, offering the best balance between computation time, path length, and cost. For local obstacle avoidance, the Artificial Potential Field (APF) approach demonstrated superior real-time performance, providing smooth and efficient trajectories with low computational demand. By combining A* for global planning with APF for local obstacle avoidance, the proposed hybrid system achieves robust, efficient, and adaptive navigation, making it well suited for real-world autonomous robotic.

Chapter 4

Experimental Setup And Evaluation

4.1 Introduction

The concept of enabling a robot to get around in Bejaia rests on the core intelligence composed of key algorithms combined with city-specific data. The robot would make a map of the sidewalks in Bejaia using a SLAM algorithm while simultaneously tracking its position, then use this map for global path planning with an algorithm like A* that determines the best overall route. A local planner like the Artificial Potential Field (APF) is essential for immediate real-time maneuvering around pedestrians and obstacles, guided by perception systems using deep learning models, such as YOLO, in detecting and classifying objects from camera and LiDAR data. Importantly, the efficacy of these algorithms depends on a robust dataset specific to Bejaia, which one would have to create through a systematic process of collecting detailed geospatial information, sensor logs, LiDAR and camera, and a wide-ranging image library capturing unique streetscapes of the city, signage, and pedestrian behaviors to train and validate the robot for operation in Bejaia both safely and effectively [28].

4.2 Development Environment

The development environment for enabling autonomous navigation in Bejaia focuses on the core robotic intelligence integrated with city-specific data. On a global scale, it relies on detailed sidewalk maps obtained from OpenStreetMap, which provide comprehensive geospatial information for the city. Global path planning is performed using algorithms such as A*, which determine the optimal route across the mapped environment. These are complemented by local planners such as the Artificial Potential Field, which enable immediate, real-time maneuvers around dynamic obstacles, including pedestrians, using sensor inputs. Perception systems are a critical part of this pipeline, employing deep learning models such as YOLO to detect and classify objects from camera and depth map data. The effectiveness of these algorithms depends greatly on a robust and comprehensive dataset specific to Bejaia that must be systematically built. This dataset includes geospatial data, sensor logs, depth maps, camera recordings, and an extensive image library capturing streetscapes, signage, and pedestrian behaviors of the city. Together, these components provide a development environment capable of training, validating, and deploying the robot to safely and effectively navigate

Bejaia.

4.2.1 Hardware Environment

The experiments were conducted on a machine with the following listed specifications: PC1 Machine type: ASUS I5 8TH Processor: Intel(R) Core(TM) i5-8200Y CPU @ 1.30GHz 1.60 GHz RAM: 8,00 Go Operating system: Windows 11 Professional Robot Tools: Phone Camera ip webcom server tcp Phone GPS server tcp

4.2.2 Software Environment

The software environment of this project consists of Python as the main programming language, along with a selection of specialized libraries that support image processing, path finding, deep learning, and geospatial analysis. These tools collectively enable the implementation of the autonomous robot's navigation, perception, and decision-making systems in a structured and efficient manner.

4.2.3 Programming Language

Python was the primary programming language used for implementing the models .It is a popular programming language recognized for its ease of use and clear syntax. It backs various programming paradigms such as procedural, object-oriented, and functional programming styles. Python is extensively utilized across different fields including web development, scientific computing, data analysis, artificial intelligence, and automation.

The development stack integrates modern web technologies to create a robust and scalable application. React serves as the core front-end library, enabling efficient component-based UI development, while Next.js provides server-side rendering and routing capabilities for improved performance and SEO. User authentication and management are handled by Clerk, offering secure and customizable sign-up and login flows. Payment processing is seamlessly integrated via Stripe, ensuring reliable and secure transactions. Finally, Sanity is allowing flexible content management and real time updates across the application.

4.2.4 Python Libraries

Cv2: The robot's eyes, used for all tasks involving reading, processing, and displaying images and video.

Os: Interacts with the computer's file system to manage files and folders, like creating a directory for output frames.

Numpy as np: Provides the tools for high-speed mathematical operations on the grids of numbers that form images.

Torch: Acts as the robot's brain, loading and running the powerful deep learning model for object detection.

Json: Reads and writes simple configuration files to easily load settings like robot speed or file paths.

Time: Works like a stopwatch to measure how fast your code runs or to pause the program. import glob: Finds and creates a list of all files that match a specific pattern, such as every .jpg image in a folder.

Pathfinding: is a Python toolkit that provides efficient implementations of grid-based pathfinding algorithms, such as A*, Dijkstra, and others, allowing developers to easily compute optimal paths while handling obstacles and diagonal movement in 2D environments.

DiagonalMovement: A setting for the Algorithem finder that allows you to specify if the robot is permitted to move diagonally.

Geopandas: A library used to work with geographic data (maps) in a table-like format, making it easy to analyze coordinates, shapes, and spatial relationships.

Networkx: A library for creating and analyzing networks and graphs, commonly used to compute shortest paths and connections between nodes.

4.3 Simulation

We apply the hybrid A* and APF algorithms, explained in Chapter 3, and for this, we need appropriate data to work with and observe the results.

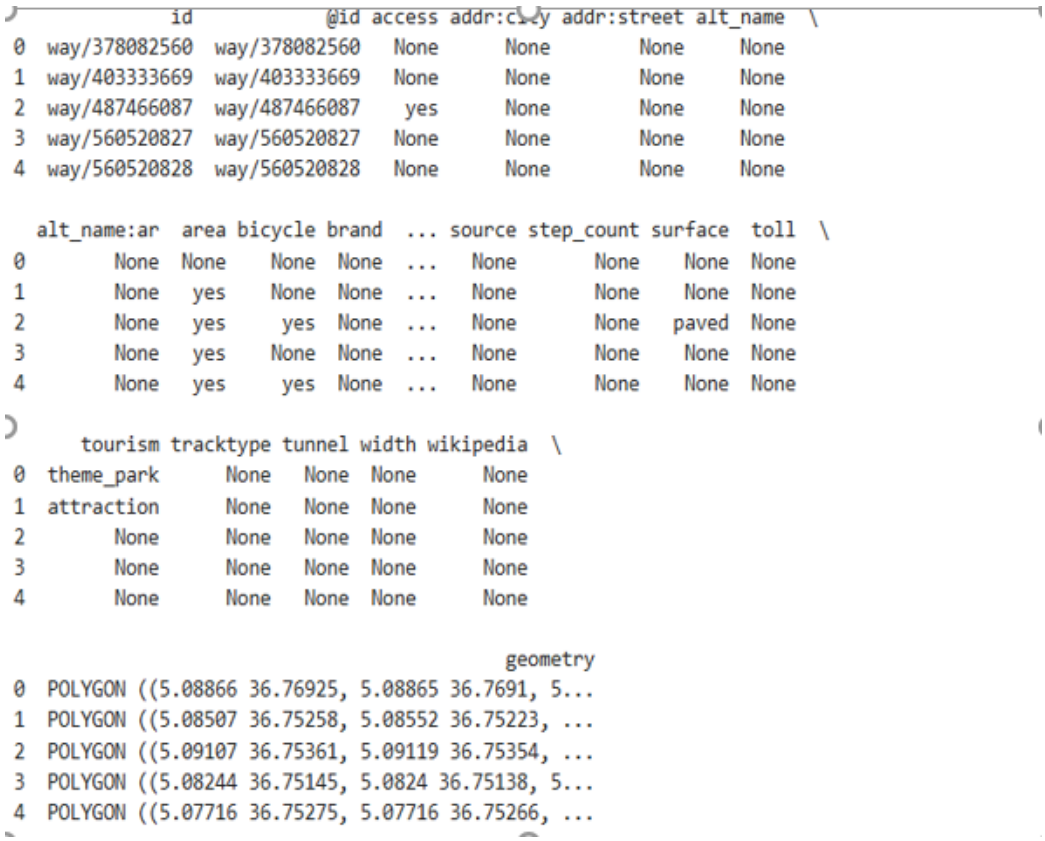
4.3.1 Data Collection

preparing or "training" map data for a pathfinding algorithm involves converting a real world location into a structured digital format that an algorithm can understand and navigate. This process is less about machine learning "training" and more about creating a detailed environmental model

with clear rules. It requires two fundamental types of information. First, you need data on all the traversable paths. This includes roads, sidewalks, open fields, or any area where movement is permitted. This information forms the backbone of the navigation map, creating a network or graph of interconnected points nodes and the pathways between them edges. This network defines all the possible routes the algorithm can consider. Second, you must identify all the non-traversable areas, which act as obstacles. These can be buildings, walls, bodies of water, dense vegetation, or any other physical barrier that blocks movement. This obstacle data is overlaid onto the path network, effectively telling the algorithm which areas are off-limits. By combining these two datasets, you create a comprehensive digital map . When tasked with finding the shortest path from a starting point to a destination, it searches through the network of traversable paths while strictly avoiding all the defined obstacle zones. The quality and accuracy of this initial data are crucial for the algorithm's ability to find a path that is not only short but also safe and practical in the real world [29].

4.3.2 Collection data of Bejaia

using OpenStreetMap (OSM) a free and editable map of the entire world, created by a global community of volunteers. Think of it as a "Wikipedia for maps," where anyone can contribute by adding or correcting data about roads, buildings, trails, and points of interest. Because it is completely open and free, its data is used by thousands of websites, mobile apps, and organizations for a huge range of purposes, including turn-by-turn navigation, humanitarian aid, and urban planning. This collaborative approach makes OSM an incredibly detailed and constantly updated resource that is available for anyone to use for any project without cost or restrictions.



	id	@id	access	addr:city	addr:street	alt_name	\
0	way/378082560	way/378082560	None	None	None	None	
1	way/403333669	way/403333669	None	None	None	None	
2	way/487466087	way/487466087	yes	None	None	None	
3	way/560520827	way/560520827	None	None	None	None	
4	way/560520828	way/560520828	None	None	None	None	

	alt_name:ar	area	bicycle	brand	...	source	step_count	surface	toll	\
0	None	None	None	None	...	None	None	None	None	
1	None	yes	None	None	...	None	None	None	None	
2	None	yes	yes	None	...	None	None	paved	None	
3	None	yes	None	None	...	None	None	None	None	
4	None	yes	yes	None	...	None	None	None	None	

	tourism	tracktype	tunnel	width	wikipedia	\
0	theme_park	None	None	None	None	
1	attraction	None	None	None	None	
2	None	None	None	None	None	
3	None	None	None	None	None	
4	None	None	None	None	None	

	geometry
0	POLYGON ((5.08866 36.76925, 5.08865 36.7691, 5...
1	POLYGON ((5.08507 36.75258, 5.08552 36.75223, ...
2	POLYGON ((5.09107 36.75361, 5.09119 36.75354, ...
3	POLYGON ((5.08244 36.75145, 5.0824 36.75138, 5...
4	POLYGON ((5.07716 36.75275, 5.07716 36.75266, ...

Figure 4.1: Data Collection

4.3.3 Training Data

Raw Data Acquisition and Exploration The process starts by gathering the necessary geographical data; Route/Path Data: represents streets, roads, and traversable pathways. Building/Point of Interest Data Represents landmarks or obstacles.

Initial Visualization The data is plotted after loading for a visual check, Roads in blue Buildings in red.

Node Definition → Each coordinate or intersection point becomes a node.

Edge Definition → Consecutive nodes are connected by edges.

Weight Assignment → Each edge gets a "cost" (often the distance between nodes).

These weights teach the algorithm what shortest means (Distance for

geometric shortest path, Time if you include speed limits or slope.),A well-trained graph = all nodes connected + accurate costs.

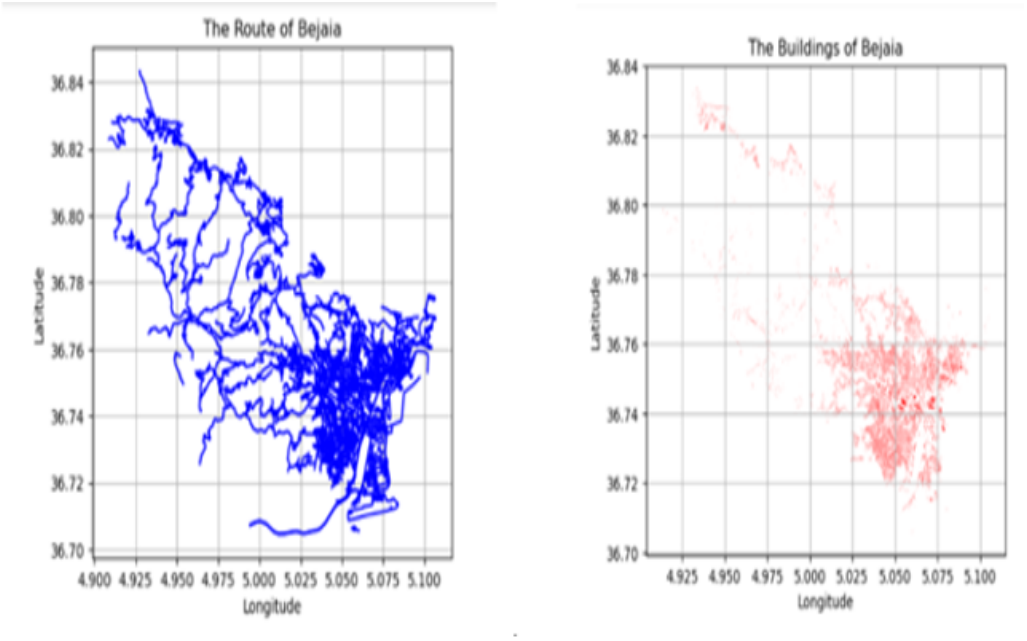


Figure 4.2: Visualisation Of Bejaia Routes And Building

4.3.4 Data From Cameras

Data from cameras acts as the robot's real time eyes, providing the critical situational awareness that a static map cannot. Through computer vision and machine learning, the camera feed is constantly analyzed to detect and classify immediate, unmapped obstacles such as pedestrians, pets, or temporary obstructions, enabling the robot to safely maneuver around them. This visual data is also essential for a deeper understanding of the environment, allowing the robot to differentiate between a sidewalk and a road, recognize crosswalks, and read traffic signals. Furthermore, by identifying unique visual landmarks like storefronts or lampposts, the camera helps the robot pinpoint its exact location on its pre loaded map, correcting for any sensor drift and ensuring precise navigation in a dynamic world.

From Video to Analyzable Frames

To effectively analyze robot's performance in Bejaia, we must first "copy" its operational videos frame by frame, a process achieved not by hand but by using powerful software tools like FFmpeg or programming libraries such as OpenCV. This technique converts a dynamic video file into a sequence of thousands of individual static images, allowing for a microscopic examination of the robot's behavior at any specific moment. Once extracted, these frames can be manually inspected to diagnose critical failures like why the robot misidentified a pedestrian or chose a certain path or they can be automatically processed by running your trained algorithms on each image to gather quantitative data, pinpointing exactly where perception failed and providing the detailed insights needed to systematically debug and improve the robot's safety and intelligence.

Figure 4.3: Camera for Collecting data

Video based Dataset (YOLO + Depth Map)

Preprocessing (Data Preparation) The purpose of preprocessing is to prepare raw video and sensor data for effective the obstacle avoidance system. This step ensures that all frames are synchronized, clean, and consistent enabling reliable detection and distance estimation.

YOLO (You Only Look Once) is used for real-time object detection. It identifies and classifies visible objects in each video frame, such as people, cars, bicycles, or static obstacles. This allows the system to understand what obstacles exist and where they are located within the scene. A depth map provides distance information for every pixel in the image. It tells how far each detected object is from the robot. This spatial information is

essential for APF, which depends on distance to compute repulsive forces closer obstacles produce stronger avoidance reactions. Integration between YOLO and Depth Map By combining YOLO's object detection with the depth map's distance information, the robot gains a full 3D understanding of its surroundings. Each detected object can be assigned a position and a depth value, enabling APF to calculate safe and efficient motion paths around obstacles. Summary

- YOLO detects and classifies objects.
- Depth Map measures how far those objects are.

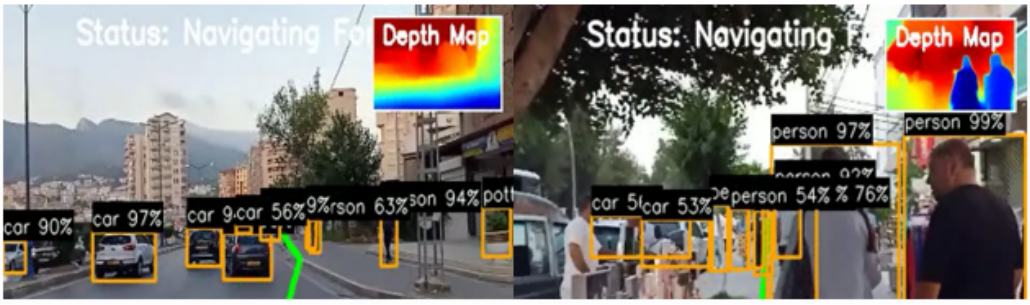


Figure 4.4: Application of YOLO + Depth Map on bejaia street

4.4 Simulation Results

These results demonstrate integration of a multi-stage robotics algorithm. The system first leverages pre-processed object detection data, loaded from JSON files, to draw orange bounding boxes and labels (e.g., car 9982neously, the core logic uses this obstacle information, along with depth maps, to dynamically generate a 2D "costmap" a grid where sidewalks are assigned a low cost and obstacles are assigned a high cost. The A* pathfinding algo rithm then calculates the optimal, safest route through this costmap, which is visualized as the blue line projected onto the image, clearly showing the robot's intelligent decision to plot a path forward that stays on the sidewalk.

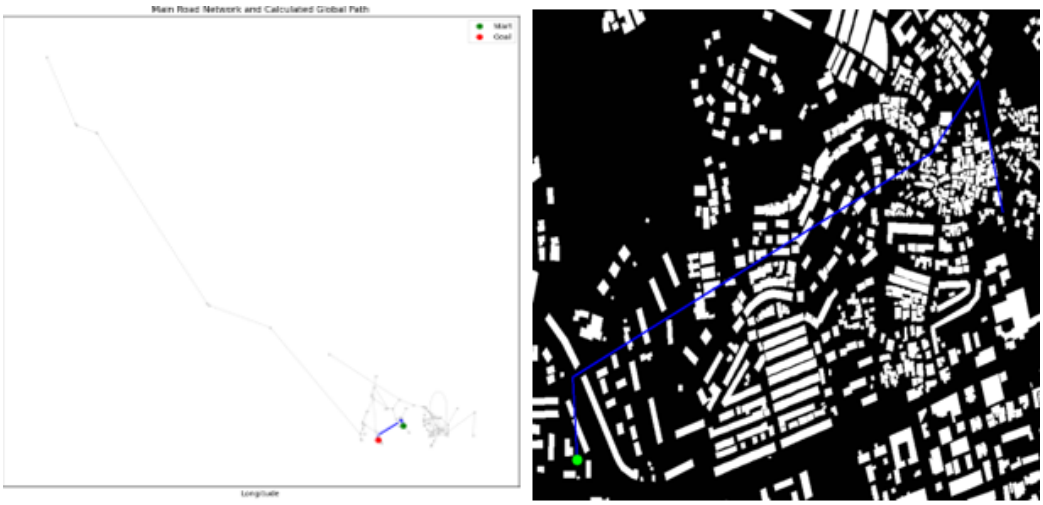


Figure 4.5: YOLO + Depth Map result

The green line represents in figure 4.6 the robot's planned and dynamic path, visually embodying how the robot responds to its perceived environment. The key observation about this green line is that it does not collide with or intersect any of the detected obstacles whether they are the people or cars highlighted by the yellow YOLO bounding boxes. This outcome underscores the effectiveness of the Artificial Potential Field (APF) algorithm in generating collision-free trajectories. By defining each detected object in the scene as a "repulsive point" with a force increasing with proximity, the robot can "feel" the presence of these obstacles before reaching them, proactively causing its path to deviate and safely circumnavigate them. The green line smoothly curving around the person in the foreground is a clear illustration of this behavior, where the robot alters its direction to pass safely by, ensuring maximum safety in navigating a complex sidewalk environment As shown in the (figure 4-6), when the robot approaches an obstacle (the woman), it changes its direction toward a valid path.

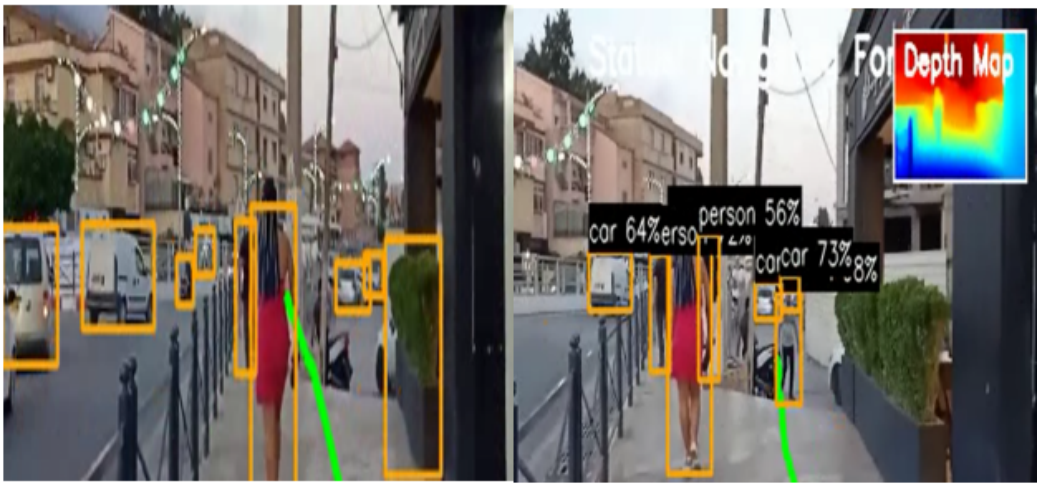


Figure 4.6: Result of the path in real life

4.5 Evaluation

Metrics we use and apply as we see in figure 4.6 :

• Precision and Recall: These are metrics for the accuracy of the labels being provided (car, person, etc.).The precision is high-that is, when it predicts "car," it is mostly correct, which means that there are very few false positives. In the images, the precision seems really high; all the objects labeled are correctly identified. High Recall means that it finds all the actual cars and people in the scene, i.e., low false negatives. On this first image, the model appears to have high recall, finding multiple cars and the person.

• Evaluation of the Images: Visually, the provided frames have a very high perception accuracy. The labels are appropriate, where the confidence score is strong-for example, car 99%, person 82% and the bounding boxes tightly wrap around the objects. This means that the underlying object detection model that generated these JSON files must be performing very 45 well on images of this specific type of street scene.

• Assessing Pathfinding "Accuracy" (Success, Safety, and Efficiency),In the case of the pathfinding part-the green line, "accuracy" is less about being "correct" and more about being successful, safe, and efficient.

• Success Rate: The most important metric is simply: Did the algorithm find a path from the start to the goal The code handles the case that no path is found if not path, which would be a failure. However, in those specific

instances, as shown by the fact that all images do have a green line, success rate would be 100

• Safety: This is the most important measure of the path's quality. A "safe" path is one that maintains a sufficient distance from all obstacles on the cost map. The algorithm attempts to ensure this by "inflating" the obstacles in the create cost map function, creating a buffer zone.

• Evaluation of the Images: This path is an excellent example of a safe path. In every frame, the green line keeps firmly in the sidewalk area and well away from the bounding boxes for both cars and people, showing the main success of the algorithm.

• Efficiency: A path is efficient when it is reasonably direct and not unnecessarily long or jerky. The algorithm is designed to find the shortest path based on the "costs" in the grid, while the smoothing function, find smooth path, aids in removing unnatural, sharp turns.

4.6 Interface

The interface of a grocery web application serves as the first point of interaction between users and the system. It is designed in a way to provide a smooth and intuitive shopping experience, allowing users to browse products, manage their shopping cart, and efficiently complete a purchase. A good interface enhances usability, accessibility, and user satisfaction and integrates features like search, filtering, and personalized recommendations to make the navigation process easier and assist in decision making.

This image 4.7 represents the registration interface, allowing users to create an account to access their profile, manage their shopping cart, and track orders, enabling them to browse and purchase products conveniently.

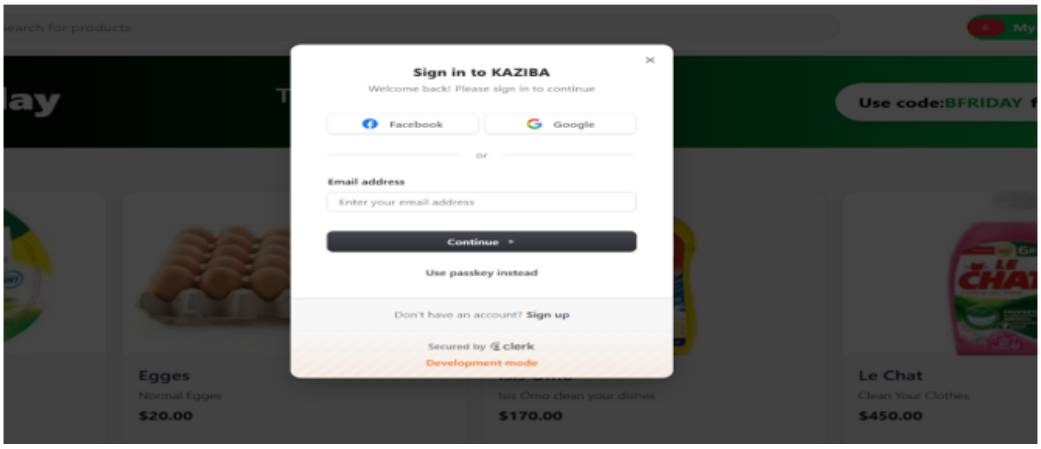


Figure 4.7: Sign up and Sign In interface

This is the homepage of a grocery shop's web application 4.8, "Kaziba Shop." The top section features the store's logo, a prominent search bar, and user-centric navigation including "My Basket" and "Sign In." A large green banner dominates the upper half of the page, likely used for promotions or important announcements. Below this banner, a "Filter by Category" option allows users to refine their product search. The main display area presents a grid of four featured products: "Cheezy" fromage, "Egges," "Isis Omo" dish soap, and "Le Chat" laundry detergent, each clearly showing its name, a short description, and price. A small "1 issue" notification is also present at the bottom of the page.

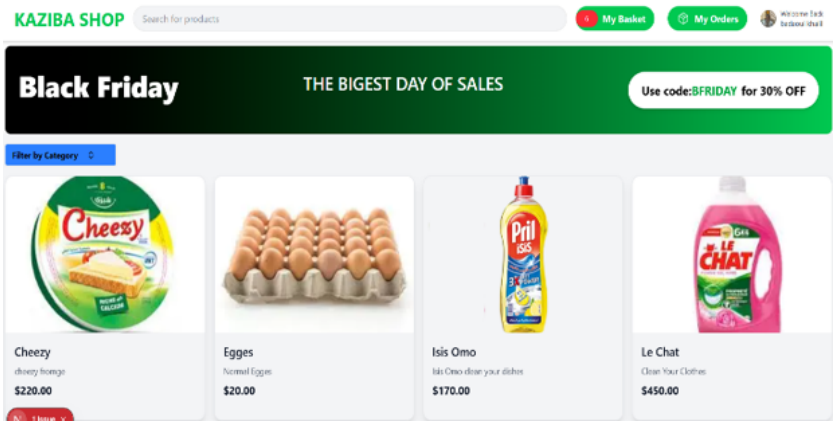


Figure 4.8: home page interface

This interface allows the user to browse grocery items by selecting a specific category4.9. The category filter helps organize all products such as drinks, dairy, oils, sweets, and other everyday grocery essentials, making it easy for the user to quickly find what they are looking for. Once a category is chosen, the related products appear with their image, name, and price, helping the user clearly view the items before deciding to add them to their basket.

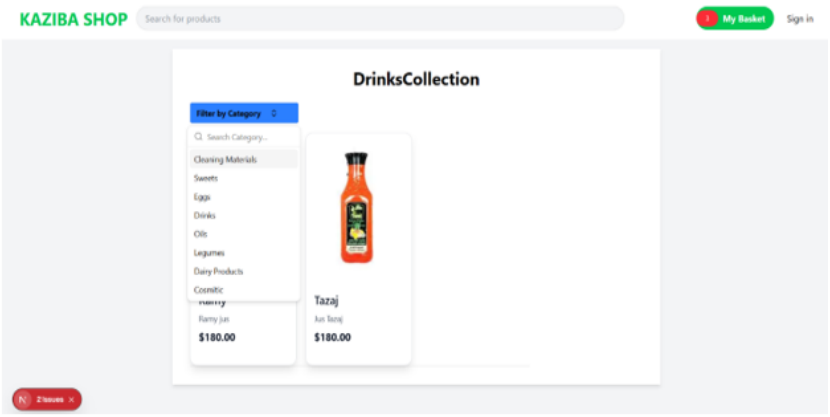


Figure 4.9: Interface category

This page serves as the final review step before completing a purchase 4.10. It displays the selected product, its unit price, and allows the user to adjust the quantity directly from the basket. The order summary on the right dynamically updates the total cost based on the chosen quantity, helping the user clearly verify the total amount they will pay. Once everything is confirmed, the user can proceed by clicking the Checkout button to finalize the sale.

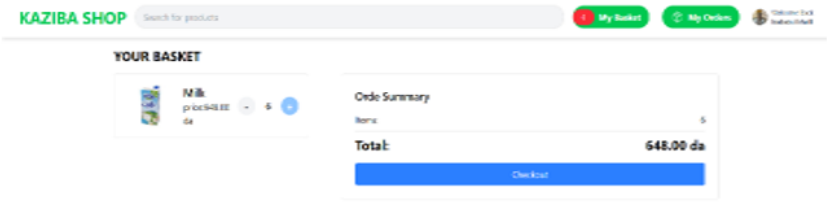


Figure 4.10: Searching Interface

This page is the secure payment interface where the user confirms their purchase and completes the transaction 4.11. It clearly shows the selected item, its quantity, and the total amount due. The user can enter their email, choose a payment method, and fill in their card information to finalize the order. There is also an option to add a promo code and to save payment details for faster checkout in the future. Once all the information is completed, the user simply clicks Pay to confirm and finish the purchase.

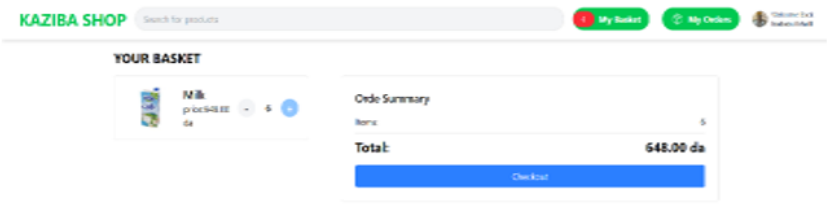


Figure 4.11: Payment interface

This interface illustrates the robot order confirmation page of the KAZIBA SHOP grocery website 4.12, where the system verifies and confirms the delivery ride fee calculated by a delivery robot. The robot's camera feed displays the detected grocery items (such as milk packs), ensuring that the products are correctly identified before confirming the delivery cost. On the right side, the Order Summary shows the total number of items, their price, and the ride fee estimated by the robot, providing users with full transparency before payment. With clear options to "Confirm and Pay" or

"Cancel Order," this page ensures that the customer can review and approve the robot-calculated delivery price before completing the purchase, creating a reliable and smart checkout experience.

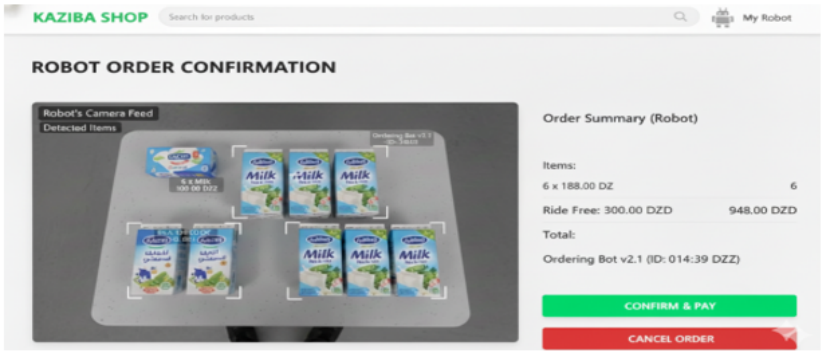


Figure 4.12: Ordering Buy Robot interface

This store interface is accessible only to the grocery manager 4.13. It allows the manager to present products by category, provide descriptions and prices, and manage an orders section to review daily orders and add new ones as needed.

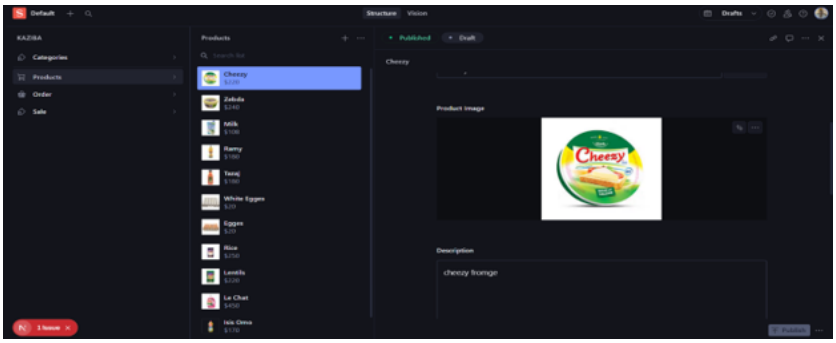


Figure 4.13: Store interface

4.7 Conclusion

In this chapter, we present the results of the simulation for the proposed approach. The findings indicate that this method is the most suitable for our project. Once the camera starts functioning, the algorithm processes at 400

frames per second, providing the robot with more time for wheel actuation and enabling instant movement, which results in significant time efficiency. The combination of the robot's performance in obstacle avoidance and path planning, along with the user interfaces, brings the project very close to a real-world implementation and positions it as one of the optimal solutions for last-mile delivery robots.

Chapter 5

Conclusion

5.1 Introduction

In recent years, autonomous sidewalk delivery robots have become an innovative solution for last-mile delivery, providing a faster, safer, and more efficient alternative to traditional human-based services. Using sensors, cameras, GPS, and intelligent algorithms, these robots navigate urban environments to deliver goods directly to customers. Major companies like Starship Technologies and Amazon have already deployed such systems, showcasing their potential to transform local commerce. Driven by the need for sustainable and contactless delivery, autonomous robots address challenges of traffic, pollution, and high operational costs associated with conventional methods. This project aims to design a complete autonomous grocery delivery ecosystem that integrates a web-based ordering platform with an intelligent robot. The robot uses a hybrid navigation system combining the A* algorithm for global path planning and the Artificial Potential Field (APF) method for local obstacle avoidance. This chapter covers the hardware, software algorithms, and web technologies necessary to build a reliable, connected, and intelligent delivery system, laying the foundation for future smart and contactless logistics solutions.

5.2 Limits

Sidewalk delivery robots face several challenges that affect their performance and real-world deployment. Their navigation heavily depends on sensor quality and environmental conditions, with poor lighting, weather, or dense urban areas reducing reliability. Infrastructure constraints, such as uneven sidewalks, stairs, and unpredictable pedestrian behavior, further challenge mobility. Testing in simulations cannot fully capture real-world dynamics, leading to potential discrepancies. Battery life limits range and continuous operation, while obstacle avoidance algorithms like APF can get trapped in local minima or react unpredictably in crowded settings. Reliable network connectivity is essential for communication and tracking, and current hardware may struggle under adverse conditions. Additionally, social acceptance, legal regulations, and limited multi-robot coordination pose barriers to large-scale deployment. Addressing these issues is crucial for safe, efficient, and widespread adoption.

5.3 Perspectives

The development of autonomous sidewalk delivery robots holds significant potential for technical and societal advancements. Future improvements in AI, computer vision, and deep learning will enhance perception, object detection, and adaptive navigation in complex urban environments. Advances in hardware, multi-sensor fusion, and high-precision localization will increase autonomy, reliability, and energy efficiency. Web platforms will evolve to offer real-time route optimization, fleet coordination, and enhanced user interaction, improving both operational efficiency and customer experience. Large-scale deployment will also require regulatory adaptation, social acceptance, and infrastructure support, such as smoother sidewalks and charging stations. Ultimately, the integration of AI, sustainable hardware, and cloud-based management could enable autonomous delivery robots to become a key component of smart, efficient, and environmentally friendly urban logistics systems.

5.4 Final Conclusion

This work investigated the conceptual and technological underpinnings required to create an autonomous sidewalk delivery system that would allow for seamless integration of robotic navigation with a web-based delivery platform. In this study, it focused on designing a complete ecosystem where software intelligence and mechanical autonomy cooperated in bringing about a contactless, safe, efficient delivery experience. We reviewed the autonomous navigation systems in depth, starting with the basic hardware composition that allows the robot to perceive and interact with its surroundings, including sensors, processors, and mobility mechanisms. In a similar vein, it has also become important to discuss pathfinding and obstacle avoidance algorithms, in which the hybrid implementation of A* and APF algorithms have been very effective. Similarly, this integration allows the robot to make globally optimal path plans while dynamically responding to local obstacles in real time, achieving a balance between precision, adaptability, and computational efficiency. Meanwhile, in web technologies, the research underlined that the digital platform acts as a critical link between the front end, which deals with users, and the actual robot that physically delivers the package. While the platform oversees everything from product selection, confirmation

of orders, authentication, and tracking through to final delivery, it assures smooth translation of online customer requests into real-world robotic actions, thereby forming a cohesive intelligent delivery ecosystem. The system has considerable potential, but quite a number of shortcomings include environmental ones like uneven sidewalks and varying weather conditions; sensor reliability; dependence on stable internet connectivity; and incomplete legal and regulatory frameworks regarding autonomous delivery. Yet, ongoing developments in the areas of artificial intelligence, embedded computing, sensor fusion, and smart city infrastructure are expected to lessen these challenges. Looking ahead, continued advancements in deep learning, real-time mapping, and multi-robot coordination will continue enhancing the autonomy, efficiency, and safety of such systems. As cities evolve toward smarter and more sustainable infrastructures, sidewalk delivery robots can become an integral part of urban logistics: reducing emissions, minimizing congestion, and improving delivery efficiency. The present research lays a solid foundation for the future development and deployment of autonomous sidewalk delivery robots. By effectively integrating intelligent software, adaptive hardware, and web-based communication, it shows that robust, scalable, and sustainable solutions for last-mile delivery are achievable. The contribution of this research extends beyond being an advanced autonomous system to a meaningful step toward a future when intelligent robotics forms a natural part of daily urban life.

Bibliography

[1] https://www.howtogeek.com/(15/02/2025)

[2] Weinberg, D., Dwyer, H., Fox, S. E., & Martelaro, N. (2023). Sharing the sidewalk: Observing delivery robot interactions with pedestrians during a pilot in Pittsburgh, PA.

[3] https://www.boston.gov/(15/02/2025)

[4] Shaklab, E., Karapetyan, A., Sharma, A., Mebrahtu, M., Basri, M., Nagy, M., Khonji, M., & Dias, J. (2023). Towards autonomous and safe last-mile deliveries with AI-augmented self-driving delivery robots. arXiv preprint.

[5] https://spectrum.ieee.org/(16/04/2025)

[6] Yang, X.-S. (2021). Nature-Inspired Optimization Algorithms (2nd ed.). Academic Press, Elsevier.

[7] Russell, S. J., & Norvig, P. (2020). Artificial Intelligence: A Modern Approach (4th ed.). Pearson.

[8] Abdi, M. I. I., Khan, M. U., Güneş, A., & Mishra, D. (2020). Escaping local minima in path planning using a robust bacterial foraging algorithm. Applied Sciences, 10(21), 7905.

[9] Rhifky, M. (2021). Comparative analysis of Greedy, A-Star, and Dijkstra algorithms in path planning. Journal of Robotics and Autonomous Systems, 134, 103676.

[10] Alkhlidi, M., Ahmed, S., & Li, Y. (2021). Multi-robot path planning using Cuckoo Search and Modified Particle Swarm Optimization. Applied Soft Computing, 111, 107677.

[11] Chen, J., Zhang, H., & Wang, L. (2021). Intelligent bionic path planning algorithms: A review. Journal of Intelligent Robotic Systems, 102(2), 45.

[12] Lui, T., & Li, Z. (2022). Hybrid mobile robot path planning with modified Gray Wolf Optimization and situation assessment. Robotics and Autonomous Systems, 150, 103955.

[13] Martins, O.O., Adekunle, A.A., Olaniyan, O.M., Bolaji, B.O. (2022). An Improved multi-objective A-star algorithm for path planning in a large workspace: Design, Implementation, and Evaluation. Scientific African.

[14] Guo, L., Chen, Y., & Wang, S. (2021). LSTM_FTR: Fusion-based local path planning for mobile robots. IEEE Access, 9, 98765–98780.

[15] Szczepanski, P., Nowak, K., & Kowalski, M. (2022). Enhanced local path planning for AGVs using modified Artificial Potential Field. International Journal of Advanced Robotic Systems, 19(4), 172988142210894.

[16] Hongwie, L., Chen, Q., & Zhao, R. (2021). Advances in hybrid and reinforcement learning-based path planning for multi-robot systems. Robotics and Autonomous Systems, 143, 103739.

[17] Liu, L., Wang, X., & Zhang, Y. (2021). Trends in environment modeling for robot path planning: A review of 105 studies. Journal of Field Robotics, 38(7), 1020–1042.

[18] Abdi, M. I. I., Khan, M. U., Güneş, A., Mishra, D. (2020). Escaping local minima in path planning using a robust bacterial foraging algorithm. Applied Sciences, 10(21), 7896.

[19] Alkhlidi, M. Y. H., & El-Sherbiny, A. (2021). Optimal path planning for mobile robot based on hybrid cuckoo search and particle swarm optimization. Journal of Physics: Conference Series, 1962(1), 012003.

[20] Chai, H., Li, S., & Song, J. (2021). Mobile robot path planning based on improved particle swarm optimization. Journal of Physics: Conference Series, 1883(1), 012015.

[21] Chen, Y., Bai, G., Zhan, Y., Zhang, X., & Liu, H. (2021). Path planning and obstacle avoidance of the USV based on improved ACO-APF hybrid algorithm with adaptive early-warning. IEEE Access, 9, 40728–40742.

[22] Guo, N., Li, C. H., Gao, T. T., & Zhang, G. L. (2021). A fusion method of local path planning for mobile robots based on LSTM neural network and reinforcement learning. Mathematical Problems in Engineering, 2021, 1–21.

[23] Liu, Y., & Li, X. (2022). A hybrid mobile robot path planning scheme based on modified gray wolf optimization and situation assessment. Computational Intelligence and Neuroscience, 2022, 8610580.

[24] Martins, L. D. C., dos Santos, V. G., & de Oliveira, H. A. (2022). Mobile robot path planning using an improved mayfly optimization algorithm. Artificial Intelligence Review, 55(6), 4647–4671.

[25] Rhifky, M., Faris, M., & Hidayat, N. (2021). Analysis of shortest path search using A Star, Dijkstra, and Greedy Best First Search algorithm. Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi), 5(2), 273–281.

[26] Szczepański, R., Tarczewski, T., & Grzesiak, L. M. (2022). Energy efficient local path planning algorithm based on predictive artificial potential field. IEEE Access, 10, 39729–39742.

[27] Annarita De Maio, G., Ghiani, G., Laganà, D., & Manni, E. (2024). Sustainable last-mile distribution with autonomous delivery robots and public transportation.

[28] Abdi, M. I. I., Khan, M. U., Güneş, A., & Mishra, D. (2020). Escaping local minima in path planning using a robust bacterial foraging algorithm. Applied Sciences, 10(21), 7896.

[29] LaValle, S. M. (2006). Planning Algorithms. Cambridge University Press.

[30] Ömer Mutlu Türk KAYA.(2022).DESIGN AND IMPLEMENTATION OF EIGHT-WHEELED SEMI-AUTONOMOUS DELIVERY ROBOT .

[31] https://github.com/khalil97766/PFE.(20/10/2025)