Dynamic AMR Navigation: Simulation with Trajectory Prediction of Moving Obstacles

Tomás Cadete
Electrical Computers Eng. Dept.
Fac. de Engenharia, Univ. do Porto
Rua Dr. Roberto Frias
4200-465, Porto, Portugal
up201904832@edu.fe.up.pt

Vítor H. Pinto
Mechanical Eng. Dept., SYSTEC (DIGI2) & ARISE
Fac. de Engenharia, Univ. do Porto
Rua Dr. Roberto Frias
4200-465, Porto, Portugal
vitorpinto@fe.up.pt

José Lima
CeDRI
Instituto Politécnico de Bragança
5300-253, Bragança, Portugal
INESC TEC, Porto, Portugal
jllima@ipb.pt

Gil Gonçalves
SYSTEC, ARISE & ECE Dept.
Fac. de Engenharia, Univ. do Porto
Rua Dr. Roberto Frias
4200-465, Porto, Portugal
gil@fe.up.pt

Paulo Costa
ECE Dept. & INESC TEC
Fac. de Engenharia, Univ. do Porto
Rua Dr. Roberto Frias
4200-465, Porto, Portugal
paco@fe.up.pt

2024 7th Iberian Robotics Conference (ROBOT) | 979-8-3503-7636-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/ROBOT61475.2024.10797420

***Abstract*—Autonomous Mobile Robots (AMRs) have significantly transformed task management in factories, warehouses, and urban environments. These robots enhance operational efficiency, reduce labor costs, and automate various tasks. However, navigating dynamic environments with moving obstacles, such as human workers, vehicles, and machinery, remains challenging. Traditional navigation systems, which rely on static maps and predefined routes, struggle to adapt to these dynamic settings. This research addresses these limitations by developing a dynamic navigation system that improves AMR performance in industrial and urban scenarios. The system enhances the A* algorithm to account for the current positions and predicted trajectories of moving obstacles, allowing the AMR to navigate safely and efficiently. Advanced sensor technologies, such as LiDAR and stereo cameras, are utilized for real-time environmental perception. The system integrates trajectory prediction and an Artificial Potential Field (APF) method for emergency collision avoidance. The solution is implemented using the Gazebo simulator and the Robot Operating System (ROS2), ensuring real-time operation and adaptive path planning. This research aims to significantly improve AMR safety, efficiency, and adaptability in dynamic environments.**

***Index Terms*—Autonomous Mobile Robot, Dynamic Navigation, Trajectory Prediction, Moving Obstacles, Simulation, Artificial Potential Field**

I. Introduction

The use of Autonomous Mobile Robots (AMRs) in industrial settings has significantly transformed the management of tasks within factories and warehouses. These autonomous systems have become highly sought after as industries aim to improve productivity, efficiency, and safety. AMRs offer a practical solution by increasing operational efficiency, reducing labor costs, and automating various tasks. However, AMR navigation faces challenges in dynamic environments. These environments often feature moving obstacles such as human workers, other vehicles, and machinery, which can disrupt the intended paths of AMRs. Traditional navigation systems, reliant on static maps and predefined routes, struggle to adapt to such changes, resulting in inefficiencies and potential safety risks. Advanced navigation systems are needed to enable AMRs to operate safely and efficiently in all sorts of environments, allowing the robots to adjust their paths dynamically in real time. This involves detecting obstacles, predicting their movements, and planning collision-free trajectories. Integrating sophisticated sensor technologies, machine learning algorithms, and robust path-planning strategies is essential to achieve these capabilities.

The advancement of AMRs and AGVs heavily depends on advanced technologies for dynamic navigation and obstacle avoidance, particularly in environments with high variability and moving obstacles. The following section thoroughly reviews the current state of the art in these areas, outlining the sensor technologies, object detection methods, trajectory prediction algorithms, and path planning and collision avoidance strategies that are fundamental to modern autonomous navigation systems. By examining these essential components, our goal is to contextualize the progress and methodologies that shape the design and implementation of our system.

II. Literature Review

Object detection plays a crucial role in the functionality of AMRs by enabling the identification and classification of obstacles within the environment. YOLO (You Only Look Once) [1] is a state-of-the-art deep learning model known for its exceptional speed and accuracy in object detection, making it particularly effective for robots operating in dynamic environments. Pooloo et al. [2] designed an AMR using image processing and deep learning for warehouse surveillance and disinfection. They employed YOLOv4 for object detection,

which was instrumental in identifying and navigating the warehouse's obstacles, such as people and machinery. Their approach highlighted YOLO's adaptability to real-world applications, ensuring safe navigation and aiding in operational tasks like disinfection. Sanchez-Cubillo et al. [3] focused on the application of YOLO in the automated inspection of critical assets. Their research benchmarked several generations of the YOLO architecture, demonstrating its effectiveness in tasks that support the perception capabilities of AMRs. Their field tests in real-world scenarios, including railway track inspection and maritime container loading, underscored YOLO's robustness in diverse and unstructured environments, paving the way for fully automated inspection and maintenance tasks.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a clustering algorithm widely used in object detection, especially in processing sensor data such as LiDAR point clouds. This technique is particularly advantageous for AMRs operating in dynamic environments, as it groups closely packed points and identifies outliers as noise, enabling the effective transformation of LiDAR data into meaningful object representations. Yabroudi et al. [4] proposed an adaptive DBSCAN method specifically designed for LiDAR point cloud clustering in autonomous driving applications. Their approach addresses the manual tuning of DBSCAN parameters (epsilon and minPts) by automating the estimation process based on the field of view division scheme and point densities within these divisions. This adaptation allows the parameters to be dynamically adjusted for each point cloud frame, improving the clustering accuracy for objects at varying distances from the LiDAR sensor. By evaluating their method on the KITTI dataset, the authors demonstrated that the adaptive DBSCAN can achieve comparable results to the traditional DBSCAN without the need for manual parameter selection, thus enhancing the robustness and efficiency of object detection in dynamic environments.

Path planning and collision avoidance play a critical role in AMR autonomous navigation. These processes enable AMRs to efficiently calculate their routes and steer clear of obstacles, both stationary and in motion, ensuring safe and effective travel. Given the complexity of these tasks, advanced algorithms are necessary to adapt to changing environments and avoid potential hazards. Ju et al. [5] combined the A* algorithm with a jump point search technique, significantly increasing the path-finding speed and generating shorter paths under certain conditions. Their improved A* algorithm optimizes the cost function, considering the shortest line segment between two points, demonstrating its suitability for AGV applications where path length is critical. Erke et al. [6] developed an optimized A* algorithm that integrates an additional heuristic function to minimize energy consumption and enhance motion control. By incorporating an artificial potential field into the heuristic function, the algorithm ensures that turning points are positioned safely away from obstacles, leading to more efficient and collision-free paths in dynamic environments.

Moreover, trajectory prediction has also a crucial role in the navigation systems of AMRs as it allows them to anticipate the movements of dynamic obstacles and plan safe, efficient paths. Methods such as the Particle Filter, the Kalman Filter and Extended Kalman Filter, and Neural Network approaches provide the mathematical frameworks needed to predict the future positions of objects based on their current and past states, which is essential for dynamic obstacle avoidance and collision prevention. Lin et al. [7] propose an enhanced Kalman Filter framework that integrates multiple sensor inputs, including LiDAR and cameras, to improve the accuracy of trajectory prediction. Their approach leverages the filter's strength in handling linear systems with Gaussian noise, making it highly effective in scenarios where the movement of objects can be approximated by linear motion models. Similarly, Guo et al. [8] present an EKF-based method for predicting the trajectories of dynamic obstacles in urban environments. Their approach incorporates advanced sensor fusion techniques to account for the nonlinear relationships between state variables, significantly enhancing prediction accuracy in complex scenarios.

III. System Implementation

The proposed system has been specifically designed to navigate dynamic environments by employing advanced perception, prediction, planning, and control techniques. Its primary objective is to enable the AMR to efficiently and safely navigate through environments containing moving obstacles, such as dynamic industrial settings with human workers and vehicles. The system is built on the Robot Operating System 2 (ROS2) framework, allowing real-time communication between its various components. It also integrates state-of-the-art sensor technologies like LiDAR and stereo cameras and advanced algorithms for object detection, trajectory prediction, path planning, and control. Every system component is seamlessly integrated to provide a robust solution for dynamic navigation.

In addition to the core navigation system, a realistic simulation environment was created within the Gazebo simulator to validate the AMR's performance. This simulation environment includes a differential wheeled robot equipped with essential sensors, an asphalt road with circulating cars, traffic lights controlled by a custom Gazebo plugin, moving obstacles such as pedestrians and cyclists, and various static obstacles. This environment serves as a controlled platform for testing the AMR's capabilities in various urban scenarios, ensuring the system's reliability and performance before real-world deployment.

Robot Operating System 2 (ROS2) is an open-source framework that provides a robust set of tools and libraries essential for building and managing robot applications. It is the successor to the original ROS framework, designed to address the limitations of ROS1 by incorporating real-time capabilities, improved performance, enhanced security, and better support for multi-robot systems. ROS2 has become the preferred choice for many robotics projects due to its flexibility, scalability, and the extensive ecosystem it supports.

The architectural design of the proposed robotic navigation system integrates multiple components, each serving a specific role, to enable seamless operation in dynamic environments. This design leverages the ROS2 framework to facilitate communication and coordination among various nodes responsible for object detection, occupancy grid generation, trajectory prediction, planning, and control. The robust interaction between these nodes and the robot's sensors and actuators ensures real-time data processing and decision-making.

The robotic navigation system is designed to support robust and efficient autonomous navigation in dynamic environments. It leverages the modular capabilities of the ROS2 framework to integrate various functional components into a cohesive system. The system comprises six primary nodes: Perception, Occupancy Grid Generator, Trajectory Predictor, Traffic Light Detection, Planner, and Control. Each node performs a specific role and communicates with other nodes to share data and coordinate actions. This architecture ensures that each component can be developed, tested, and maintained independently while providing a structured flow of information and control commands throughout the system. The interaction between nodes is primarily handled through ROS2's publish-subscribe messaging system, ensuring data is transmitted efficiently and reliably.

The data flow within the robotic navigation system is a critical aspect of its architecture, enabling the seamless transfer of information between various nodes and ensuring real-time processing and decision-making. This system employs a structured data flow that integrates sensor inputs, computational processing, and actuator commands to navigate dynamic environments effectively.

Data flow begins with the collection of sensor data. The stereo camera captures RGB and depth images, which are then published to specific ROS2 topics. Simultaneously, the LiDAR sensor provides high-resolution distance measurements, publishing this data on another ROS2 topic. The robot's odometry data, which includes information about its position and orientation, is also continuously published to a ROS2 topic.

The Perception Node subscribes to the RGB and depth image topics and the odometry data. It processes the RGB images using the YOLOv8 model to detect and classify objects. The depth information is used in conjunction with odometry data to calculate these objects' world coordinates and sizes. The Perception Node then publishes the detected objects' types, sizes, and coordinates to a topic.

The Occupancy Grid Generator Node subscribes to the object data from the Perception Node and the raw LiDAR data and odometry information. It fuses these data sources to create and update an occupancy grid, providing a comprehensive view of the environment. The updated occupancy grid is then published to a topic for use by other nodes.

The Predictor Node subscribes to the occupancy grid data. It tracks the positions of objects over time, calculating their velocities and accelerations based on the history of positions and time differences. Using this information, the Predictor Node predicts the future trajectories of moving objects and publishes these predicted trajectories to a topic.

The Traffic Light Detection Node subscribes to the RGB images from the stereo camera and odometry data. It uses the YOLOv8 model to detect the state of traffic lights in the images and publishes the detected state, such as red or green, to a topic.

The Planner Node subscribes to the occupancy grid from the Occupancy Grid Generator Node, the predicted trajectories from the Predictor Node, and the traffic light state from the Traffic Light Detection Node. Using the enhanced A* algorithm, it generates a path for the robot, considering both static and dynamic obstacles. Additionally, it employs the Artificial Potential Field (APF) method to handle imminent collision situations. The Planner Node then publishes the planned path to a topic.

The Control Node subscribes to the planned path from the Planner Node and the odometry data from the robot. It continuously adjusts the robot's movements to follow the planned path, generating velocity commands based on the comparison between the robot's current position and the next waypoint on the path. In situations requiring rapid evasion, it increases the robot's linear velocity. These velocity commands are published to a topic that the robot's wheels' controllers subscribe to, ensuring that the planned movements are executed accurately.

The overall data flow is managed through ROS2's publish-subscribe messaging system, allowing for asynchronous communication between nodes and ensuring that data is processed in real time. This structured flow of data, from sensor inputs to actuator commands, enables the robot to perceive its environment, predict the movements of dynamic obstacles, plan safe paths, and execute these paths effectively.

Figure 1 provides a visual representation of the data flow, illustrating the sequentiality of interactions between the nodes and the robot's sensors and wheels.

The stereo camera-based object detection system plays a vital role in this robotic system, allowing it to perceive and comprehend its dynamic surroundings. By utilizing advanced deep learning techniques for object classification and integrating them with depth information, the system can precisely identify and locate objects in 3D space. This section provides a detailed overview of the detection process, covering image acquisition, YOLOv8 object classification, depth calculation, and world coordinates calculation. The choice of the YOLOv8s architecture for object classification is driven by its balance of accuracy and computational efficiency. Calculating world coordinates from the bounding box coordinates and depth information is critical in enabling the robot to perceive its environment accurately.

Transforming the 3D coordinates from the camera to the world frame accounts for the robot's position and orientation. The pose information is obtained from the odometry data and is represented as a translation vector (t_x, t_y, t_z) and a rotation matrix derived from the orientation quaternion (qx, qy, qz, qw).

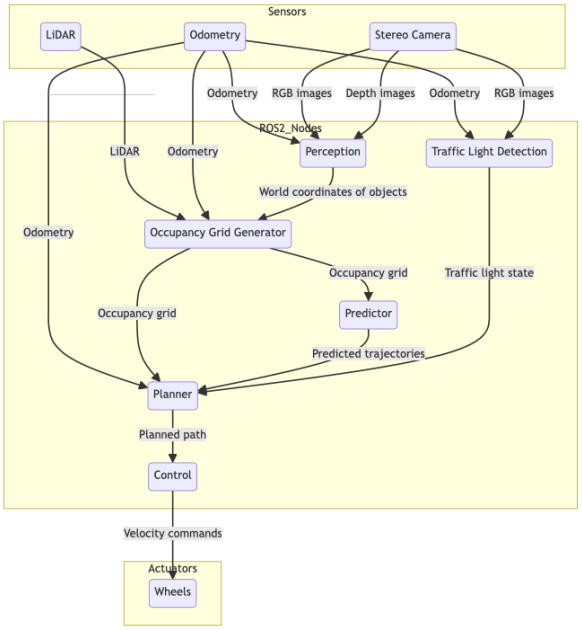

Fig. 1: Data flow of the system.

The transformation from the camera frame to the world frame involves two steps:

- **Rotation**: Apply the rotation matrix to the camera coordinates to align them with the world frame.
- **Translation**: Add the translation vector to the rotated coordinates to obtain the final world coordinates.

Mathematically, the transformation can be expressed as:

$$\begin{bmatrix} x_{\text{world}} \\ y_{\text{world}} \\ z_{\text{world}} \end{bmatrix} = R \begin{bmatrix} x_{\text{camera}} \\ y_{\text{camera}} \\ z_{\text{camera}} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

where R is the rotation matrix derived from the quaternion.

Creating an accurate occupancy grid is a crucial element of our robotic system, empowering it to navigate dynamic environments with precision and safety. LiDAR (Light Detection and Ranging) is a remote sensing technology that measures distances by illuminating the target with laser light and analyzing the reflected pulses. This technology is integral to our robotic system, providing high-resolution, three-dimensional information about the surrounding environment. Transforming LiDAR detections from the sensor frame to the world frame is essential for accurately updating the occupancy grid with the correct positions of detected obstacles. This transformation involves converting the LiDAR measurements, which are initially relative to the sensor's position and orientation, into global coordinates relative to a fixed world frame. It is essential to update the occupancy grid to ensure the robot maintains an accurate and current representation of its environment. This involves integrating data from the LiDAR sensor and the stereo camera-detected objects to update the grid with the positions of obstacles. The Occupancy Grid Generator node is tasked with this responsibility to enable the robot to navigate safely and efficiently. Accurately predicting the trajectory of obstacles is a very important aspect of autonomous navigation systems, as it allows robots to anticipate and respond to the movements of dynamic objects in their surroundings. The goal is to ensure the robot can navigate safely and effectively in dynamic environments, such as industrial settings or crowded public spaces.

The path planning strategy implemented in our AMR autonomous driving system combines the strengths of the A* algorithm and the Artificial Potential Fields (APF) method. This dual-strategy approach ensures optimal pathfinding for standard navigation and robust collision avoidance in dynamic environments. Our approach extends the traditional A* algorithm by incorporating the predicted trajectories of dynamic obstacles into the path-planning process. This approach combines the discrete search of A* with continuous optimization to generate feasible and smooth paths that consider the movements of other objects in the environment. The A* algorithm uses a cost function $f(n) = g(n) + h(n)$, where:

- $g(n)$ is the cost from the start node to the current node n.
- $h(n)$ is the heuristic estimate of the cost from node n to the goal.

By integrating the predicted trajectories of dynamic obstacles, the enhanced A* algorithm ensures that the generated path is clear of static obstacles and avoids the predicted future positions of moving objects.

The APF method generates a navigation path by simulating attractive forces toward the goal and repulsive forces away from obstacles. The attractive force F_{att} pulls the robot towards the goal, defined as:

$$F_{\text{att}} = \eta \cdot (x - x_{\text{goal}})$$

where η is a positive scaling factor, x is the current position, and x_{goal} is the goal position. The repulsive force F_{rep} pushes the robot away from obstacles, defined as:

$$F_{\text{rep}} = \begin{cases} \zeta \left(\frac{1}{d(x)} - \frac{1}{d_0} \right) \frac{1}{d(x)^2} \cdot \frac{x - x_{\text{obs}}}{d(x)} & \text{if } d(x) \leq d_0 \\ 0 & \text{if } d(x) > d_0 \end{cases}$$

where ζ is a positive scaling factor, $d(x)$ is the distance to the obstacle, d_0 is the distance threshold, and x_{obs} is the obstacle position.

Our strategy leverages the strengths of both A* and APF to navigate dynamic environments efficiently and safely with a hybrid approach. The enhanced A* algorithm is used for standard path planning, ensuring the robot follows an optimal and smooth path by considering the predicted positions of dynamic obstacles in advance. However, when the robot detects an imminent collision situation—such as being on the path of a moving obstacle—the APF method quickly computes a new path that guides the robot away from danger.

The process involves the following steps:

1) **Path Planning with A***: The Planner node continuously runs the enhanced A* algorithm to determine the optimal

path from the robot's current position to the goal, considering the predicted positions of dynamic obstacles.

2) **Collision Detection**: The robot monitors its environment using sensors and predicts potential collisions by comparing its planned path with the trajectories of moving obstacles.
3) **Emergency Path Adjustment with APF**: If a collision is imminent, the APF method generates a new path by applying repulsive forces to steer the robot into free space while trying to maintain an overall direction toward the goal.
4) **Path Refinement and Execution**: The Planner node refines the path to ensure smooth transitions and publishes the final path for the Control node to execute.

This hybrid strategy provides a robust solution for path planning in dynamic environments, balancing the need for optimal navigation and reactive collision avoidance.

IV. Simulation Environment

Gazebo is an open-source 3D dynamic simulator designed to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments. It offers a wide range of features, including high-fidelity physics simulation, sensor simulation, and a rich library of robot models and environments.

Gazebo's key features include:

- **Physics Simulation**: Realistic physics simulation with support for multiple physics engines.
- **Sensor Simulation**: Accurate simulation of various sensors, such as cameras, LiDAR, GPS, and IMUs.
- **Robot Models**: A comprehensive library of robot models, including mobile robots, manipulators, and drones.
- **Environments**: A rich set of predefined environments and the ability to create custom environments.
- **Plugins**: Extensible architecture with support for custom plugins to add new features and functionalities.
- **Actors**: Simulation of dynamic entities, such as pedestrians or vehicles, to create more realistic and unpredictable environments.

Gazebo is particularly well-suited for ROS2 simulations due to its tight integration with the ROS ecosystem. It provides a ROS interface that allows for seamless communication between the simulation and ROS nodes, enabling the use of ROS tools and libraries in the simulation environment. This integration facilitates the development and testing of robotic algorithms in a controlled and reproducible manner before deploying them on physical robots.

The Traffic Light Control Plugin is a custom Gazebo plugin developed to manage the state transitions of traffic lights in the simulation environment. Written in C++, the plugin interfaces directly with the Gazebo simulation, allowing for real-time control of traffic light signals. The primary function of this plugin is to control the traffic light's joints, which represent the light colors (red, yellow, and green), and switch them according to predefined timing intervals.

The plugin is structured into two main files: TrafficLight.cpp and TrafficLight.hpp. The header file declares the class and its member functions, while the source file contains the implementation details.

Key components of the plugin include:

- **Initialization**: Setting up the plugin and loading the parameters.
- **Update Function**: A periodic function that updates the state of the traffic light based on the elapsed time.
- **Joint Control**: Mechanisms to control the joints of the traffic light model to represent different colors.

This plugin provides essential functionality for simulating realistic urban environments where traffic signals are crucial in traffic management and can interfere with autonomous robot behavior.

The simulation environment plays a crucial role in developing and validating autonomous navigation systems. It offers a controlled and reproducible platform for testing the performance of robots in various scenarios. The environment comprises several key components, each playing a vital role in creating a realistic and comprehensive testing platform for the autonomous navigation system. The primary components are as follows:

- **Differential Wheeled Robot:** The robot is equipped with a ROS2 interface and various sensors, including a LiDAR sensor, an odometry sensor, and a stereo camera. These sensors provide the necessary data for obstacle detection and environment mapping.
- **Asphalt Road:** A static road model where cars circulate is included.
- **Circulating Cars:** Dynamic actors representing cars are programmed to follow specific trajectories on the road, creating moving obstacles for the robot to interact with. These cars simulate real traffic scenarios and test the robot's ability to handle road crossing situations.
- **Semaphore:** A traffic light model indicates when the robot is allowed to cross the road. This semaphore is controlled by the Traffic Light Control Plugin, which changes the light state (red, yellow, green) at predefined intervals.
- **Moving Objects:** Dynamic obstacles, such as pedestrians and cyclists, are simulated using actor models with predefined trajectories. These models test the robot's ability to detect and avoid moving obstacles in industrial or urban scenarios.
- **Static Objects:** Various static obstacles, such as buildings and trees, are placed within the environment to simulate a realistic urban setting. These objects provide additional richness to the environment.

These components collectively create a comprehensive simulation environment that tests the robustness and reliability of the autonomous navigation system under various conditions and scenarios.

Figure 2 provides a visual representation of the simulation environment within Gazebo. This screenshot highlights the

various components, including the differential wheeled robot, the asphalt road, circulating cars, the semaphore, and other static and dynamic obstacles. The comprehensive setup allows for robust testing of the autonomous navigation system in a realistic simulated environment.

Fig. 2: Gazebo simulation environment.

Through the following video — www.youtube.com/watch?v=WpbglVdaJ9s — it is possible to visualize the operation of the simulation environment and the AMR autonomous navigation system in a complex environment with multiple moving obstacles.

V. Tests and Results

To evaluate the performance of the developed AMR navigation system, we designed multiple test scenarios within the Gazebo simulation environment. Each scenario consists of a world with cars circulating on the road and various moving obstacles off-road, such as pedestrians and cyclists. The robot's objective in each scenario is to:

1) Reach the crosswalk entry while avoiding all static and dynamic obstacles.
2) Safely cross the road when the semaphore is green and all safety conditions are met.
3) From the crosswalk exit, reach the final goal while avoiding all the static and dynamic obstacles once more.

The test scenarios differ in the number of dynamic obstacles on the road (cars) and off the road and the trajectories each of these obstacles follows. The increasing complexity of these scenarios challenges the robustness and adaptability of the navigation system.

Table I provides a comparison of the test scenarios, detailing the number of cars on the road, the number of moving obstacles off the road, and the total number of moving obstacles in each scenario.

TABLE I: Comparison of Test Scenarios

Scenario	Cars on Road	Moving Obstacles off Road	Total Moving Obstacles
1	1	4	5
2	2	6	8
3	3	8	11
4	4	10	14
5	5	12	17

Each scenario was designed to progressively increase the number of dynamic elements, providing a rigorous test of the system's performance in navigating complex and unpredictable environments. The robot must employ real-time decision-making and adaptive path-planning strategies to navigate toward its goal while maintaining safety and efficiency. The increasing difficulty of the scenarios ensures a thorough evaluation of the system's capabilities under varying conditions, thereby highlighting both its strengths and potential areas for improvement.

A set of evaluation metrics was defined to evaluate the developed AMR navigation system's performance comprehensively. These metrics provide insights into the system's efficiency, safety, and robustness under different test scenarios. The following metrics were used:

- **Path Length (m)**: The total distance traveled by the robot from the start to the final goal. This metric helps in assessing the efficiency of the path-planning algorithm. A shorter path length indicates a more efficient navigation strategy.
- **Optimal Path Length (m)**: The shortest possible distance between the start and the final goal, calculated for comparison purposes. This metric serves as a benchmark to evaluate the efficiency of the actual path taken by the robot.
- **Evasion Maneuvers**: The number of evasive actions the robot takes to avoid collisions with dynamic obstacles. A higher number of evasion maneuvers may indicate a more cautious navigation strategy but could also reflect the complexity of the environment.
- **Near Misses**: Instances where the robot came close to a collision but successfully avoided it. This metric provides insights into the system's ability to avoid potential collisions and maintain minimum safety distances.
- **Collisions**: The number of collisions the robot encounters during the navigation task. This is a critical safety metric, with fewer collisions indicating better performance.

These evaluation metrics collectively provide a detailed assessment of the navigation system's performance, highlighting its strengths and identifying areas for improvement. Analyzing these metrics allows us to inspect if the system meets the desired efficiency, safety, and robustness levels in dynamic environments.

The performance of the AMR navigation system was evaluated based on multiple metrics, including path length, total mission time, and the number of evasive maneuvers.

Table II summarizes the key evaluation metrics for each test scenario.

TABLE II: Key evaluation metrics for each test scenario

Scenario	Path Length (m)	Optimal Path Length (m)	Evasion Maneuvers	Near Misses	Collisions
1	127.1	123.8	2	0	0
2	130.9	123.8	4	1	0
3	132.2	123.8	5	0	0
4	134.1	123.8	5	0	0
5	128.7	123.8	7	0	0

To better demonstrate the capabilities of the AMR navigation system, Figure 3 presents a comparison between the actual robot trajectory (blue line) and the optimal trajectory (red line) for test scenario 5.

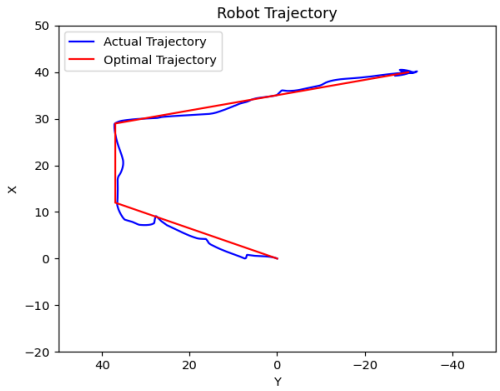

Fig. 3: Comparison of the real trajectory of the robot (blue line) against the optimal trajectory (red line) in test scenario 5.

The test results show that the AMR navigation system performed consistently well across all scenarios. The system was efficient in simpler environments (Scenarios 1 and 2) with minimal evasion maneuvers and shorter mission times. However, as the complexity of the scenarios increased (Scenarios 3, 4, and 5), the system maintained a high level of safety without collisions, although there was an increase in the number of evasion maneuvers. This demonstrates the system's capability to handle challenging environments while ensuring safety.

The results highlight the system's effectiveness in dynamic navigation, particularly in real-time decision-making and path planning. While the system is already efficient and safe, there are opportunities for further optimization to enhance performance in even more complex scenarios. The following section will present a detailed analysis and discussion of these results.

A notable observation is that most near misses were caused by the movement of dynamic objects (e.g., cars and pedestrians) toward the robot rather than the robot moving toward obstacles. This behavior can be attributed to the design of the simulation, where moving objects follow predefined trajectories without considering the robot's position. The AMR's limited linear velocity capabilities (typically around a maximum of 2m/s) also contributed to this, as the robot had to react to faster-moving obstacles in its vicinity. This finding highlights the system's robustness in real-world scenarios, where humans and other robots will likely exercise discernment and take active part in avoiding collisions.

The system's path planning and real-time decision-making capabilities were particularly effective in avoiding collisions and navigating complex environments. This was made possible by using advanced algorithms for object detection, trajectory prediction, and path planning, enabling the AMR to adapt to dynamic changes and ensure safe navigation.

VI. Conclusion and Future Research

The primary objective of this research was to develop an efficient and robust AMR navigation system capable of navigating dynamic environments using advanced sensor technology, object detection algorithms, and real-time path planning. The following specific objectives were identified and pursued, such as the integration of state-of-the-art sensor technologies for accurate environment perception, implementation of advanced object detection and trajectory prediction algorithms, development of a robust path planning algorithm to ensure efficient and collision-free navigation and extensive testing and evaluation of the system in various dynamic scenarios to validate its performance. These achievements demonstrate the successful realization of the project's primary objectives and contribute valuable advancements to the field of autonomous navigation systems.

Some courses for future research are possible, such as advanced trajectory prediction, enhanced evasion strategies and real-world testing and validation. This proposed future work aims to expand the capabilities of AMRs and address the limitations identified in this research. By exploring these areas, future research can continue to advance AMR navigation systems and their practical applications across various industries.

Acknowledgment

This work was financially supported by *PPS 18: 3D navigation system for mobile robotic equipment* from *Agenda GreenAuto: Green Innovation for the Automotive Industry*, no. C644867037-00000013, investment project no. 54, from the Incentive System to Mobilising Agendas for Business Innovation, funded by the Recovery and Resilience Plan and by European Funds NextGeneration EU.

References

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[2] N. Pooloo, W. Aumeer, and R. Khoodeeram, "Design of an amr using image processing and deep learning for monitoring safety aspects in warehouse," in *2022 IST-Africa Conference (IST-Africa)*. IEEE, 2022, pp. 1–10.

[3] J. Sanchez-Cubillo, J. Del Ser, and J. L. Martin, "Toward fully automated inspection of critical assets supported by autonomous mobile robots, vision sensors, and artificial intelligence," *Sensors*, vol. 24, no. 12, p. 3721, 2024.

[4] M. El Yabroudi, K. Awedat, R. C. Chabaan, O. Abudayyeh, and I. Abdel-Qader, "Adaptive dbscan lidar point cloud clustering for autonomous driving applications," in *2022 IEEE International Conference on Electro Information Technology (eIT)*. IEEE, 2022, pp. 221–224.

[5] C. Ju, Q. Luo, and X. Yan, "Path planning using an improved a-star algorithm," in *2020 11th International Conference on Prognostics and System Health Management (PHM-2020 Jinan)*. IEEE, 2020, pp. 23–26.

[6] S. Erke, D. Bin, N. Yiming, Z. Qi, X. Liang, and Z. Dawei, "An improved a-star based path planning algorithm for autonomous land vehicles," *International Journal of Advanced Robotic Systems*, vol. 17, no. 5, p. 1729881420962263, 2020.

[7] C.-Y. Lin, L.-J. Kau, and C.-Y. Chan, "Bimodal extended kalman filter-based pedestrian trajectory prediction," *Sensors*, vol. 22, no. 21, p. 8231, 2022.

[8] B. Guo, N. Guo, and Z. Cen, "Obstacle avoidance with dynamic avoidance risk region for mobile robots in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 5850–5857, 2022.