Robotica (2025), **43**, pp. 449–498
doi:10.1017/S0263574724001954

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

An omnidirectional mecanum wheel automated guided vehicle control using hybrid modified A* algorithm

Ankur Bhargava[1], Mohammad Suhaib[1] and Ajay K. S. Singholi[2]

[1]Department of Mechanical Engineering, Faculty of Engineering and Technology, Jamia Millia Islamia, New Delhi, India
[2]University School of Automation & Robotics, Guru Gobind Singh Indraprastha University, New Delhi, India
Corresponding author: Ankur Bhargava; Email: ankurgsb21@gmail.com

Received: 11 April 2024; **Accepted:** 21 October 2024; **First published online:** 6 December 2024

Keywords: automated guided vehicle; hybrid modified A* (HMA*) algorithm; particle swarm optimization (PSO); omnidirectional mecanum wheel; path planning

Abstract

This paper presents Hybrid Modified A* (HMA*) algorithm which is used to control an omnidirectional mecanum wheel automated guided vehicle (AGV). HMA* employs Modified A* and PSO to determine the best AGV path. The HMA* overcomes the A* technique's drawbacks, including a large number of nodes, imprecise trajectories, long calculation times, and expensive path initialization. Repetitive point removal refines Modified A*'s path to locate more important nodes. Real-time hardware control experiments and extensive simulations using Matlab software prove the HMA* technique works well. To evaluate the practicability and efficiency of HMA* in route planning and control for AGVs, various algorithms are introduced like A*, Probabilistic Roadmap (PRM), Rapidly-exploring Random Tree (RRT), and bidirectional RRT (Bi-RRT). Simulations and real-time testing show that HMA* path planning algorithm reduces AGV running time and path length compared to the other algorithms. The HMA* algorithm shows promising results, providing an enhancement and outperforming A*, PRM, RRT, and Bi-RRT in the average length of the path by 12.08%, 10.26%, 7.82%, and 4.69%, and in average motion time by 21.88%, 14.84%, 12.62%, and 8.23%, respectively. With an average deviation of 4.34% in path length and 3% in motion time between simulation and experiments, HMA* closely approximates real-world conditions. Thus, the proposed HMA* algorithm is ideal for omnidirectional mecanum wheel AGV's static as well as dynamic movements, making it a reliable and efficient alternative for sophisticated AGV control systems.

1. Introduction

Automated guided vehicle (AGV) plays a major function in Industry 4.0, the most recent industrial revolution. For robots in motion, planning a path has been a significant area of research throughout their evolution. The steps needed to navigate an environment full of obstacles in the best way are referred to as "path planning" [1]. These issues have to be taken into consideration by AGV path planning algorithms. The optimization goals of routing computations are the shortest route, the least path nodes, and minimizing turn magnitude. These computations are made to take an autonomous vehicle from its current position to its objective location, avoid obstacles, and improve its motion track [2]. The local path planner is in service of making changes in real time so that the path stays on track with the global path and avoids any moving or fixed obstacles that the global path planner doesn't see. The vehicle's navigation system uses both methods of planning. Effective global and local path planning are crucial for autonomous vehicle navigation, particularly for mecanum wheeled-based vehicles. They have different goals and use different methods of navigation. When the AGV operates in a static or dynamic environment, there is a need to determine its path planning type [3]. Using

© The Author(s), 2024. Published by Cambridge University Press.

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

real-time local path and sensor data integration, AGV may maneuver in unexpected and dynamic environments [4]; however, because it heavily relies on the environment around it, dynamic route planning alone might not be sufficient to identify the completely ideal route [5]. Before the AGV moves, the global perfect path is found by scanning the current environment map model utilizing a static path planning mechanism. With unparalleled flexibility and efficiency, omnidirectional mecanum wheels on AGVs transformed complex industrial mobility. Traditional path planning and control algorithms struggle with static and dynamic real-time control. Conventional techniques such as A* can provide optimal pathways, but they aren't appropriate for dynamic applications since they are computationally costly and lead to jerky trajectories. Although effective, Probabilistic Roadmap (PRM)-based global travel planning is not adaptable in real-time or in densely populated places. Large regions can be rapidly investigated with Random Tree (RRT) and Bidirectional RRT (Bi-RRT) methods, but they also generate less-than-ideal pathways that must be relaxed after processing, adding to the computational load [6].

While state-of-the-art techniques such as the Modified Elman Recurrent Neural Network are effective for learning-based dynamic control, their relevance to real-world AGV navigation is constrained by their difficulty with standardization in unstructured circumstances and their enormous data set requirements [7]. Hybrid swarm optimization techniques integrate various meta-heuristics to optimize routes; yet, they are slow for convergence and ineffective when running in real time [8]. Kinematic path-tracking controls employ the Back-Stepping Slice Genetic robust Algorithms being durable but computationally costly and challenging to scale in applications that require real-time control [9]. Recent hybrid algorithms like RRT*-PSO [10] and the A*–FAHP Hybrid Algorithm [11] incorporate path planning with Particle Swarm Optimization [12] and Fuzzy Analytic Hierarchy Process [13]; however, they have significant computational costs and impoverished convergence in unpredictable circumstances. A more durable and effective solution is needed due to these limitations. To solve these issues, this article suggests a Hybrid Modified A* method that combines modified A*'s path optimal with PSO real-time adaptability and efficient swarm-based techniques and compares with A*, PRM, RRT, and Bi-RRTs to prove the effectiveness of HMA*. This complete method simplifies and speeds up omnidirectional AGV path planning in complicated and dynamic contexts.

AGV setups with omnidirectional mecanum wheels are highly useful due to their distinctive feature of being able to move in any direction regardless of orientation. This feature allows them to move across restricted spaces with more dexterity and accuracy, which makes it ideal for complicated and rapid industrial environments [14]. Although mecanum wheel AGVs have several benefits, operating and maneuvering them is very challenging. Precise and efficient path planning is necessary to fully utilize them, and this necessitates the inclusion of advanced algorithms capable of handling the intricacies of omnidirectional movement. Traditional path planning algorithms, such as A*, are still commonly used, but they often fall short of optimizing routes for mecanum wheel AGVs due to their reliance on predetermined movement patterns and restrictions when it comes to handling dynamic obstacles [15].

The Hybrid Modified A* algorithm was created especially for omnidirectional mecanum wheel AGVs in order to overcome difficulties stated above. The proposed algorithm synergizes the robust pathfinding capabilities of the MA* algorithm with dynamic path optimization techniques tailored for omnidirectional navigation. Through the utilization of multi-sensor fusion, the algorithm is able to dynamically adapt to continuous changes in the surrounding environment, thereby guaranteeing the most effective route selection and avoiding obstacles.

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

The paper's outline is as follows: Section 2 provides literature review, and Section 3 provides the detail about AGV's locomotion. Section 4 provides a summary of Modified A* and PSO evolutionary algorithms, while Section 5 provides Hybrid Modified A* (HMA*) algorithm in detail. Section 6 presents the experiment of path planning and simulation. Section 7 examines obstacle environment-based simulation and real-time experiments. Section 8 offers experimental validation, and Section 9 gives the conclusion and the scope for the future.

2. Literature review

One of the biggest problems with the task is that the algorithms used to plan paths for autonomous vehicles are very complicated. Static route design employs procedures that can be broadly classified into artificial intelligence algorithms and heuristic algorithms [16]. Numerous heuristic techniques have been thoroughly studied, including A* [17], Dijkstra [18], D* [19], LPA* [20], RRT [21], and PRM [22]. A* algorithm based on guidelines, inverted length weighted interpolation for the D* Lite method [23], enhanced PRM algorithm [24], hybrid heuristic-based A* method [25], B-spline curve combined with A* [26], bidirectional alternate searching A* technique [27], RRT method with an APF background [28], Triangle inequality-based RRT-connect algorithm [29], and quickly growing arbitrary tree algorithm [30] are a few of the recent advances. Furthermore, a multitude of artificial intelligence optimization techniques, including genetic algorithms (GAs) [31], divergent evolutionary [32], ants colonies optimizing (ACO) [33], and particle swarm optimizing (PSO) [34], are employed in static path planning. The path planning process is optimized through the application of these solutions. Several novel methods, including enhanced PSO with minimum-maximum normalization [35], adaptive ACO [36], GA based on the Bessel curve [37], and PSO based on the smoothing principle [38], have been developed to solve difficulties in static path planning. A method integrating a logarithmic decline strategy with Cauchy perturbation, drawing on a bat method [39], a combination of whale optimization approach [40] and differential evolution for vehicle routing [41], an enhanced artificial fish swarm approach with continuous segmented Bessel curve for mobile robot trajectory planning [42], and optimization of chemical processes for global route planning [43] are a few examples of the new solutions to global path planning issues that have been made possible by the continuous evolution of AI optimization algorithms. Additionally, in order to circumvent the constraints that are associated with single-approach systems, hybrid path planning approaches have been investigated [44]. Two different static approaches are combined in these methods. Examples include a hybrid Dijkstra-BFS algorithm to enhance processing efficiency and path accuracy [45], a hybrid A*-ACO method to reduce running time and improve the planning of paths in complex environments [46], Dijkstra-ACO that optimizes the route using ACO after initial path planning [47], D*-GWO demonstrating its practicality and effectiveness [48], and ACPSO that combines PSO with the A* algorithm's heuristic function [49, 50]. While a single A* processing is quick but produces a lot of pivot points, extra nodes, and erroneous paths, a single PSO's path search is precise but struggles with population initialization in complex environments [51]. Most of the previous emphasis has been concentrated on enhancing the path smoothness and heuristic function of A* in static circumstances; there have been very few attempts to simultaneously enhance A* and PSO. The comparative performance of the algorithms found in literature review is as follows (Table I):

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

Table I. *Performance comparison of algorithms.*

Author	Approach	Advantages	Disadvantages
J. Guo, X. Huo et al. [17]	A* algorithm	• Optimal Path Finding: Guarantees the shortest path if a path exists. • Efficiency: Typically faster than Dijkstra's algorithm due to its use of heuristics. • Widely Used: Well-documented and understood, making it easy to implement and modify.	• Memory Intensive: Can consume a lot of memory for large maps. • Heuristic Dependency: The efficiency highly depends on the heuristic function used. • Not Suitable for Dynamic Environments: Struggles with real-time changes in the environment.
K. Wei, Y. Gao et al. [18]	Dijkstra algorithm	• Optimal Path: Guarantees the shortest path. • Simplicity: Conceptually simple and easy to implement. • Versatile: Can be used with various types of graphs (weighted, unweighted).	• Computationally Expensive: Time complexity is higher compared to A* when no heuristic is used. • Memory Usage: Requires significant memory for large graphs. • Inefficient for Large Networks: Performance degrades with the size of the network.
J. Guo et al. [19]	D* algorithm	• Dynamic Adaptation: Efficiently handles changes in the environment. • Optimal Paths: Provides optimal paths for the current known map. • Reusable Paths: Efficient for reusing paths by updating only the necessary parts.	• Complexity: More complex to implement compared to A* and Dijkstra. • Computational Overhead: Higher computational cost due to continuous updates. • Initial Setup Time: Takes more time to initially compute paths.
Karur, K et al. [20]	LPA* algorithm	• Dynamic Handling: Handles dynamic changes efficiently. • Incremental Updates: Updates paths incrementally, reducing computational overhead. • Optimal Path Finding: Provides optimal paths similar to A*.	• Complexity: More complex to understand and implement. • Memory Usage: Requires additional memory for storing incremental changes. • Overhead: Additional computational overhead for maintaining incremental updates.

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

Table I. *Continued.*

Author	Approach	Advantages	Disadvantages
Wang, X., Wei, J. et al. [21]	RRT (Rapidly-exploring Random Tree) algorithm	• Exploration Efficiency: Quickly explores large spaces. • Scalability: Scalable to high-dimensional spaces. • Simple Implementation: Relatively simple to implement.	• Suboptimal Paths: Does not guarantee the shortest path. • Completeness: May not always find a path if one exists. • Randomness: Can produce paths that are not smooth.
Li, Q. et al. [22]	PRM (Probabilistic Roadmap) algorithm	• Efficient Preprocessing: Good for multi-query environments. • Scalability: Scales well to high-dimensional spaces. • Smooth Paths: Produces smoother paths compared to RRT.	• Memory Usage: Requires significant memory for storing roadmaps. • Initialization Time: Long preprocessing time for building the roadmap. • Path Quality: Path quality can vary depending on the roadmap.
S. He, T. Xing et al. [23]	D star algorithm (Lite method)	• Real-Time Updates: Efficiently updates paths in real-time. • Optimality: Provides optimal paths. • Lower Computational Cost: Less computationally intensive than the original D* algorithm.	• Complexity: Implementation complexity can be high. • Memory Usage: Requires additional memory for path updates. • Initial Computation: Initial path computation can be time-consuming.
M. P. Gopika et al. [24]	PRM algorithm (improved)	• Path Quality: Produces higher quality paths compared to basic PRM. • Efficiency: More efficient in finding paths in complex environments. • Scalability: Handles high-dimensional spaces well.	• Complex Implementation: More complex to implement than basic PRM. • Computational Cost: Higher computational cost due to additional improvements. • Memory Usage: Increased memory usage for storing enhanced roadmaps.

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

Table I. *Continued.*

Author	Approach	Advantages	Disadvantages
Katiyar S et al. [28]	APF-based RRT algorithm	• Path Quality: Produces smoother paths by incorporating artificial potential fields. • Obstacle Avoidance: Improved obstacle avoidance capabilities. • Exploration Efficiency: Maintains efficient exploration of the environment.	• Complexity: More complex to implement due to the combination of APF and RRT. • Computational Cost: Higher computational cost for path planning. • Parameter Tuning: Requires careful tuning of parameters for optimal performance.
Wang J et al. [30]	RRT-connect algorithm	• Fast Path Planning: Faster than basic RRT due to bidirectional search. • High Success Rate: Higher success rate in finding paths. • Simplicity: Relatively simple to implement.	• Suboptimal Paths: Does not guarantee the shortest path. • Randomness: Can produce non-smooth paths. • Memory Usage: Requires additional memory for bidirectional search trees.
Xiaohua Cao et al. [34]	PSOs (Particle Swarm Optimization) algorithm	• Optimization Capability: Effective for global optimization problems. • Scalability: Scales well to high-dimensional problems. • Simplicity: Simple and easy to implement.	• Convergence Speed: Can have slow convergence. • Local Minima: Prone to getting stuck in local minima. • Parameter Sensitivity: Performance depends on parameter settings.
V. Ruiz Molledo et al. [35]	Adaptive ACO (Ant Colony Optimization) algorithm	• Optimization Efficiency: Efficient in finding optimal paths. • Dynamic Handling: Adapts well to changes in the environment. • Scalability: Scales well to large and complex environments.	• Computational Cost: High computational cost due to multiple iterations. • Parameter Tuning: Requires careful tuning of parameters for optimal performance. • Complexity: More complex to implement and understand.

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

Table I. *Continued.*

Author	Approach	Advantages	Disadvantages
Rong Lin et al. [36]	GA based on the Bessel curve algorithm	• Path Smoothness: Produces smooth paths due to Bessel curve integration. • Optimization Capability: Effective for global optimization. • Adaptability: Adapts well to various environments.	• Complexity: Complex to implement and understand. • Computational Cost: Higher computational cost due to genetic algorithms. • Convergence: Convergence can be slow.
Si, Q et al. [41]	whale optimization approach and differential evolution algorithm	• Optimization Efficiency: Effective in global optimization. • Simplicity: Simple and easy to implement. • Adaptability: Adapts well to various types of optimization problems. • Simplicity: Simple to implement and understand. • Robustness: Robust to changes in the environment.	• Convergence Speed: Can have slow convergence. • Parameter Sensitivity: Performance is sensitive to parameter settings. • Local Minima: liable to become confined in nearby minima. • Parameter Tuning: requires precise parameter adjustment. • Complexity: Implementation can be complex for certain problems.
M. Jiang et al. [42]	Enhanced artificial fish swarm approach	• Optimization Capability: Effective for global optimization problems. • Adaptability: Adapts well to dynamic environments. • Scalability: Scales well to high-dimensional problems.	• Computational Cost: High computational cost due to complex operations. • Parameter Tuning: Requires careful tuning of parameters for optimal performance. • Implementation Complexity: More complex to implement compared to basic fish swarm approaches.
Yassami, M. et al. [45]	Hybrid Dijkstra-BFS algorithm	• Efficiency: Combines the strengths of Dijkstra and BFS for efficient pathfinding. • Scalability: Scales well to large graphs. • Optimal Paths: Provides optimal paths.	• Complexity: More complex to implement than individual algorithms. • Memory Usage: Requires significant memory for large graphs. • Initial Computation: Initial path computation can be time-consuming.

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

Table I. *Continued.*

Author	Approach	Advantages	Disadvantages
S. Abi et al. [46]	Hybrid A*-ACO method	• Path Optimality: Provides high-quality paths by combining A* and ACO. • Efficiency: Efficient in finding paths in complex environments. • Adaptability: Adapts well to dynamic changes.	• Complexity: Complex to implement and understand. • Computational Cost: Higher computational cost due to the combination of algorithms. • Parameter Tuning: Requires careful tuning of parameters for optimal performance.
Z. Nie et al. [47]	Dijkstra-ACO algorithm	• Optimization Capability: Combines the strengths of Dijkstra and ACO for optimal paths. • Dynamic Handling: Adapts well to changes in the environment. • Path Quality: Provides high-quality paths.	• Complexity: More complex to implement than individual algorithms. • Computational Cost: Higher computational cost due to the combination of algorithms. • Memory Usage: Requires additional memory for storing paths.
G. Shial et al. [48]	D*-GWO (Grey Wolf Optimizer) algorithm	• Optimization Efficiency: The GWO can help the D* algorithm find optimal paths in complicated scenarios. • Dynamic Handling: Effective in dynamic contexts due to D*'s real-time adaptability and GWO's global optimization. • Exploration and Exploitation Balance: Solution quality improves with GWO's exploration-exploitation balance.	• Complexity: Integration of GWO with D* makes the algorithm harder to implement and understand. • Computational Cost: Optimization and real-time path updates may increase computing costs in the combined strategy. • Parameter Sensitivity: For best hybrid algorithm performance, GWO parameters must be carefully tuned.

Several hybrid approaches have been proposed, some of which are as follows Dijkstra-BFS, A*-ACO, and Dijkstra-ACO hybrid techniques are examples; however, these approaches primarily address specific constraints of individual algorithms instead of offering a comprehensive solution that tackles multiple issues simultaneously. These innovations have not been properly integrated with heuristic approaches in order to make use of their combined strengths, despite the fact that several research have proposed improvements to PSO and ACO, such as adaptive ACO and PSO with smoothing principles. A lack of uniformity is frequently observed in the evaluation of algorithms, with different measurements and experimental setups being utilized. Because of this inconsistency, it is difficult to arrive at conclusive results regarding the effectiveness of various treatments [52]. PSO is known for its sensitivity to initial population settings. Although there has been a lot of research on individual improvements to heuristic algorithms like A*, Dijkstra, and RRT, as well as AI optimization techniques like PSO and ACO, there aren't the comprehensive solutions that successfully combine the advantages of these methods to address multiple constraints at once. Dijkstra-BFS, A*-ACO, and Dijkstra-ACO hybrid algorithms optimize processing speed and path correctness but often focus on technique restrictions [53].

Hybrid methods increase efficiency in challenging circumstances by solving multiple problems at once while offering a broader response. Robust, effective, and adaptive algorithms to handle AGVs' many real-world challenges are lacking. The limitations of this demonstrate the need for additional study and improvement in path planning for AGVs. These domains mostly concern the development of algorithms that are more dependable, efficient, and flexible. The Hybrid Modified A* (HMA*) algorithm, which combines Modified A* with PSO, was created in response to this situation. The Modified A* algorithm first establishes the initial path, from which duplicate point elimination is used to remove essential nodes. The best global path is subsequently identified by PSO, which is seeded with these significant nodes and mixes a random opposition-based method of learning with a random inertia weight. AGV path planning models are used to assess a method with a more expansive goal function. Its unique features are below:

- The Hybrid Modified A* method improves multiple constraints by combining MA* heuristics with Particle swarm optimization.
- The algorithm's dynamic heuristic enhances path planning precision by adjusting to changing environmental conditions and PSO findings. The Hybrid Modified A* uses PSO to optimize the initial population, that is, minimizing the dependence on initial population decisions.
- In hybrid technique, the PSO's multi-objective capabilities optimize path length, smoothness, and obstacle avoidance simultaneously. The system responds to environmental changes, constructing it ideal for instantaneous applications with unexpected challenges.
- The hybrid approach finds optimal or near-optimal routes faster than either alone by MA* and PSO. The suggested approach is flexible for multiple uses as it operates under various AGV types and operating circumstances.

3. Locomotion of the AGV

AGVs with four Mecanum wheels are agile and can go anywhere. Each roller on a Mecanum wheel is 45 degrees; thus, the AGV can go sideways, diagonally, forward, and backward without moving orientation. Mecanum wheel AGVs are ideal for flexible, accurate industrial applications since they can negotiate confined spaces and complex situations. By managing each wheel's speed and direction, the AGV can conduct precise actions, including spinning on the spot, improving its operational efficiency in multiple settings [54].

As shown in Figure 1, four-wheel omnidirectional AGV is made up of four Mecanum wheels. The following is the nomenclature of symbols used, ψ is the constant slope angle of the rollers, O_r is the moveable AGV coordinate framework, O_q is an abbreviation for fixed coordinating framework, O_{wi} is the wheel coordinating framework, P_{wi} is the wheel position vector in O_{wi} [54].

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

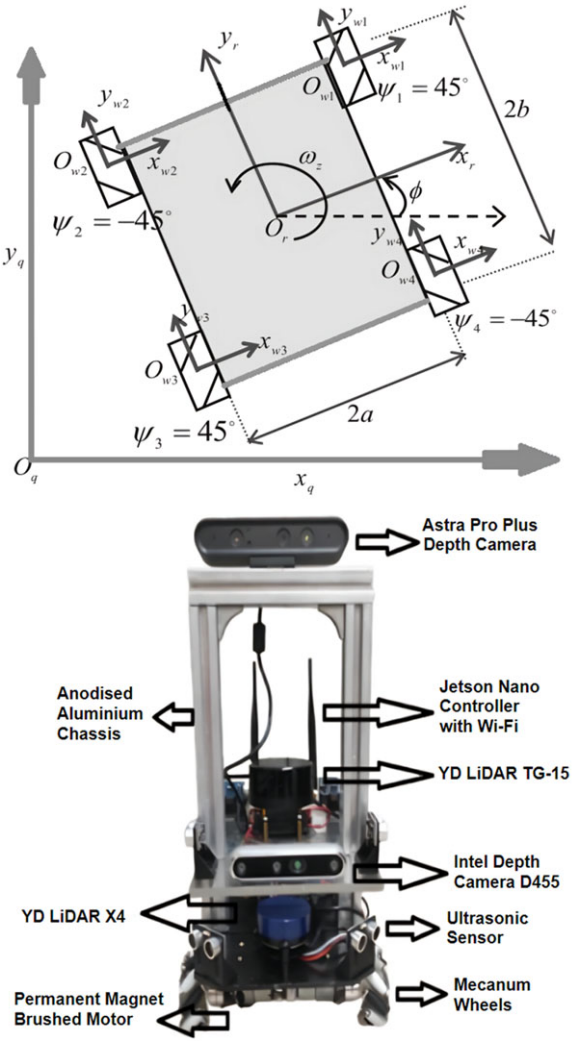


Figure 1. *The schematic of automated guided vehicle.*

The rollers around the edges of these wheels are tilted at a constant angle. For this case, the angle ψ is 45 degrees, which means that the wheel can move easily at an angle of 45 degrees with the motion that is pushing it. Each wheel has a permanent magnet direct current motor connected to it. This motor provides the torque that the AGV needs to move. It is now considered that the AGV is moving across a level, uniformly distributed surface. To discover the equation of motion, it is presumed that each component of the AGV, including the wheels, is rigid. We make use of the O_r and O_q coordinate frames within the framework of the kinematic modeling scenario as in ref. [54, 65].

There is a wheel coordinate frame denoted by O_{wi} (where i = 1 to 4). P_{wi} is the wheel's position vector in O_{wi} as in ref. [55]:

$$P_{wi} \text{ (where i = 1 to 4)} = [x_{wi} y_{wi} \varphi_{wi}]^T \tag{3a}$$

$\dot{\theta}_{ix}$ (where i = 1 to 4) shows how fast the wheel is turning around the hub, $\dot{\theta}_{ir}$ (where i = 1 to 4) represent the roller angular velocity, and $\dot{\theta}_{iz}$ (where i = 1 to 4) represents the wheel angular velocity around the contact point where it is located. R_i (where i = 1 to 4) represents the radius of the wheel, ψ_i

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

(where i = 1 to 4) is the angle of the roller slope for each wheel, and a roller's radius is denoted by the letter r. This is how we can write the AGV's motion vector as in ref. [55]:

$$\dot{\mathbf{P}}_{wi} = \begin{bmatrix} \dot{x}_{wi} \\ \dot{y}_{wi} \\ \dot{\phi}_{wi} \end{bmatrix} = \begin{bmatrix} 0 & r\sin(\psi_i) & 0 \\ R_i & -r\cos(\psi_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\theta}_{ix} \\ \dot{\theta}_{ir} \\ \dot{\theta}_{iz} \end{bmatrix} \tag{3b}$$

Let

$$\mathbf{T}^{\mathbf{r}}_{\mathbf{wi}} = \begin{bmatrix} \cos\left(\phi^r_{wi}\right) & -\sin\left(\phi^r_{wi}\right) & d^r_{wiy} \\ \sin\left(\phi^r_{wi}\right) & \cos\left(\phi^r_{wi}\right) & d^r_{wix} \\ 0 & 0 & 1 \end{bmatrix} \tag{3c}$$

In the given equation, the rotating angle of O_{wi} concerning O_r is denoted by ϕ^r_{wi}(where i = 1 to 4). In addition, the distance in radians between two coordinate frames is shown byd^r_{wix} and d^r_{wiy}.

Let us say that the equation for the AGV's position vector in O_r is $P_r = [x_r\ y_r\ \phi_r]^T$, then the relationship between P_r and P_{wi} may be expressed as $P_r = T^r_{wi}.P_{wi}$. Furthermore, it is evident from Figure 1 that the value of $\phi^r_{w1} = \phi^r_{w2} = \phi^r_{w3} = \phi^r_{w4} = 0$. Consequently, the AGV's velocity vector may be stated as [65]:

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\phi}_r \end{bmatrix} = \begin{bmatrix} 1 & 0 & d^r_{wiy} \\ 0 & 1 & d^r_{wix} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x}_{wi} \\ \dot{y}_{wi} \\ \dot{\phi}_{wi} \end{bmatrix} \tag{3d}$$

From (3b) and (3d) we get,

$$\dot{\mathbf{P}}_{\mathbf{r}} = \mathbf{J}_{\mathbf{i}}\dot{\mathbf{q}}_{\mathbf{i}} \tag{3e}$$

where

$$\mathbf{J}_{\mathbf{i}} \in R^{3\times 3} = \begin{bmatrix} 0 & r\sin(\psi_i) & d^r_{\mathrm{wiy}} \\ R_i & -r\cos(\psi_i) & d^r_{\mathrm{wix}} \\ 0 & 0 & 1 \end{bmatrix}$$

$\mathbf{J_i}$ is the Jacobian matrix of the ith wheel and

$$\dot{q}_i = \begin{bmatrix} \theta_{ix} & \theta_{ir} & \theta_{iz} \end{bmatrix}$$

Remark 1: |Ji|=0 and $\psi_i = 0$

Consequently, Mecanum wheels do not exhibit any signs of singularity.

Remark 2: rank(J_i) = 3

Therefore, there are three degrees of freedom associated with each wheel.

As each of the four wheels is the same, the following equations are used to figure out its geometric and kinematic properties:

$R_1 = R_2 = R_3 = R_4 = R$ and,

$d^r_{w1x} = a$; $d^r_{w1y} = b$; $d^r_{w2x} = -a$; $d^r_{w2y} = b$; $d^r_{w3x} = -a$; $d^r_{w3y} = -b$; $d^r_{w4x} = a$; $d^r_{w4y} = -b$

Regarding this, the Jacobian matrix for each wheel can be found as follows:

$$\mathbf{J_1} = \begin{bmatrix} 0 & r/1.414 & b \\ R & -r/1.414 & a \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{J_2} = \begin{bmatrix} 0 & -r/1.414 & b \\ R & -r/1.414 & -a \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{J_3} = \begin{bmatrix} 0 & r/1.414 & -b \\ R & -r/1.414 & -a \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{J_4} = \begin{bmatrix} 0 & -\frac{r}{1.414} & -b \\ R & -\frac{r}{1.414} & a \\ 0 & 0 & 1 \end{bmatrix} \tag{3f}$$

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

The solution to the inverse kinematics problem can be obtained by using equations (3e) and (3f) as in ref. [56].

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\phi}_r \end{bmatrix} = \frac{R}{4} \begin{bmatrix} -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 \\ 1/a+b & -1/a+b & -1/a+b & 1/a+b \end{bmatrix} \times \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{\theta}_4 \end{bmatrix} \tag{3g}$$

where $\dot{\theta}_i$ (where i = 1 to 4) is the angular velocity of each wheel.

Remark 3: It is possible for the AGV to follow any desired trajectory even if the value of ϕ_r is equal to zero. This is something that can be observed from the kinematic solution that has been constructed. As an illustration, the AGV need not alter its orientation if it follows a curved path, one of the primary justifications for its usage in cramped spaces. Within the global coordinate system O_q, the velocity vector may be expressed as follows since position and rotation are closed-loop feedback [54]:

$$\dot{\mathbf{P}}_\mathbf{q} = \left[\dot{x}_q \dot{y}_q \dot{\phi}\right]^T = \mathbf{R}(\boldsymbol{\varphi})\dot{\mathbf{P}}_\mathbf{r} \tag{3h}$$

where

$$\mathbf{R}\,(\varphi) = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

is the rotation matrix of O_r with regard to O_q.

The above equations are essential for finding out the wheel velocities and the needed revolutions per minute (rpm) necessary for a mecanum wheel AGV in order to accomplish the required linear and angular motions in various environments for real-time obstacle avoidance and path planning. The next section discusses about the algorithms that are implemented on the AGV for its control in static and dynamic environments.

4. Modified A* and PSO evolutionary algorithms

By the implementation of Modified A*, the conventional A* algorithm, which is frequently used in the process of path finding and network travel, is enhanced. In contrast to A*, which finds the shortest path by calculating the expense of traveling from one location to another using a heuristic, MA* usually performs modifications for the intention of improving the computational effectiveness, accuracy, or flexibility of the system in surroundings that are complex and dynamic [55].These changes might include the following:

Dynamic re-planning: Altering current routes to take account of new circumstances or impediments. Heuristic improvement is a method of adjusting the heuristic function to the specific issue in order to incorporate discovery and extraction in an appropriate manner.

Integration of hybrid systems: To overcome its limitations in extremely dynamic or unpredictable environments, MA* is sometimes used in conjunction with other optimization techniques such as PSO.

PSO is an algorithm for evolution that draws influence from the way groups of animals, such as fish or birds, behave together. A population, or swarm, of particles looks for the most suitable solution in PSO by traveling over the correct space [34]. Every particle modifies its place of existence according to two criteria.

- *Personal best (pBest):* A fundamental idea in PSO, pBest, or personal best, is the finest location that a single particle possesses thus far while moving across the solution space. When moving, the particle assesses the objective function at its current position. For that particle, the current position becomes the new pBest if it provides a better objective value than the previous position.
- *Global best (gBest):*The best position or solution that each particle in the swarm discovered after all iterations is often referred as gBest. The gBest place is one that behaves the greatest among all the particle locations at every iteration, determined using an objective function.

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

4.1. The modified–A (MA*) algorithm*

The MA* algorithm merges the Dijkstra algorithm with Breadth-First Search (BFS), combining Dijkstra's heuristic global search with BFS's systematic approach [57]. This fusion is known for its robustness and effectiveness in AGV path planning. The algorithm commences at the initial node and evaluates its child nodes, processing them in turn. It continues this process until the target node is found and added to the OpenList. The next node to process is chosen based on the lowest A(n) value, using the function formula to estimate costs as in ref. [58].

$$A(n) = B(n) + C(n) \tag{4.1a}$$

In the MA* algorithm, the estimated total cost A(n) for achieving the target from the start to the current position is divided into two parts, that is B(n), the real cost, and C(n), the heuristic estimate [58]. Manhattan distance, which aggregates absolute coordinate differences, or the distance from Euclid, the distance in a straight line, can be used by C(n). The heuristic chosen is Euclidean distance, in this study because of its accuracy, and the algorithm uses an eight-way search on a raster map with obstacles [59].

$$\begin{aligned} B\ (n) &= [(y_s - y_n)^2 + (z_s - z_n)^2]^{1/2} \\ C\ (n) &= [(y_n - y_t)^2 + (z_n - z_t)^2]^{1/2} \end{aligned} \tag{4.1b}$$

Here, y_n and z_n are the current node's abscissa and ordinate, respectively, and y_s and z_s are for the source or starting nodes. The target node's abscissa and ordinate are y_t and z_t (Figure 2).

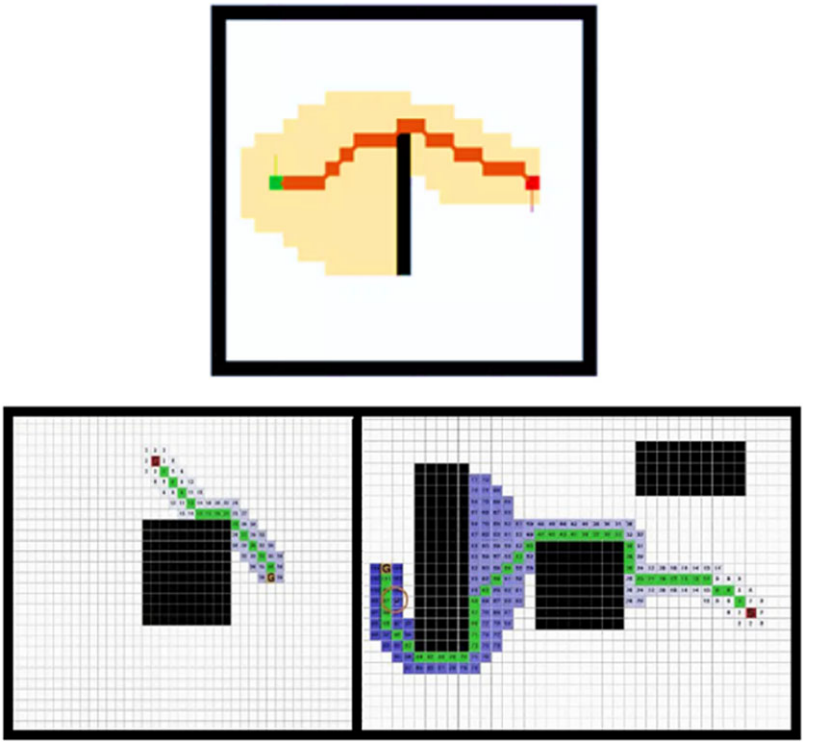


Figure 2. *Example of MA* algorithm.*

Two sets are created from all extended nodes by the MA* algorithm:

- OpenList: There are nodes in the OpenList that have been found but not thoroughly investigated. The algorithm selects the node having less overall estimated cost among these nodes, which are candidates for expansion.

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

- CloseList: Nodes that have previously been processed are stored in the CloseList to avoid having to be revisited. This eliminates the requirement for repeated calculations and improves search times by avoiding needless node expansion.
- Together, the OpenList and CloseList allow the MA* algorithm to effectively achieve its objective through enabling it to systematically examine the most probable paths, removing recurrent checks and loops.

The MA* algorithm has many advantages like faster path finding, efficient handling of dynamic environments, reduced computational complexity, improved memory utilization, better scalability, more accurate heuristics, handling of multiple objectives, and smoother path generation as compared to traditional A* algorithm. Below is the compression between A* and MA* algorithm.

The A* algorithm, tested using MATLAB 2023 on an Intel i7 workstation, demonstrates the following results on 10 x 10, 20 x 20, and 30 x 30 grid maps (Figure 3).

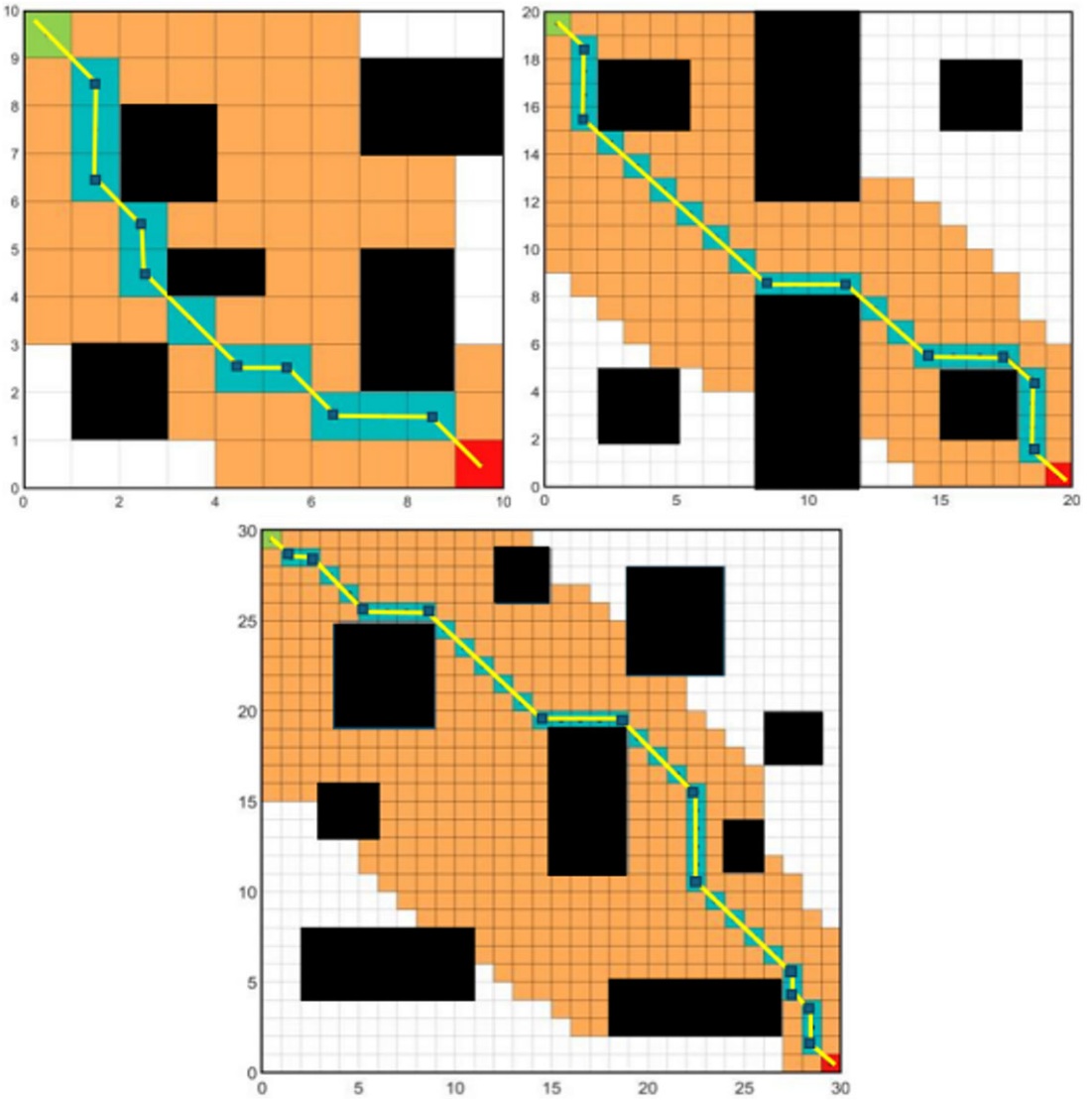


Figure 3. *Matlab results based on the conventional A* method's simulation in various map contexts. (a) The map measures 10 by 10. (b) The map measures 20 by 20. (c) The map measures 30 by 30.*

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

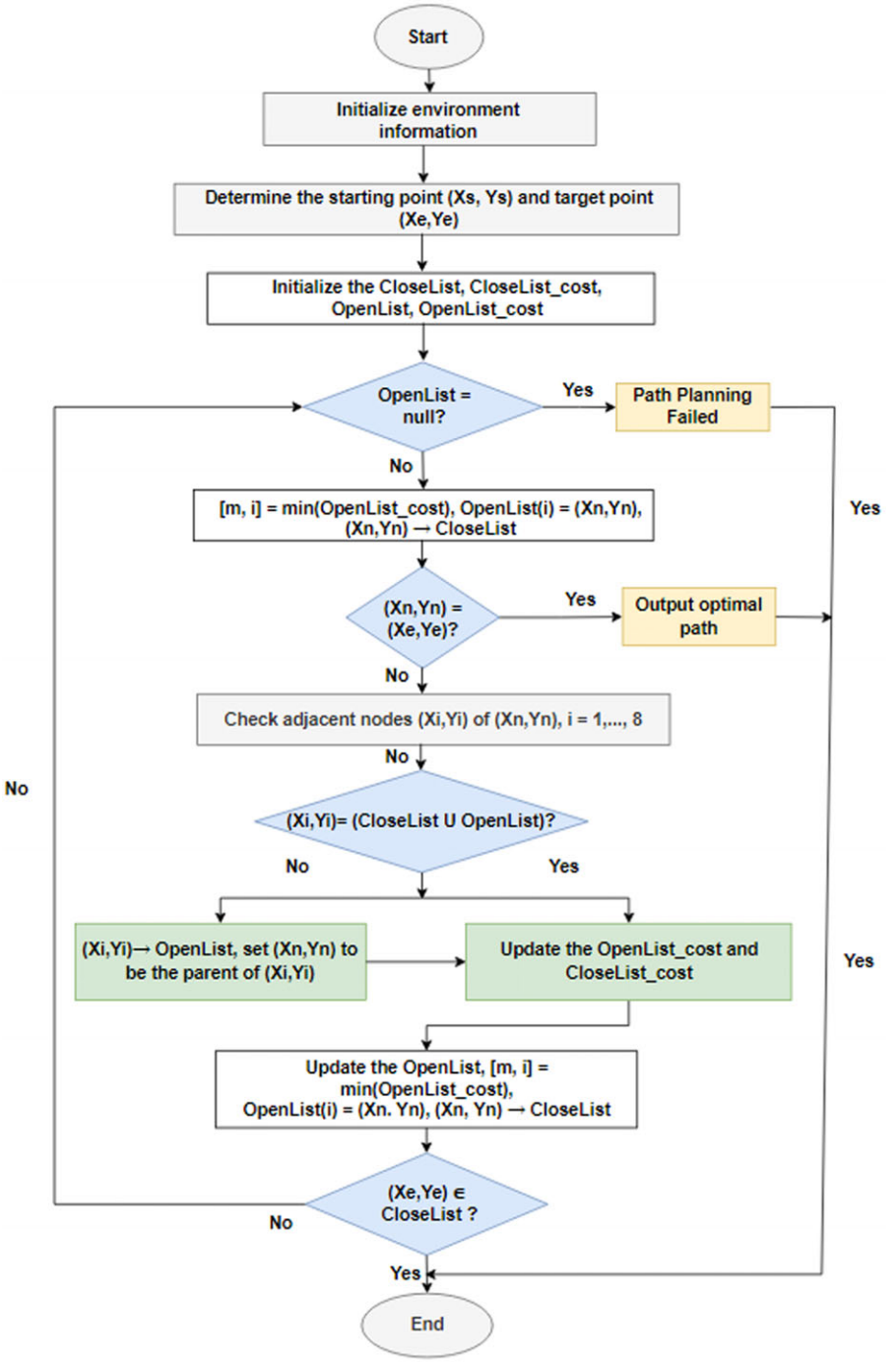


Figure 4. *Workflow of the MA* algorithm program.*

The workflow of the MA* algorithm is shown in Figure 4. The Modified A* (MA*) algorithm, tested using MATLAB 2023 on an Intel i7 workstation, demonstrates superior efficiency compared to the traditional one. Tested on 10 x 10, 20x 20, and 30 x 30 grid maps, the MA* algorithm decreases the number of nodes searched, minimizes turn points, and shortens computation time, as shown in Table II. While the overall path length reduces as compared to A*, MA* excels in optimizing route smoothness and driving performance, especially as map size increases [60] (Figure 5).

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

Table II. *Comparative analysis of A* and MA*.*

Size of the map (in Units)	Algorithm	Calculated average time (in seconds)	Nodes (in numbers)	Length of the path (in Units)
10 X 10	A*	23.94	64	14.48
	MA*	13.51	38	14.31
20 X 20	A*	38.03	188	30.37
	MA*	32.38	82	27.28
30 X 30	A*	43.60	440	45.68
	MA*	35.70	128	43.56

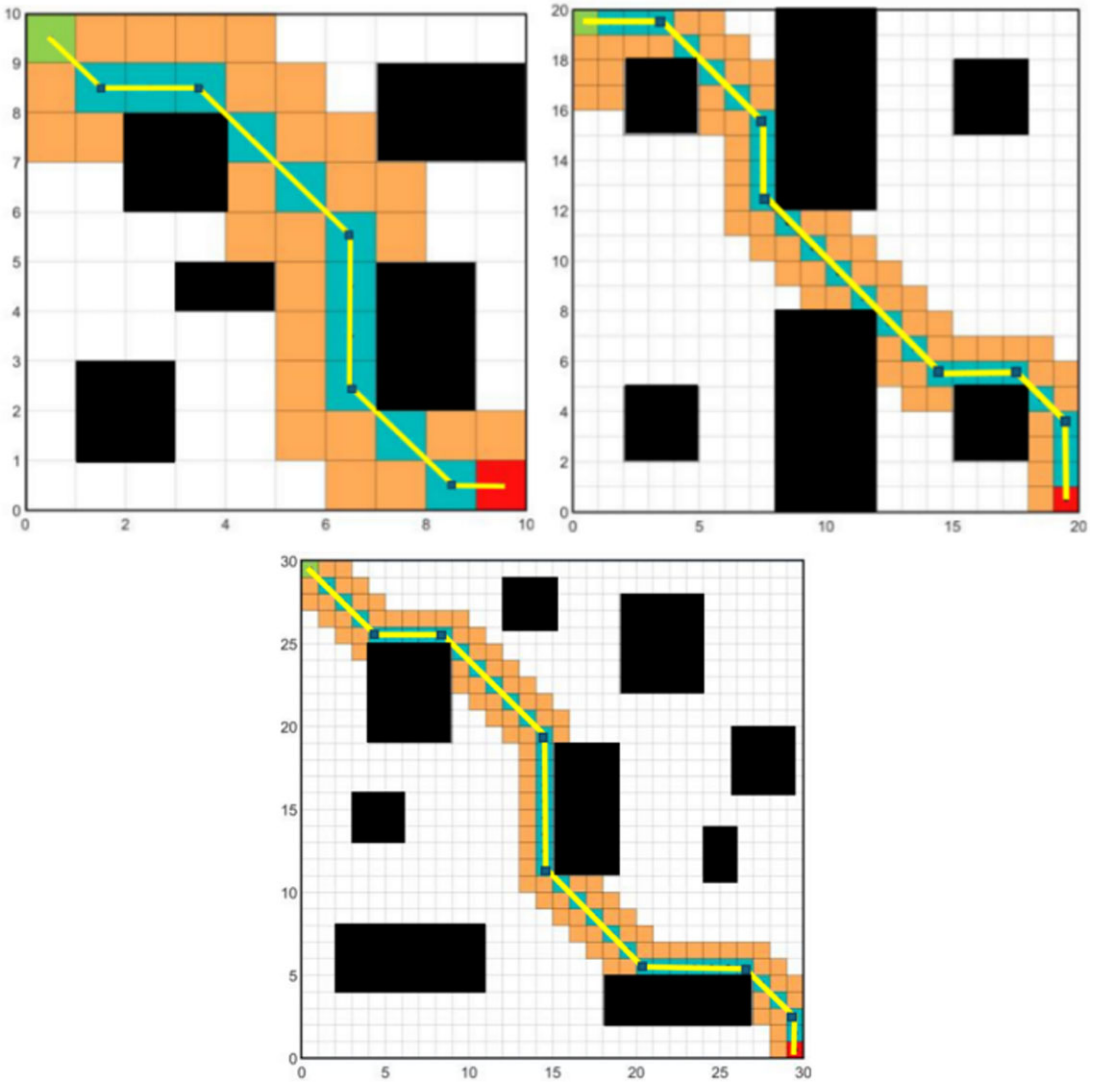


Figure 5. *Matlab results of the MA* algorithm's simulation in various map contexts. (a) The map measures 10 by 10. (b) The map measures 20 by 20. (c) The map measures 30 by 30.*

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

4.2. Particle swarm optimization (PSO) algorithms

From bird feeding data, Kennedy and Eberhart created Particle Swarm Optimization (PSO) [61]. A cluster of randomly initialized particles indicating potential fixes searches for the ideal solution in a multidimensional space in PSO. The simplicity and fast convergence of PSO make it popular in flow shop scheduling, medical picture classification, nonlinear power system optimization, and AGV path planning [61, 62].

The PSO method iteratively refines particle positions and velocities to obtain the optimal solution by evaluating Gbest and Pbest. As an example, the following equation shows how its position and speed are updated [62]:

$$v_i^{j+1} = \omega v_i^j + c_1 r_1 (Pbest_i^j - x_i^j) + c_2 r_2 (Gbest^j - x_i^j)$$
$$x_i^{j+1} = x_i^j + v_i^{j+1} \tag{4.2a}$$

An increase in the potential for self-learning factor denoted by c_1 of a bird increases the bird's incentive to arrive in its most advantageous spot, where ω represents inertia, that is indicates the amount of weight that is carried by flying to its ideal place [62]. v_i^{j+1} indicates the i-th particle's total process velocity in the $(j+1)$th iteration. A bird is more likely to fly to the group's optimum position if its social learning factor, or c_2, is high. Here, r_1 and r_2 are uniformly distributed random values, and $pbest_i^j$is the particle's optimal solution identified in the i-th search after j iterations. After the j^{th} iteration, $Gbest^j$ is the optimal solution. The symbol x_i^j positions the i-th particle after j repetitions [63]. The Workflow of the PSO algorithm is shown below with Matlab 2023 simulation result (Figures 6 and 7).

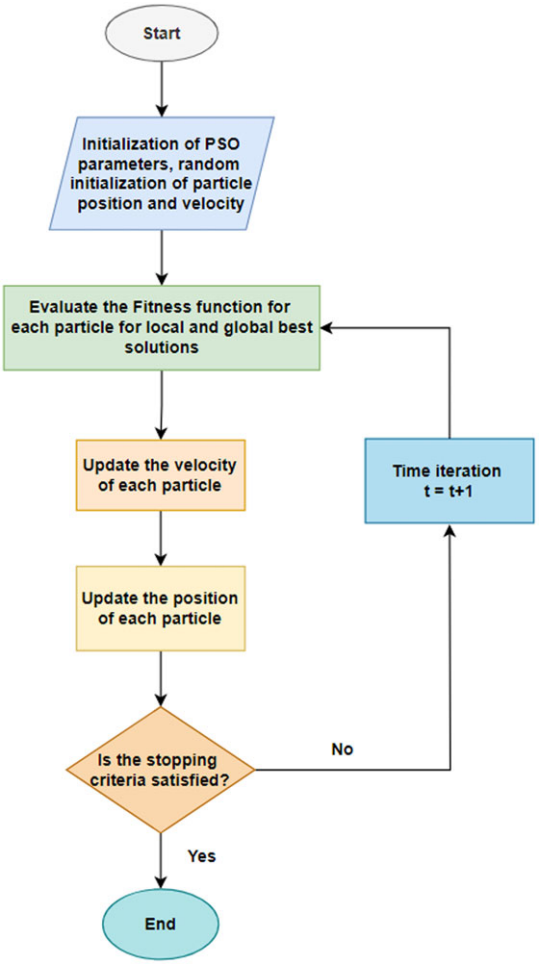


Figure 6. *Workflow of the PSO algorithm.*

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

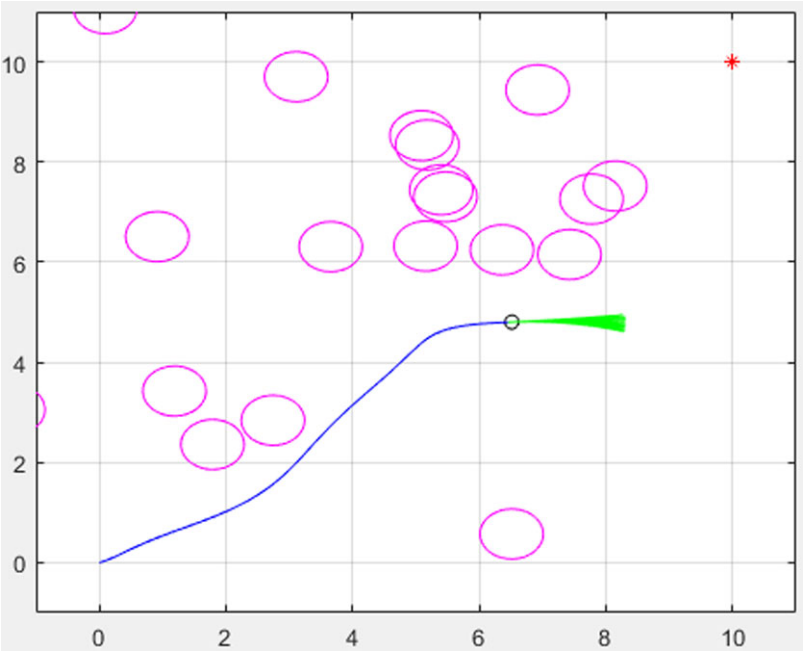


Figure 7. *PSO algorithm implementation in Matlab.*

5. Implementing the Hybrid Modified A* algorithm

The Hybrid Modified A* (HMA*) method combines PSO and modified A* (MA*) to improve path finding. The method of "particle swarm optimization," or "PSO," optimizes populations by modifying particle placements and speeds. This technique also considers fish and avian behavior [64]. By using MA*'s ordered search and PSO's optimization, HMA* can find network routes efficiently and effectively while improving path quality. HMA* improves the path planning and navigation for automated guided vehicle. Dynamic heuristics adapt to environmental changes, enhancing accuracy of the system [64]. HMA* changes paths in real time for best routing accuracy and smoother navigation in congested areas, unlike PRM, which employs pre-computed maps [12]. HMA* provides high-performance obstacle avoidance, and post-processing enables stable paths [65]. Lowering search space and processing time improves HMA*'s scalability and computational efficiency, making it suited for complex, dynamic AGVs.

5.1. Steps of algorithm

Step 1: The raster method creates the raster map and initiates the MA* method's parameters.
Step 2: Make two void sets, designated as *close*List and *open*List. The nodes that have been investigated are included in a *close*List, whilst the nodes that have not yet been explored are included in an *open*List.

After the first node has been incorporated into *open*List, the current node will be demolished if it is on a path that includes both the previous and next nodes in the current node's sequence, indicating that the node is redundant. For example, node A will be eliminated if it is collinear with node S, the node

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

before it, and node B, the node after it. The redundant path node's judging process can be written as in ref. [59]:

$$J_1 = \frac{z_n - z_{n-1}}{y_n - y_{n-1}}$$
$$J_2 = \frac{z_{n+1} - z_n}{y_{n+1} - y_n} \tag{5.1a}$$

For the current node n, the abscissa and ordinate are represented by y_n and z_n, respectively. On the other hand, y_{n-1} and z_{n-1} are meant to represent the node $n-1$, and similarly, y_{n+1} and z_{n+1} are for node $n+1$. If J_1 is equals to J_2, the present node is redundant and co-linear with both the nodes that came before it and the nodes that will come after it.. If J_1 is not equal to J_2, then the present node is not redundant and does not have any co-linearity with the nodes that came before or after it.

Step 3: Determine which node in *open*List has the lowest f(n) value, and then transform that node as an active node. When evaluating duplicated transition nodes, the implicit function listed below is employed as in ref. [59]:

$$\begin{aligned} \text{path} &= f(y, z) = \left(\frac{z - z_{n-1}}{z_{n+1} - z_{n-1}}\right) - \left(\frac{y - y_{n-1}}{y_{n+1} - y_{n-1}}\right), \\ obs &= f(y, z) = (y - y_{obs})^2 + (z - z_{obs})^2 - r_{obs}^2 \end{aligned} \tag{5.1b}$$

Here, y_{obs} and z_{obs} represent each obstacle's abscissa and ordinate and r_{obs} is the obstacle area radius following expansion. There are barriers between the two turning sites when $path = obs$; in opposite case, that is when $path \neq obs$, the intermediate turning locations are superfluous or redundant because there are no obstacles between them.

Step 4: Remove the current node "n" and add it to *close*List rather than *open*List.

Step 5: The goal is to determine whether or not the node "n" to be targeted. In the event that it is the node of interest, the current node identified as 'n' should be used as the source node in order to scan the eight neighbors.

Step 6: It is necessary to ascertain which of the eight nearby neighbors of the points of node "n" are included in list, if they are not obstacles. If this is the case, figure out each node's f(n) value, choose whatever node has the lowest value of f(n) to be the upcoming node $n+1$, and then go back to step number four. Add it to *open*List and compute f(n) if it is not in *open*List.

Step 7: The redundant node removal technique is utilized to eliminate the points that were present in the initial path, that is, remove path nodes and transition points that are redundant in nature.

Step 8: When traveling through the remaining nodes in the route points set or path-points set , each PSO optimization iteration should start and terminate at the separated nodes n and $n+2$.

Step 9: One must provide the population no. (possize), the maximum no. of repetitions (T_{max}), the factor based on self-learning (c_1), the factor based on social learning (c_2), the inertia weights correlation coefficients (ω_{max} and ω_{min}), and the iteration to establish the PSO parameters. In order to enhance the PSO algorithm's performance, the adaptive inertia weight [66] is frequently employed.

$$\omega = (\omega_{max} + \omega_{min}) \times \left(\frac{p_{gbest}^{j}}{p_{gbest}^{j-1}}\right) - \frac{w_{max} \times j}{T_{max}} \tag{5.1c}$$

where ω_{min} is the minimum inertia weight; ω_{max} is the maximum inertia weight, the ideal fitness (global) at the j^{th} iteration is denoted by $p^{j}{}_{gbest}$, the ideal fitness (global) at the j-1^{th} iteration is denoted by $p^{j-1}{}_{gbest}$ and T_{max} represents the highest possible quantity of possible iteration.

The algorithm can quickly depart from the optimal local solution by changing the inertia weight to a random number with a predetermined range. The algorithm's search is improved as a result of these efforts, which also preserve the population's diversity. It is for this reason that this research suggests a

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

stochastic inertia weight by employing the following equation, which is derived from the linear declining inertia weight as in ref. [66].

$$\omega = \omega_{\max} - (\omega_{\max} - \omega_{\min}) \times \left(\frac{\mathrm{j}}{T_{\max}}\right) \times e^{\rho} \tag{5.1d}$$

ρ is a number that can be chosen at random and falls anywhere between −0.1 and 0.1.

Step 10: In order to produce a new historical ideal population, every element of the group undergoes disruption to the individual ideal gbest as well as the individual ideal pbest, respectively. This is done in order to build an entirely novel population. The new population is supposed to be increased as the main objective. The random technique, which can be expressed as follows, is used to first randomly generate the initial particles with a particular population size as in [67].

$$\begin{aligned} \text{pos y} &= \mathrm{y}_n + ((\mathrm{y}_{n+2} - \mathrm{y}_n) \times r_{\mathrm{o}}) \\ \text{pos z} &= \mathrm{z}_n + ((\mathrm{z}_{n+2} - \mathrm{z}_n) \times r_{\mathrm{o}}) \end{aligned} \tag{5.1e}$$

Here, the random particle coordinates are presented by *pos*y and *pos*z, respectively, and the node coordinates are represented by y_n and z_n, respectively.

The y and z coordinates of nodes that are divided from node n are denoted by the symbols y_{n+2} and z_{n+2}, respectively. A random number that falls somewhere between 0 and 1 is denoted by the symbol r_{o}. After the population has been initialized, a random approach is utilized to initialize the velocity of each individual particle once more as in ref. [68]. This can be represented as follows:

$$\begin{aligned} v_{\mathrm{y}} &= v_{\max} \times r_{\mathrm{o}} \\ v_{\mathrm{z}} &= v_{\max} \times r_{\mathrm{o}} \end{aligned} \tag{5.1f}$$

Here, v_{max} represents the maximum speed, and the velocities of each particle on the abscissa and ordinate coordinates are represented by v_y and v_z respectively.

Each particle's fitness is computed following the initialization of the population and population velocity. The particles that doesn't avoid barriers have a fitness value of zero, whereas the other particles lacking it are the individual optimum solution pbest. The most advantageous group solution, denoted by letter gbest, is found to be the particle that has the greatest efficiency or highest fitness. The fitness calculation is as follows [67, 68]:

$$\begin{aligned} f &= \mathrm{k} \times c, \\ \mathrm{k} &= \sum_{m=1}^{n} \sqrt{(\mathrm{y}_{m+1} - \mathrm{y}_m)^2 + (\mathrm{z}_{m+1} - \mathrm{z}_m)^2} \\ c &= \begin{cases} 1, & \text{path} \neq obs \\ 0, & \text{path} = obs \end{cases} \end{aligned} \tag{5.1g}$$

where f, k, and c denote the fitness value of particle, particle path distance, and collision test function respectively; z_{m} and y_{m} denote ordinate and abscissa of the node m. The ordinate and abscissa of node m's adjacent nodes are denoted by $\mathrm{z}_{\mathrm{m}+1}$ and $\mathrm{y}_{\mathrm{m}+1}$. The fact that the value of m equals one indicates that the node m is the launching node of the local route is indicated by this value. On the other hand, when m equals n, it signifies that m is the node that serves as the aim or target of the local path [69].

Step 11: The fitness values of the global ideal, denoted by 'gbest', and the individual ideal, denoted by 'pbest' should be compared to the fitness value determined for the opposite position [69]. In the event that the particle's fitness value is sufficiently high at the opposite position, it will be moved to its original location. If that not be the case, it will not be transferred.

Step 12: Find out if the maximum quantity of repetitions, $\mathrm{T}_{\max}$, has been reached; if it has not, return to the eleventh step and terminate the iteration while preserving the local optimization route.

Step 13: Check to see if the destination node is included in the iteration; if it is, the process should be terminated and the total path, which is the sum of the local pathways, should be reported. If not, proceed to step eight once more (Figures 8, 9 and 10).

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

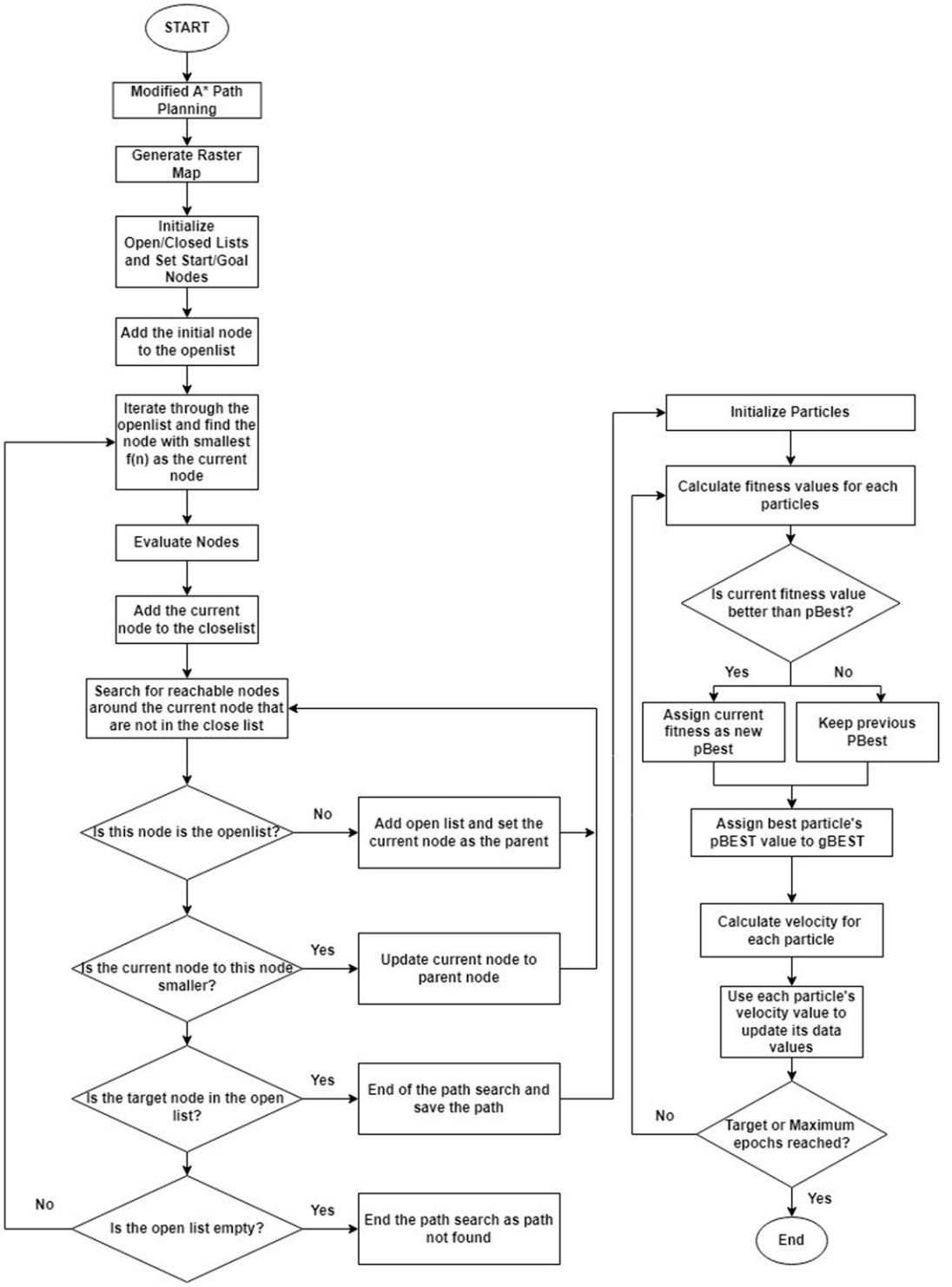


Figure 8. *Hybrid modified A* algorithm flow chart.*

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

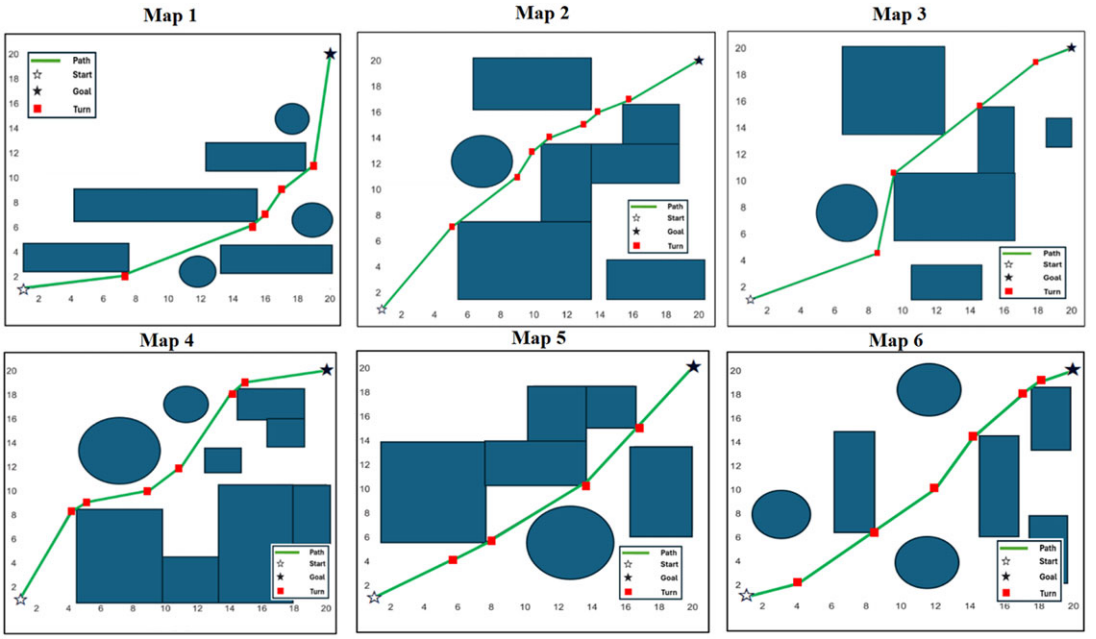


Figure 9. *Matlab simulation of HMA* algorithm in Map1 to Map 6.*

Algorithm Pseudo-code of HMA*

```
Input the information of start node, target node and environment map
Create grid map
Create a collection to store the search node
Set the minimum node of f(n) as the current node n
while The current node is not the target node do
    Search for eight-neighbor of current nodes
    if The eight-neighbor points are in openLest then
        Calculate the f(n) of these points and select the minimum as the current node
    else if The eight-neighbor points are not in openLest then
        Add them in openList
        Calculate the f(n) of these points and select the minimum as the current node
    end if
end while
Trace to find the initial path and place the path node in pathPoints
Delete non-start and non-destination nodes on local paths in pathPoints
Delete the intermediate node when obstacles exist between the separated nodes on pathPoints
while The target node n does not enter the iteration do
    n = n + 1
    Take n and n+2 isolated nodes in pathPoints as starting and destination nodes for local path optimization
    Initialize the PSO parameters
    while t ≤ maxIter do
        t = t + 1
        Calculate the objective function values F of each particle
        Update individual and group optimal values
        Calculate the objective function values F_o of each particle's opposite position
        if F_o ≤ F then
            Move the particle to its opposite position
        end if
        Update the velocity and position of each particle
    end while
    Output a global optimization path composed of local optimization paths
end while
Output nodes of the final route
```

Figure 10. *Generalized pseudo-code steps of hybrid modified A* algorithm.*

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

5.2. Problem modeling

Path planning must reduce AGV runtime. Most path planning algorithm research measures path length. AGV running time depends on path length, turning radius, path planning method, computation time, and path nodes. Therefore, these variables are in the following objective functions.

A. *Objective function for minimizing path length*

Path length directly impacts AGV's linear travel time; hence, the path design method aims to minimize it. Therefore, the goal function should prioritize path length that must be the shortest. The objective function of shortest length of the path is given as in ref. [69]:

$$\min D(\mathrm{y}, \mathrm{z}) = \sum_{i=1}^{n-1} \sqrt{(\mathrm{y}_{i+1} - \mathrm{y}_i)^2 + (\mathrm{z}_{i+1} - \mathrm{z}_i)^2} \tag{5.2a}$$

Here, n is the nodes in total number, while $\mathrm{y_i}$ and $\mathrm{z_i}$ are node i's abscissa and ordinate and $\mathrm{y_{i+1}}$ and $\mathrm{z_{i+1}}$ for node i + 1.

B. *Objective function for the shortest Path node*

The AGV must choose its next move at each node. Decision-making frequency increases with node count, limiting AGV motion fluency. The objective function should reduce path nodes to handle this. The formula for this function is given as in ref. [69]:

$$\textit{Min}\ \mathrm{P}\ (\mathrm{n}) = \mathrm{n}^{\mathrm{th}}\ \text{node in the path.} \tag{5.2b}$$

C. *Objective function for the shortest Turning amplitude*

The AGV's motion stability, path smoothness, and overall travel time are directly influenced by the amplitude of its turns. Smaller turning amplitudes lead to greater stability, smoother paths, and shorter travel durations. Consequently, the AGV's objective function should be designed to minimize the turning amplitude [69]. The corresponding formula is as follows:

$$\mathrm{minI}(\theta) = \sum_{i=1}^{n-2} \theta_n$$

$$\theta_n = \left| \arctan \left(\frac{\dfrac{\mathrm{z}_{n+1} - \mathrm{z}_n}{\mathrm{y}_{n+1} - \mathrm{y}_n} - \dfrac{\mathrm{z}_{n+2} - \mathrm{z}_{n+1}}{\mathrm{y}_{n+2} - \mathrm{y}_{n+1}}}{1 + \dfrac{\mathrm{z}_{n+1} - \mathrm{z}_n}{\mathrm{y}_{n+1} - \mathrm{y}_n} \times \dfrac{\mathrm{z}_{n+2} - \mathrm{z}_{n+1}}{\mathrm{y}_{n+2} - \mathrm{y}_{n+1}}} \right) \right| \tag{5.2c}$$

($\mathrm{n} = \mathrm{n}^{\mathrm{th}}$ node in the path.)

D. *Objective function for minimizing operating time*

Vehicle efficiency depends on AGV path planning algorithm execution time, which influences operational effectiveness. Thus, the objective function should minimize algorithm operation time. This objective function formula is given as in [71]:

$$\textit{Min}\mathrm{C(a)} = \mathrm{R_t} \tag{5.2d}$$

where $\mathrm{R_t}$ is the algorithm running time.

The goal-oriented functions are able to influence the execution time of the AGV in either a direct or indirect manner without causing any conflicts. We may integrate these objective functions into a

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

single, complete objective function that minimizes AGV running time [69, 70]. Presenting the combined objective function:

$$min t(n) = \frac{D(y, z)}{v_L} + P(n) \times t_o + \frac{I(\theta)}{v_s} + C(a) \tag{5.2e}$$

where v_L denotes the AGV's linear speed of motion. In this work, $v_L = 1$ m/s. When the AGV arrives at a node, t_o stands for the amount of time it needs to make a decision. This paper uses $t_o = 0.1$ s. The AGV's turning speed, or the amount of time it takes to turn one radian, is represented by v_s. This paper uses π rad/s for v_s.

6. Experiment of path planning simulation

To prove that the Hybrid Modified A* algorithm is a good way to plan routes for AGVs, experiment will utilize it. To do this, it will be necessary to evaluate the Hybrid Modified A* algorithm in conjunction with A*, PRM, RRT, and Bi-RRT. Researchers discovered that even when faced with substantial challenges depicted on the environment map, the hybrid intelligent optimization technique autonomously produced a path that maximized efficiency [71].

6.1. Description of the problem

In addition to its other functions, the intelligent warehouse also serves as the environment for path planning. Despite obstacles, the automated guided vehicle (AGV) continues to navigate towards completing its tasks [71]. When it comes to the process of planning a path, it is of the utmost importance to determine the smallest possible length of the path and to reduce the overall rotating radius [72]. In order to generate the environment map, the following assumptions were taken into consideration:

1. There are fixed, predefined areas wherein obstructions will be detected.
2. The AGV operates at a constant rate.
3. Because the environment map only appears in two dimensions, it is feasible to ignore the AGVs and barriers height information.

6.2. Composition of models

Raster, visible, and topological diagrams are frequently employed for environment modeling in path planning. This study builds an environment map model using a raster technique. The workspace where the AGV is tested is represented by a 20 x 20 decimeter (or 200 × 200 cm) two-dimensional raster map. The raster-based environment map models that were employed for assessment are shown in Figure 11. The percentage of environment maps with obstacles covered is 25% on map 1, 30% on map 2, 35% on map 3, 40% on map 4, 45% on map 5, and 20% on map 6. On these maps, physical barriers are shown in blue, and derivable areas are shown in white. For the AGV to operate correctly, the starting and target nodes should be (1,1) and (20,20), respectively. The most crucial element in determining the path planning algorithm's evaluation is path length. AGV runtime is influenced by path length, node count, turning amplitude, and algorithm planning time. Linear transit time is impacted by path planning, which shortens AGV paths [13]. Number of nodes that are located along the path affects AGV's decision-making time and motion fluency, which in turn affects how quickly it moves in the direction of its destination. Turn amplitude affects AGV smoothness, motion stability, and travel time; smoother, more stable, and faster motion times are achieved with smaller turning amplitudes (Figure 11).

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

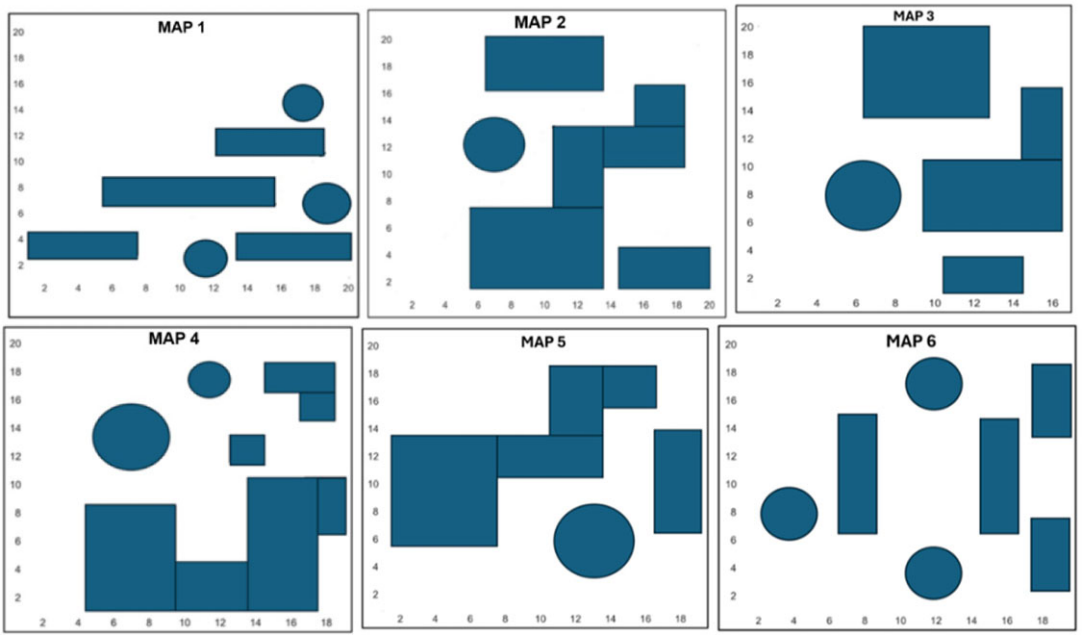


Figure 11. *Maps of the environment with obstacles (20 X 20 (in decimeters)).*

7. Simulation and real-time experiments

Setting up the conditions for obstacle environment simulations includes setting limits, goals, and rules. The following step is to select effective routing and obstacle avoidance algorithms. The chosen method is optimized for higher efficiency by a series of testing and data-gathering steps pertaining to path length, finish time, and rate of success [13, 73]. This systematic technique helps the design and evaluation of obstacle simulations by assisting robotics and navigation analysis. Mecanum wheel AGV navigation is tested in simulations to make sure algorithms and strategies work in actual situations. The following describes the various parts and process.

For the purpose of evaluating navigation algorithms, the virtual setting of an AGV is replicated in either two or three dimensions, including walls and barriers. The size, form, and position of sensors of an AGV must be incorporated into a precise model so as to offer realistic simulations that's particularly true for mecanum wheels. The evaluation and control of AGVs is accomplished through the modeling of navigating, routing, and avoiding obstacles algorithms in Matlab (R2023a) and ROS environment specially for dynamic obstacle avoidance [65]. The evaluation regarding the trajectory of the AGV, the amount of crashes, the length of the route, and the duration of execution are employed to determine performance. A comparison of algorithms provides data on their advantages, disadvantages, and ways of improvement. Improvements in the speed and navigational efficiency of automated guided vehicles (AGVs) may have achieved via recurring simulations, updated analysis-based programs, and enriched the situations.

Table III to Table VIII show the functional values of simulation and real-time experimental data from Map 1 to Map 6; comparison of the percentage deviation in the experimental path length and simulation path length is presented in Table IX; comparison of the percentage deviation in experimental motion time and simulation motion time is presented in Table X; comparison of the percentage difference or enhancement in experimental path length and experimental motion time between HMA* and other algorithms is presented in Table XII and Table XIII respectively (Figures 12–23).

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

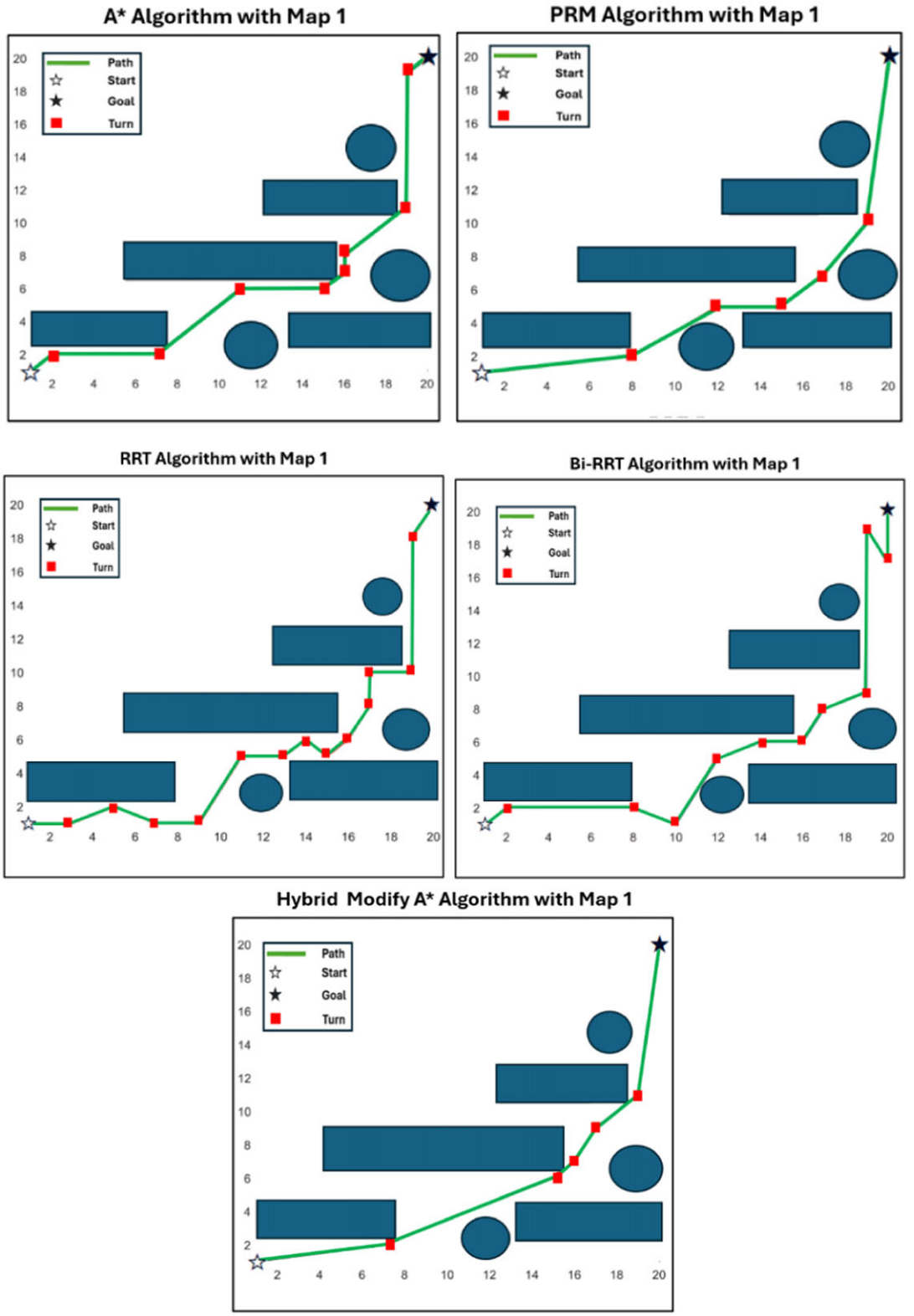


Figure 12. *Simulation analysis of the AGV's navigation in map 1.*

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

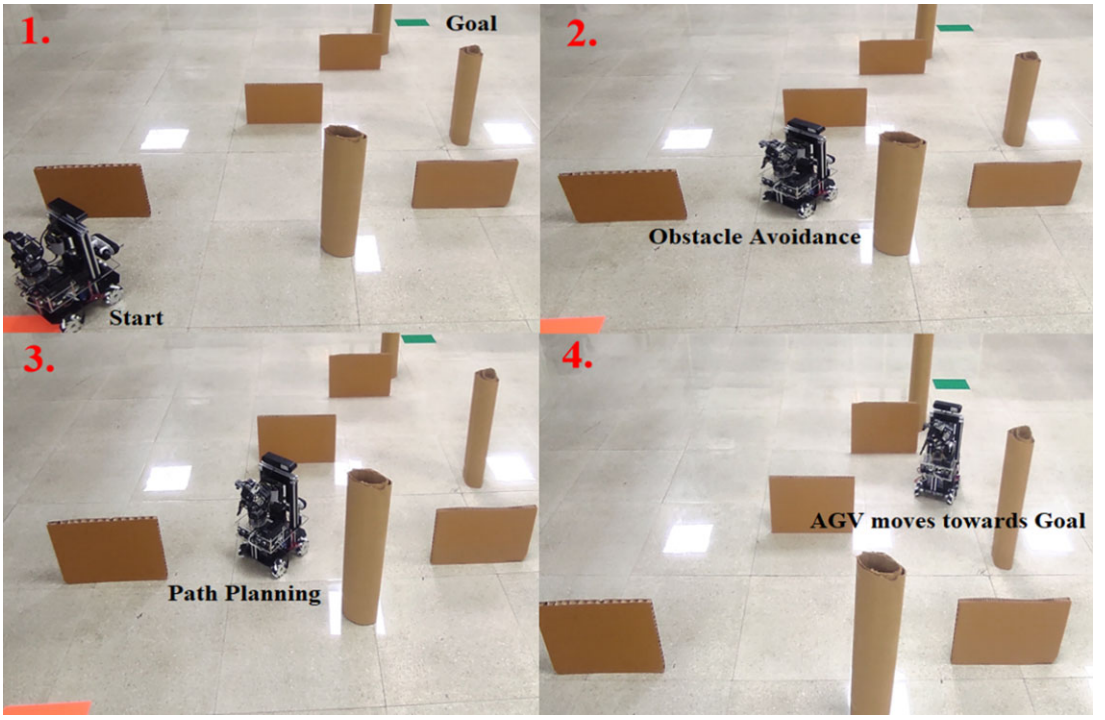


Figure 13. *An experimental analysis of the AGV's navigation in map 1.*

Table III. *The simulation and experimental functional values of each algorithm in map 1.*

Parameters for Map 1	**A***	**PRM**	**RRT**	**Bi-RRT**	**HMA***
Node No.	28.00	6.00	20.00	18.00	6.00
Turn amplitude (rad)	6.27	2.20	11.20	11.57	1.95
Operating time (s)	0.04	0.75	0.01	0.01	0.46
Simulation path length (cm)	307.76	301.66	292.88	289.87	272.97
Simulation motion time (s)	29.63	27.38	26.99	25.23	23.28
Experimental path length (cm)	323.14	321.54	309.91	304.33	288.56
Experimental motion time (s)	30.98	28.66	28.41	26.53	24.04

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

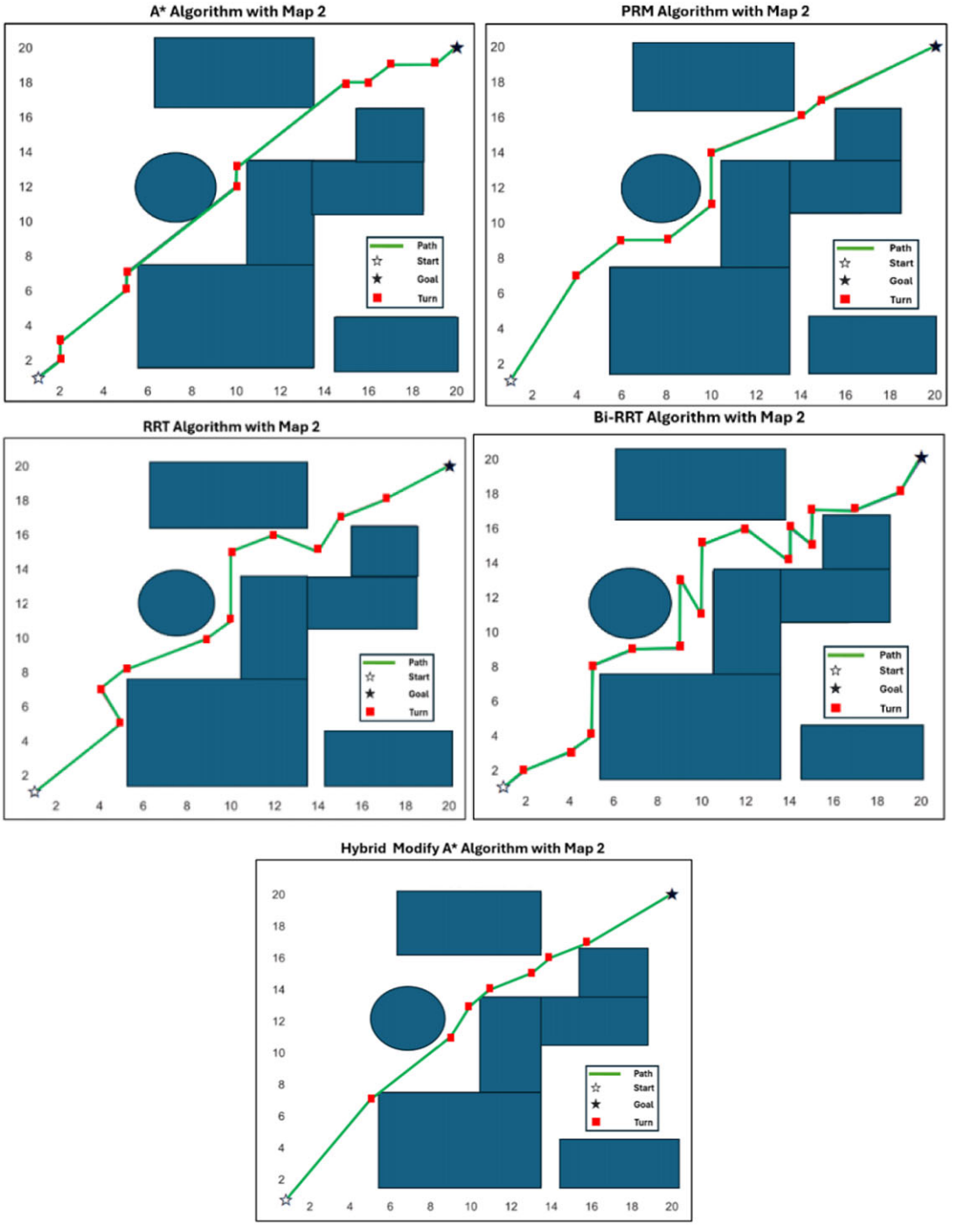


Figure 14. *Simulation analysis of the AGV's navigation in map 2.*

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

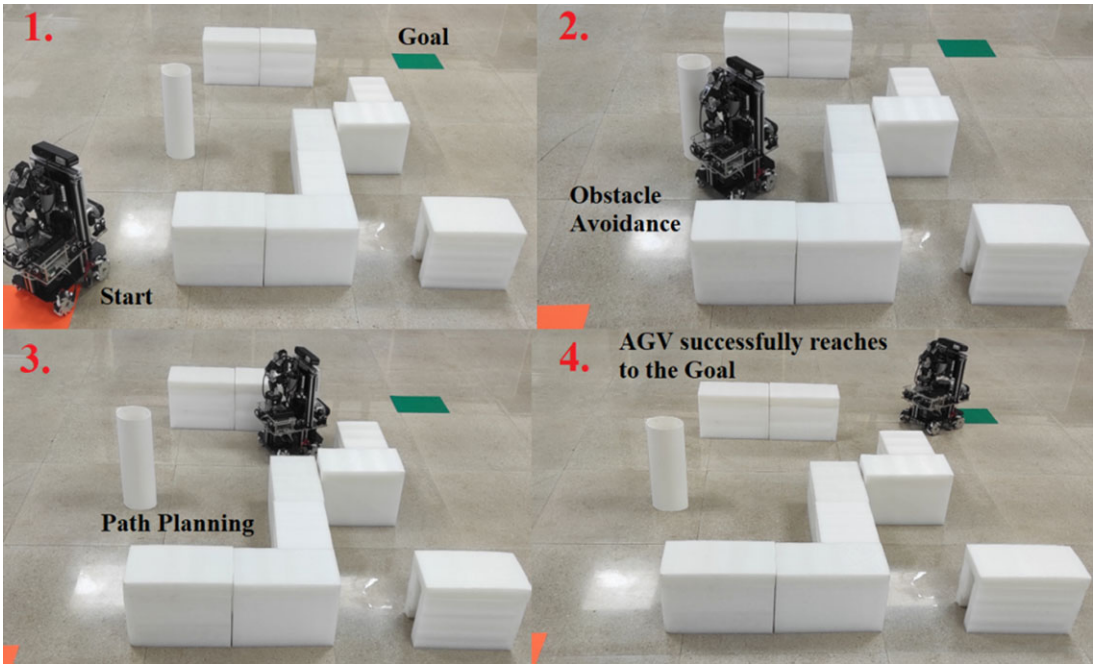


Figure 15. *An experimental analysis of the AGV's navigation in map 2.*

Table IV. *The simulation and experimental functional values of each algorithm in map 2.*

Parameters for Map 2	**A***	**PRM**	**RRT**	**Bi-RRT**	**HMA***
Node No.	22.00	9.00	17.00	17.00	9.00
Turn amplitude (rad)	6.18	3.51	10.63	8.56	2.17
Operating time (s)	0.04	0.74	0.01	0.01	0.24
Simulation path length (cm)	318.99	309.91	306.48	291.52	279.41
Simulation Motion time (s)	31.29	29.29	28.24	25.86	24.40
Experimental path length (cm)	333.53	326.05	319.53	305.32	290.45
Experimental motion time (s)	32.15	30.05	29.22	27.36	25.24

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

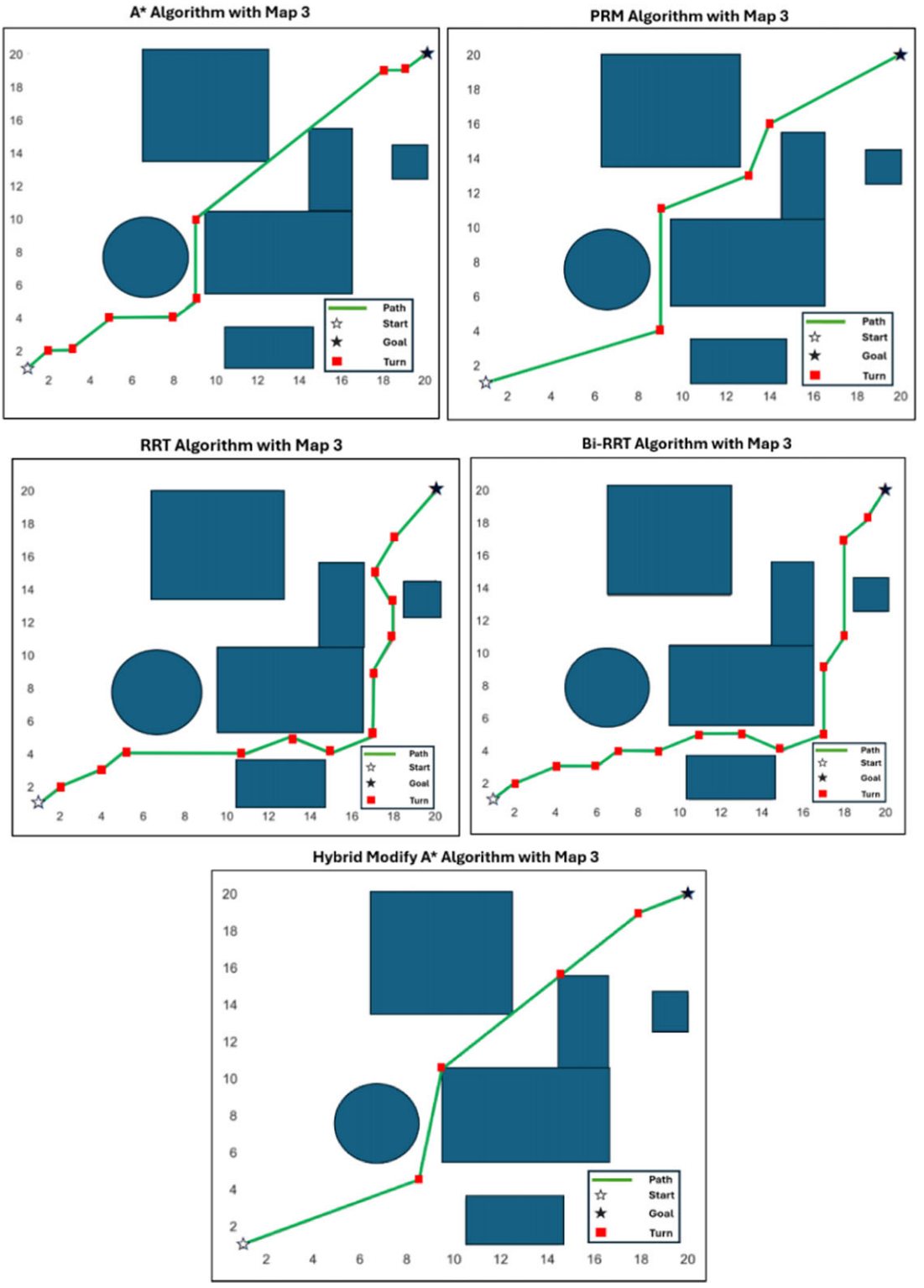


Figure 16. *Simulation analysis of the AGV's navigation in map 3.*

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

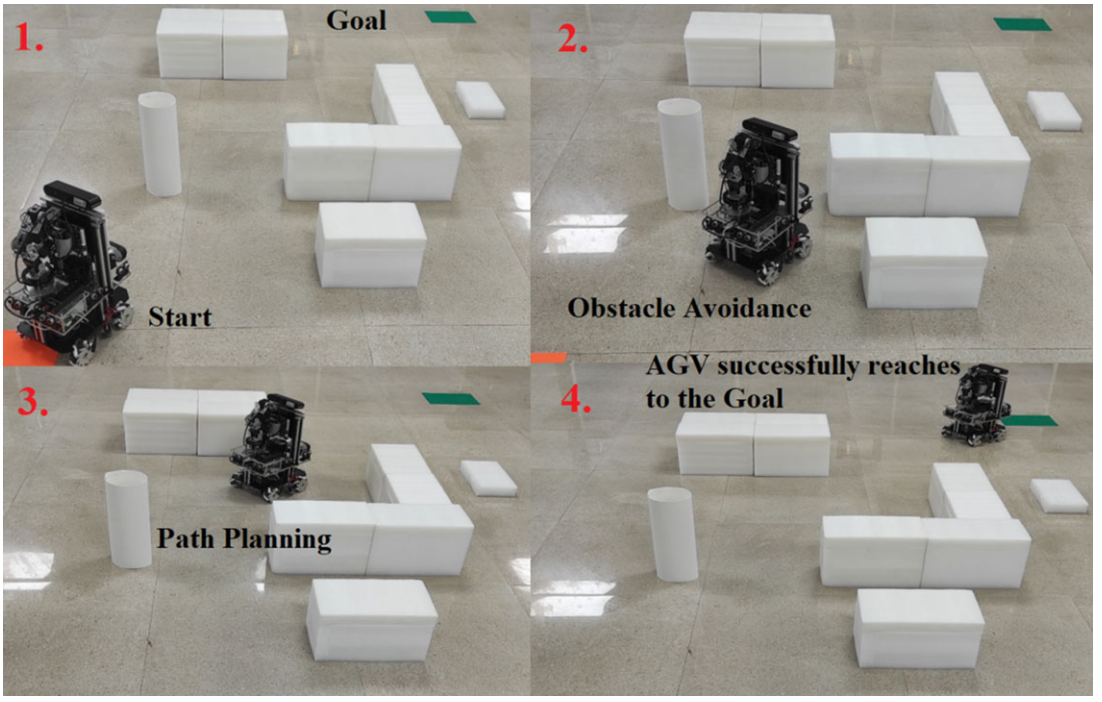


Figure 17. *An experimental analysis of the AGV's navigation in map 3.*

Table V. *The simulation and experimental functional values of each algorithm in map 3.*

Parameters for Map 3	**A***	**PRM**	**RRT**	**Bi-RRT**	**HMA***
Node No.	24.00	10.00	18.00	22.00	5.00
Turn amplitude (rad)	7.80	3.25	17.85	13.86	2.65
Operating time (s)	0.08	0.77	0.02	0.02	1.39
Simulation path Length (cm)	319.69	312.68	309.03	292.26	282.77
Simulation motion time (s)	31.82	29.51	28.67	26.88	25.16
Experimental path Length (cm)	335.8	327.14	320.48	308.73	294.98
Experimental motion time (s)	32.85	30.36	30.05	28.1	26.21

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

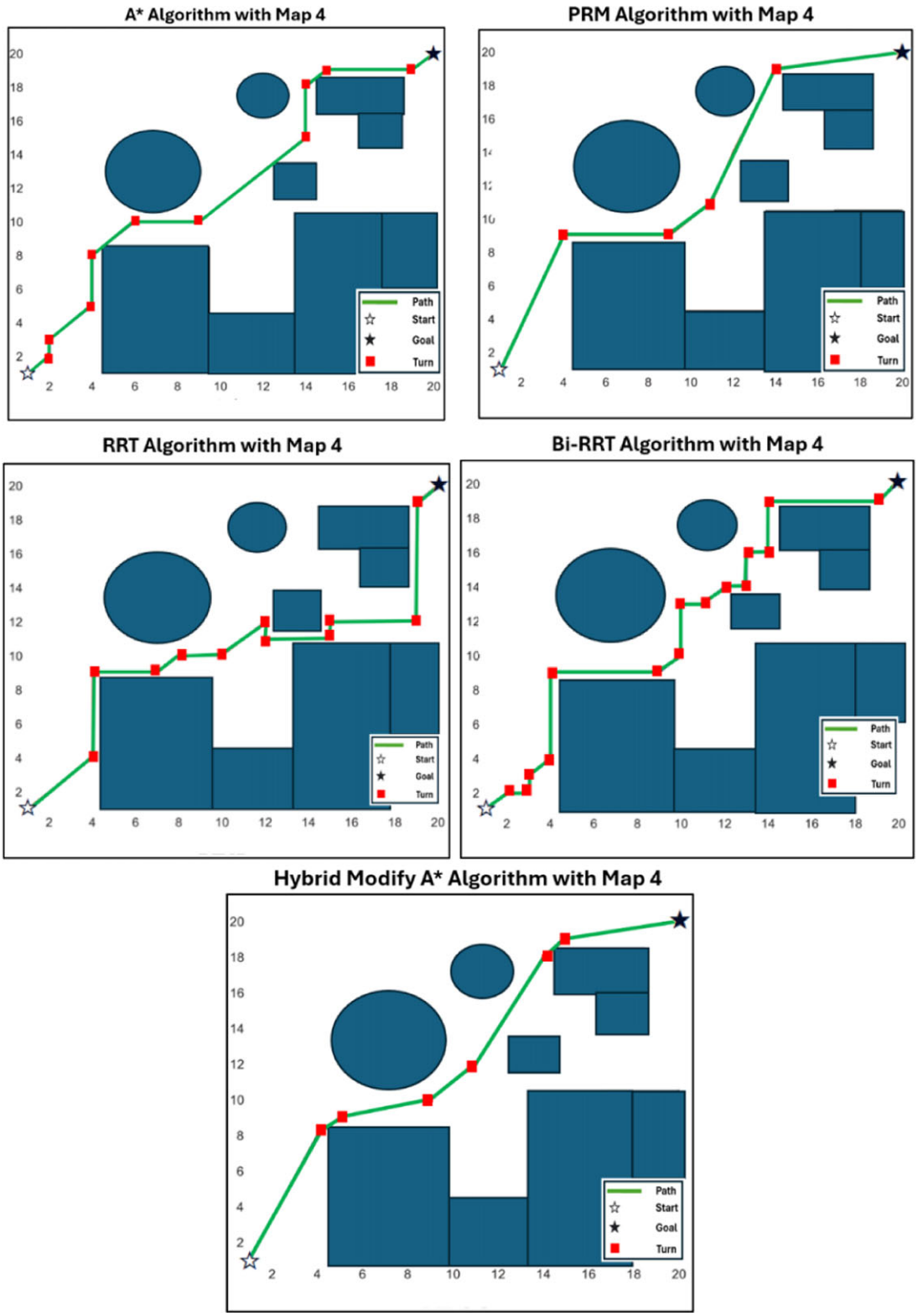


Figure 18. *Simulation analysis of the AGV's navigation in map 4.*

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

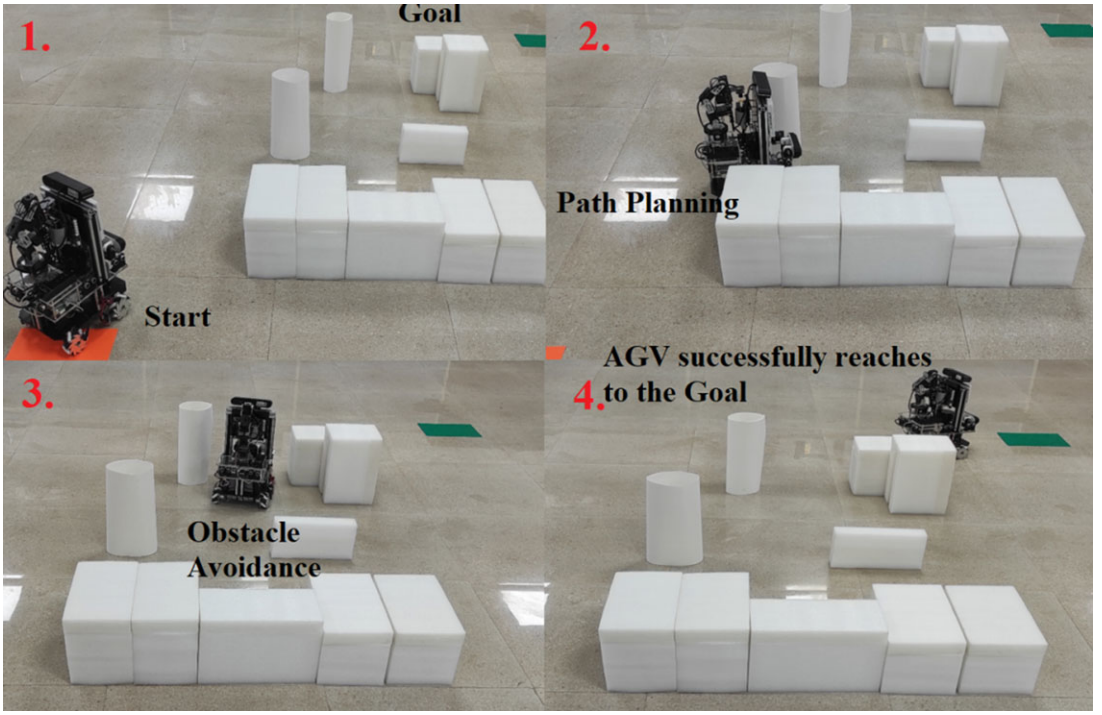


Figure 19. *An experimental analysis of the AGV's navigation in map 4.*

Table VI. *The simulation and experimental functional values of each algorithm in map 4.*

Parameters for Map 4	**A***	**PRM**	**RRT**	**Bi-RRT**	**HMA***
Node No.	26.00	6.00	35.00	33.00	7.00
Turn amplitude (rad)	7.84	3.25	15.15	14.43	0.34
Operating time (s)	0.04	0.82	0.01	0.02	0.61
Simulation path length (cm)	323.85	317.88	306.48	293.76	283.98
Simulation motion time (s)	32.04	29.63	28.44	28.66	26.24
Experimental path length (cm)	339.97	329.93	321.17	309.1	295.2
Experimental motion time (s)	33.94	31.13	30.07	29.64	27.02

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

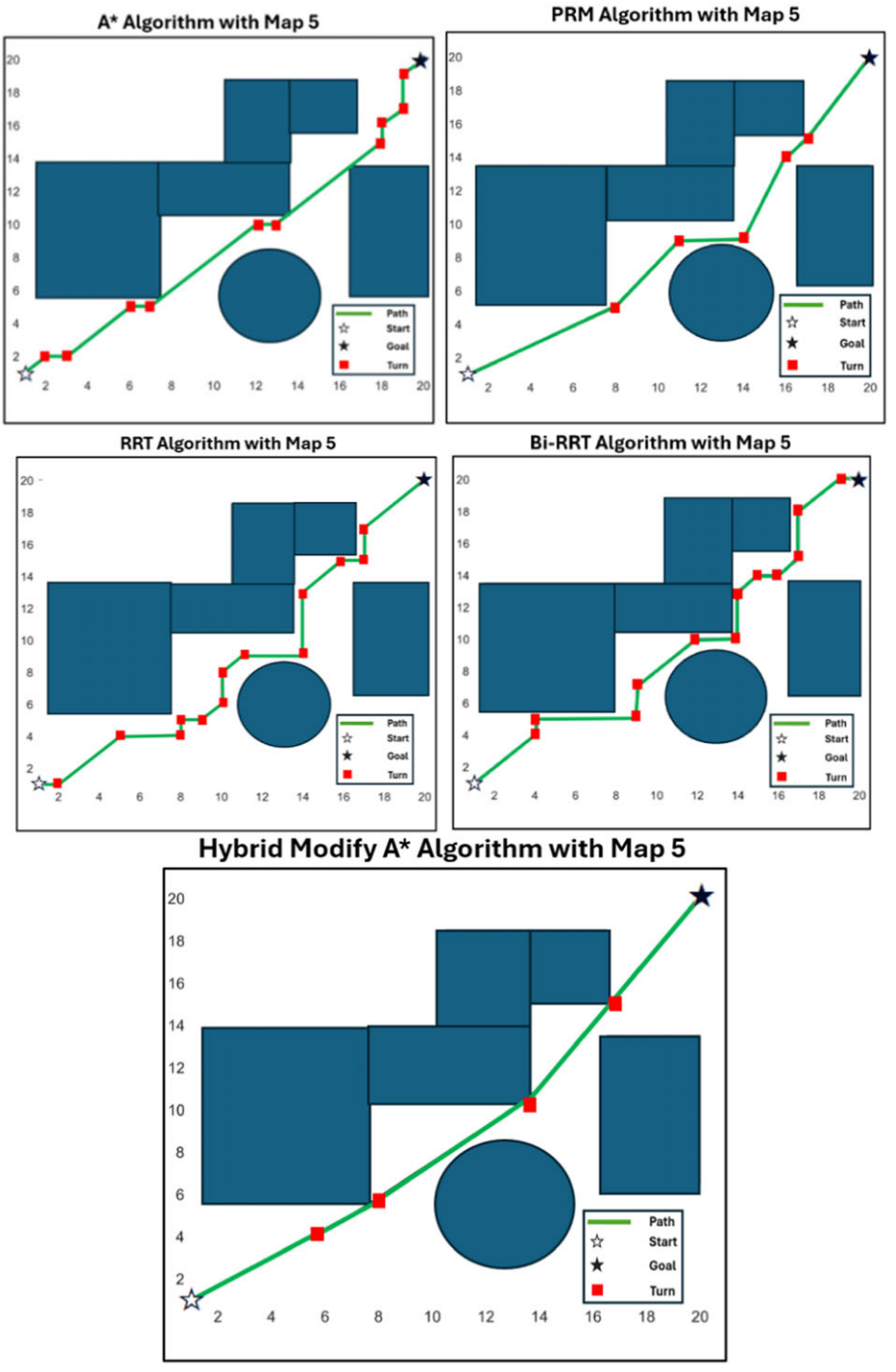


Figure 20. *Simulation analysis of the AGV's navigation in map 5.*

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

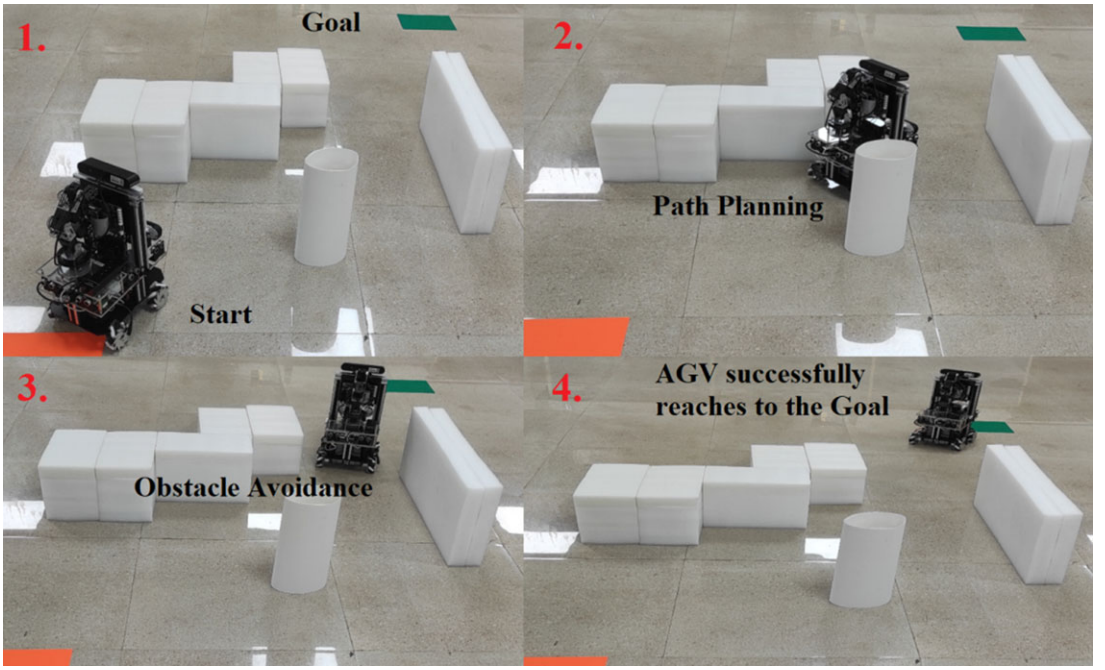


Figure 21. *An experimental analysis of the AGV's navigation in map 5.*

Table VII. *The simulation and experimental functional values of each algorithm in map 5.*

Parameters for Map 5	**A***	**PRM**	**RRT**	**Bi-RRT**	**HMA***
Node No.	22.00	7.00	28.00	28.00	4.00
Turn amplitude (rad)	6.27	4.05	16.32	13.68	1.33
Operating time (s)	0.06	0.80	0.01	0.02	0.29
Simulation path length (cm)	327.16	315.69	308.58	298.11	287.22
Simulation motion time (s)	33.82	31.47	30.86	29.93	27.41
Experimental path length (cm)	342.62	330.55	324.9	310.33	300.24
Experimental motion time (s)	34.78	32.45	31.55	30.76	28

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

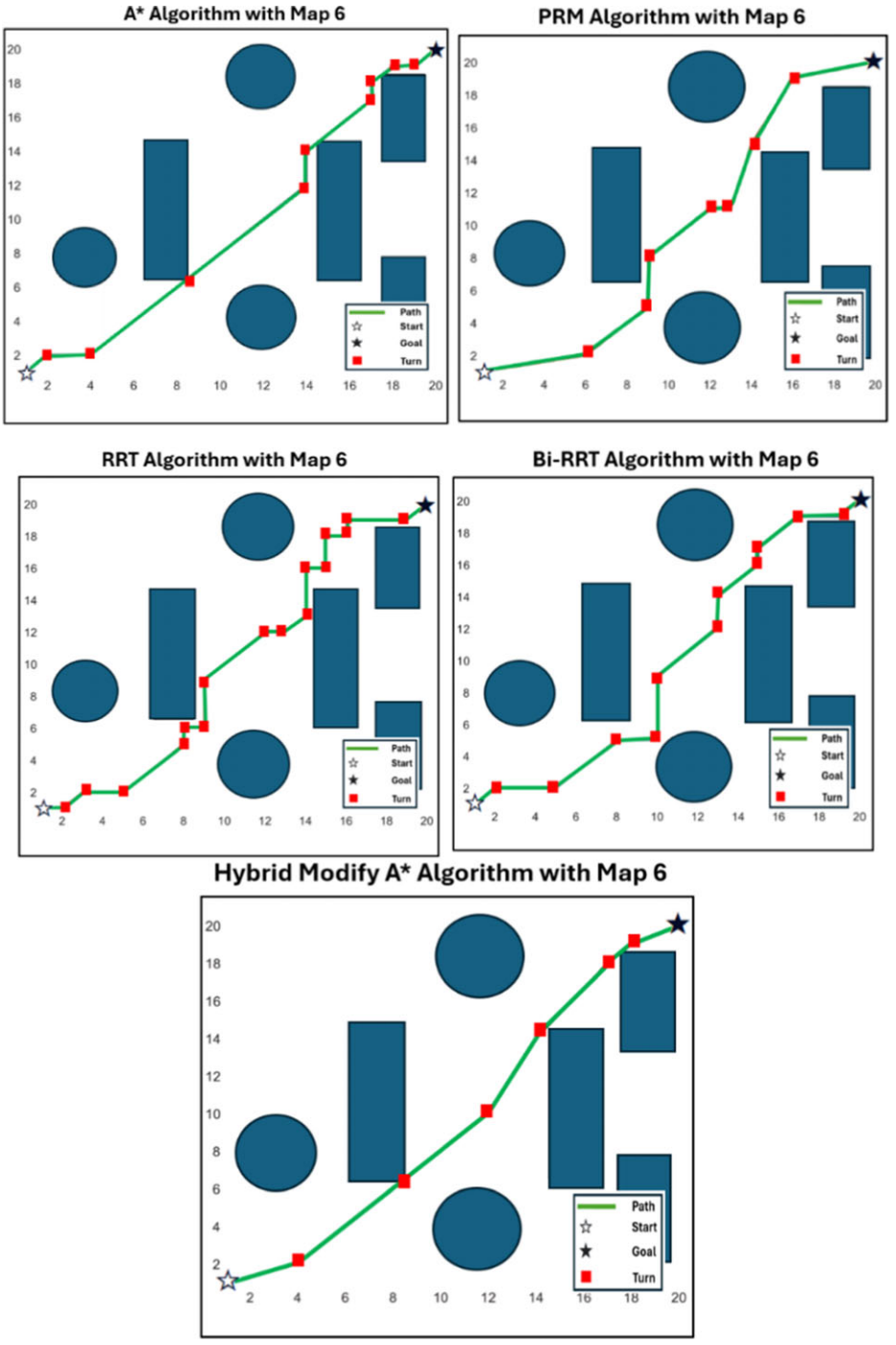


Figure 22. *Simulation analysis of the AGV's navigation in map 6.*

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

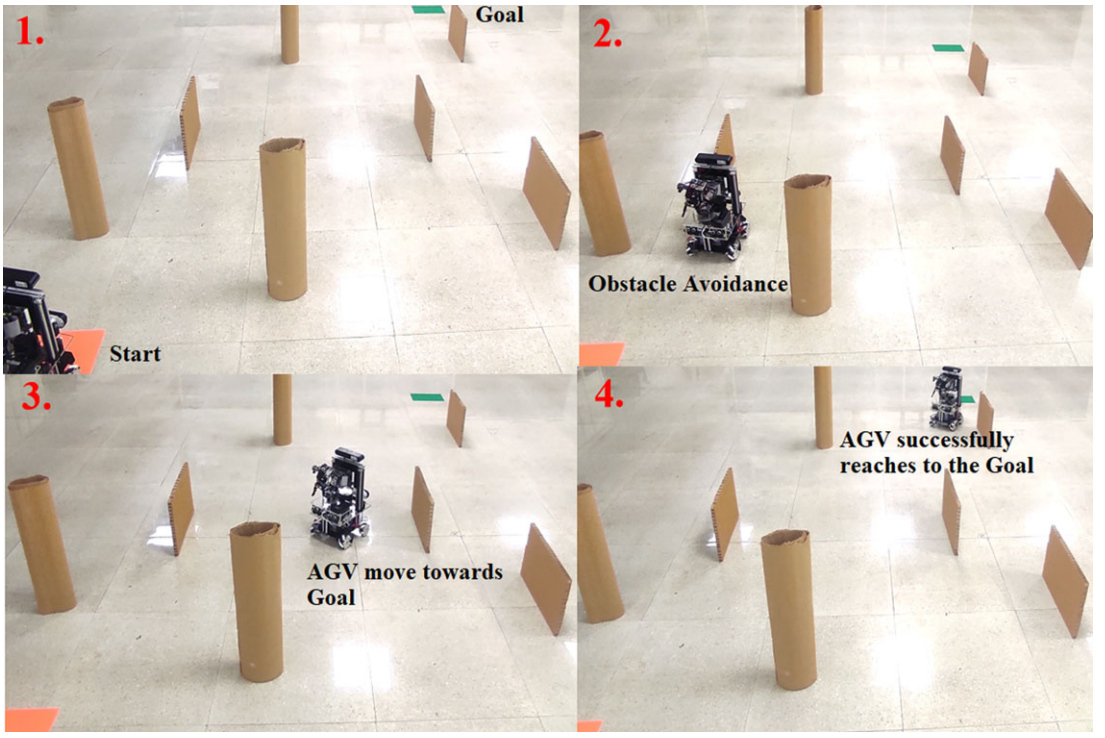


Figure 23. *An experimental analysis of the AGV's navigation in map 6.*

Table VIII. *The simulation and experimental functional values of each algorithm in map 6.*

Parameters for Map 6	**A***	**PRM**	**RRT**	**Bi-RRT**	**HMA***
Node No.	22.00	10.00	33.00	32.00	6.00
Turn amplitude (rad)	5.65	2.29	11.16	10.45	1.25
Operating time (s)	0.03	0.64	0.02	0.01	0.24
Simulation path length (cm)	308.07	310.08	297.10	291.25	275.62
Simulation motion time (s)	28.65	27.80	26.62	25.29	23.27
Experimental path length (cm)	321.63	320.56	308.2	303.6	285.6
Experimental motion time (s)	29.28	28.55	27.32	25.89	23.75

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

Table IX. *Comparison of the percentage deviation in experimental path length and simulation path length.*

	EPL: Experimental Path Length (in cm)						**SPL: Simulation Path Length (in cm)**			
	A*		**PRM**		**RRT**		**Bi-RRT**		**HMA***	
Environment	**EPL**	**SPL**	**EPL**	**SPL**	**EPL**	**SPL**	**EPL**	**SPL**	**EPL**	**SPL**
Map 1	323.14	307.76	321.54	301.66	309.91	292.88	304.33	289.87	288.56	272.97
% Deviation	**5.00**		**6.59**		**5.81**		**4.98**		**5.71**	
Map 2	333.53	318.99	326.05	309.91	319.53	306.48	305.32	291.52	290.45	279.41
% Deviation	**4.55**		**5.20**		**4.25**		**4.73**		**3.95**	
Map 3	335.80	319.69	327.14	312.68	320.48	309.03	308.73	292.26	294.98	282.77
% Deviation	**5.03**		**4.62**		**3.70**		**5.63**		**4.31**	
Map 4	339.97	323.85	329.93	317.88	321.17	306.48	309.10	293.76	295.20	283.98
% Deviation	**4.97**		**3.79**		**4.79**		**5.22**		**3.95**	
Map 5	342.62	327.16	330.55	315.69	324.90	308.58	310.33	298.11	300.24	287.22
% Deviation	**4.72**		**4.70**		**5.28**		**4.09**		**4.53**	
Map 6	321.63	308.07	320.56	310.08	308.20	297.10	303.60	291.25	285.60	275.62
% Deviation	**4.40**		**3.38**		**3.73**		**4.24**		**3.62**	
Average %Deviation	**4.77**		**4.71**		**4.59**		**4.81**		**4.34**	

Table X. *Comparison of the percentage deviation in experimental motion time and simulation motion time.*

	EMT: Experimental Motion Time (in Seconds)						**SMT: Simulation Motion Time (in Seconds)**			
	A*		**PRM**		**RRT**		**Bi-RRT**		**HMA***	
Environment	**EMT**	**SMT**	**EMT**	**SMT**	**EMT**	**SMT**	**EMT**	**SMT**	**EMT**	**SMT**
Map 1	30.98	29.63	28.66	27.38	28.41	26.99	26.53	25.23	24.04	23.28
% Deviation	**4.55**		**4.67**		**5.26**		**5.15**		**3.26**	
Map 2	32.15	31.29	30.05	29.29	29.22	28.24	27.36	25.86	25.24	24.40
% Deviation	**2.74**		**2.59**		**3.47**		**5.80**		**3.44**	
Map 3	32.85	31.82	30.36	29.51	30.05	28.67	28.10	26.88	26.21	25.16
% Deviation	**3.23**		**2.88**		**4.81**		**4.53**		**4.17**	
Map 4	33.94	32.04	31.13	29.63	30.07	28.44	29.64	28.66	27.02	26.24
% Deviation	**5.93**		**5.06**		**5.73**		**3.41**		**2.97**	
Map 5	34.78	33.82	32.45	31.47	31.55	30.86	30.76	29.93	28.00	27.41
% Deviation	**2.83**		**3.12**		**2.23**		**2.77**		**2.15**	
Map 6	29.28	28.65	28.55	27.80	27.32	26.62	25.89	25.29	23.75	23.27
% Deviation	**2.19**		**2.69**		**2.62**		**2.37**		**2.06**	
Average %Deviation	**3.57**		**3.50**		**4.02**		**4.00**		**3.00**	

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

Table XI. *Experiment-based path length and motion time of the algorithms in different maps.*

	PL: Path Length (in cm)						MT: Motion Time (in Seconds)			
	A*		PRM		RRT		Bi-RRT		HMA*	
Environment	**PL**	**MT**	**PL**	**MT**	**PL**	**MT**	**PL**	**MT**	**PL**	**MT**
Map 1	323.14	30.98	321.54	28.66	309.91	28.41	304.33	26.53	288.56	24.04
Map 2	333.53	32.15	326.05	30.05	319.53	29.22	305.32	27.36	290.45	25.38
Map 3	335.80	32.85	327.14	30.36	320.48	30.05	308.73	28.10	294.98	26.21
Map 4	339.97	33.94	329.93	31.13	321.17	30.07	309.10	29.64	295.20	27.02
Map 5	342.62	34.78	330.55	32.45	324.90	31.55	310.33	30.76	300.24	28.00
Map 6	321.63	29.28	320.56	28.55	308.20	27.32	303.60	25.89	285.60	23.75

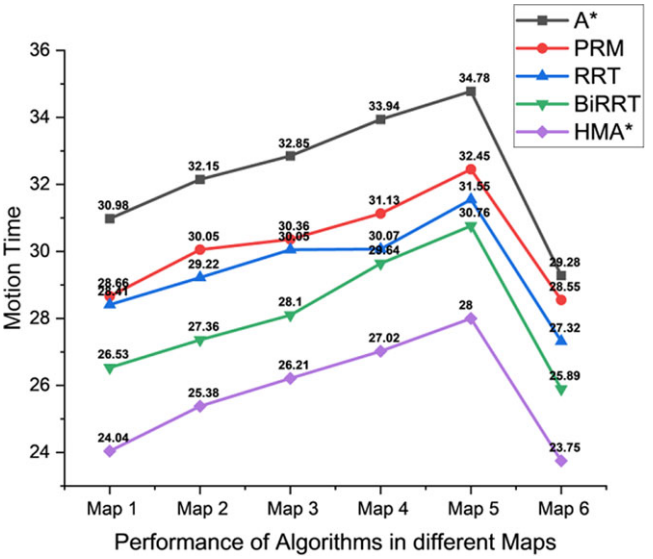


Figure 24. *Algorithm comparison according to motion time.*

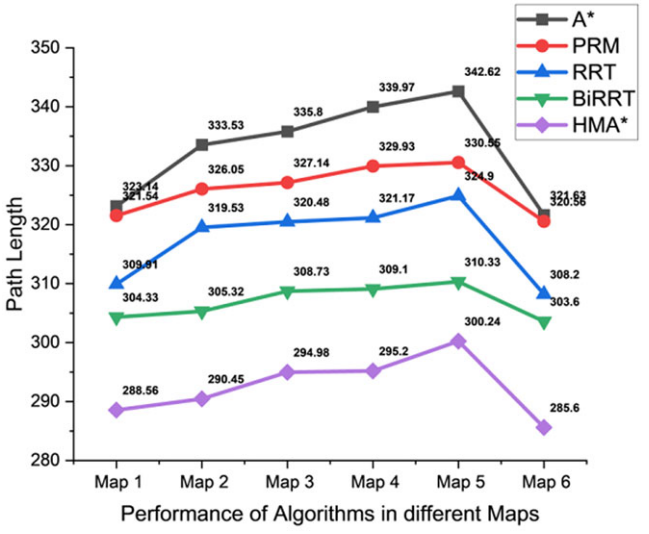


Figure 25. *Algorithm comparison according to path length.*

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

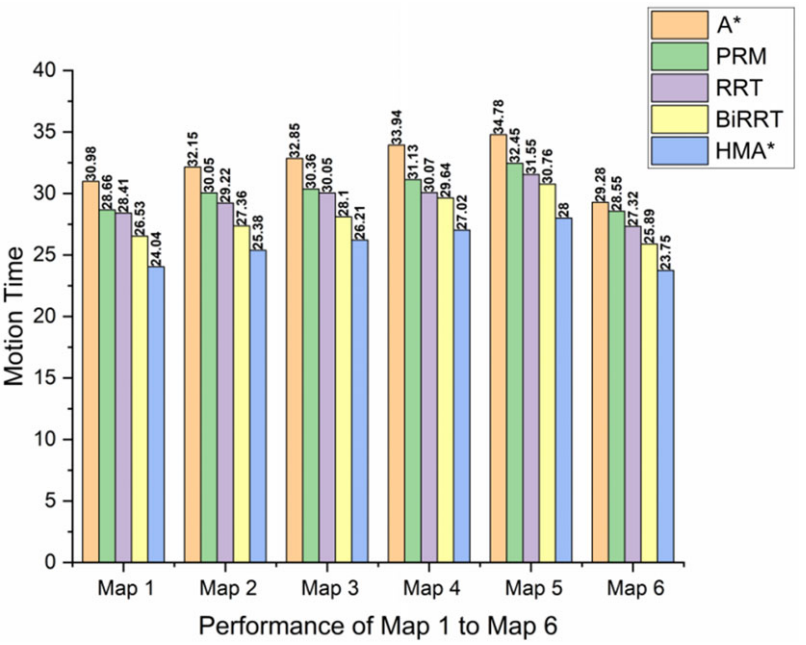


Figure 26. *Chart comparing different approaches based on motion time.*

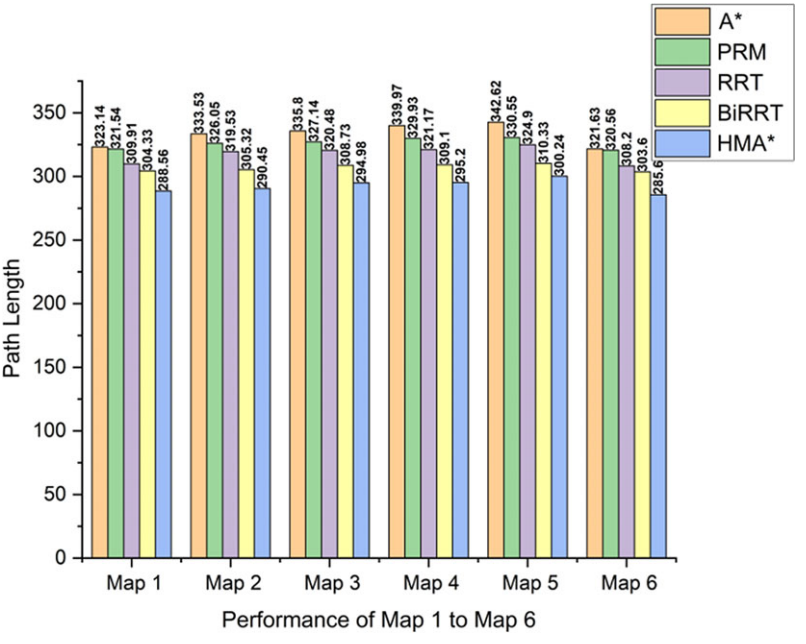


Figure 27. *Chart comparing different approaches based on path length.*

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

Table XII. *Comparison of the percentage difference in experimental path length between HMA* and other algorithms.*

	A*-HMA*	PRM-HMA*	RRT-HMA*	Bi-RRT-HMA*
Environment	**% Change in path length**	**% Change in path length**	**% Change in path length**	**% Change in path length**
Map 1	10.70	10.25	6.88	5.18
Map 2	12.91	10.91	9.10	4.87
Map 3	12.15	9.83	7.95	4.45
Map 4	13.16	10.52	8.08	4.49
Map 5	12.36	9.16	7.59	3.25
Map 6	11.20	10.90	7.33	5.92
Average	**12.08**	**10.26**	**7.82**	**4.69**

Table XIII. *Comparison of the percentage difference in experimental motion time between HMA* and other algorithms.*

	A*-HMA*	PRM-HMA*	RRT-HMA*	Bi-RRT-HMA*
Environment	**% Change in motion time**	**% Change in motion time**	**% Change in motion time**	**% Change in motion time**
Map 1	22.40	16.12	15.38	9.38
Map 2	21.05	15.54	13.14	7.23
Map 3	20.21	13.66	12.77	6.72
Map 4	29.22	13.20	10.14	8.83
Map 5	19.49	13.71	11.25	8.97
Map 6	18.88	16.81	13.06	8.26
Average	**21.88**	**14.84**	**12.62**	**8.23**

8. Experimental validation

Mecanum wheels allow omnidirectional movement in automated guided vehicles (AGVs) by using inclined rollers to enable movement in any direction without rotation. This capability, unlike traditional wheels that require steering, allows AGVs to move sideways, diagonally, or rotate in place, offering exceptional maneuverability in tight spaces. To fully leverage this, advanced algorithms for path planning, navigation, localization, and collision avoidance are essential. Path planning must account for dynamic limits such as maximum speed and acceleration, while navigation and control algorithms must optimize trajectory planning for smooth and precise movement. Localization algorithms need to incorporate the AGV's motion model and sensors, like odometry and wheel encoders, for accurate positioning. Collision avoidance algorithms must ensure safe navigation by considering the AGV's unique movement capabilities (Figures 24, 25, 26, 27 and 28).

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

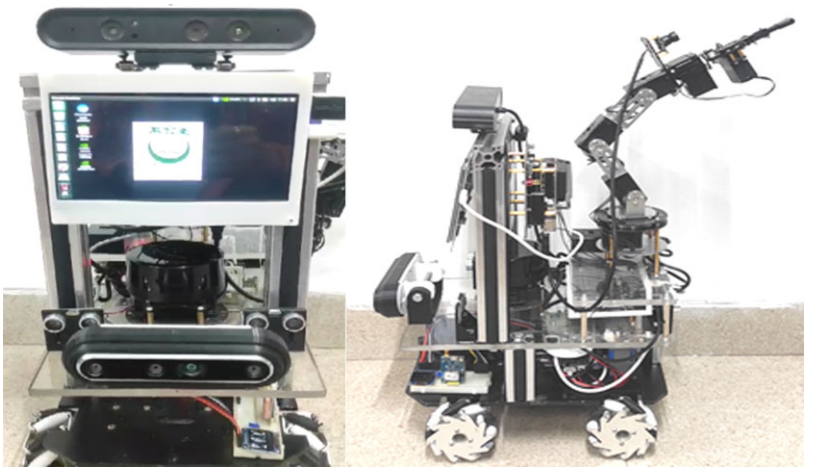


Figure 28. *AGV design and implementation using laser-cut mecanum wheels with robotic arm.*

Table XIV. *Overview of hardware and software.*

Parameters	**Technical Specifications**
Dimensions of AGV	300*200*450 mm
Type of motor	Permanent Magnet brushed motor
AGV weight	7 KG
AGV Maximum payload	11 KG
AGV Maximum speed	2.5 m/s
Working environment	Indoor and outdoor
Sensing and feedback	RP Lidar, IMU, Depth Camera
Slam technique	Hector, RTAB-Map, Gmapping, and Cartographer
Algorithms	A*, MA*, PSO, PRM, RRT, Bi-RRT, HMA*
Navigation and drive	ROS navigation Stack, Web-based GUI, Hand Gesture
Simulation tool	ROS,Gazebo, Rviz, Matlab
Actuation	Nvidia Jetson Nano, and Arduino Mega
Operating system	ROS Melodic in Ubuntu 18.04 or Neotic in Ubuntu 20.04
Encoder parameters	500 line AB phase high precision photoelectric encoder
Rated current of motors	360mA
Rated torque	1.0 kgf.cm
Rated motor voltage	12V
Rated motor power	4.32 W
Drive structure	Mecanum Wheel with pendulum suspension
Depth camera	Asrta Pro, Intel Real Sense
IMU sensor	MPU 9250
USB Interface	4 X USB 3.0
Network interface	Gigabit Ethernet/ M.2 Key E

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

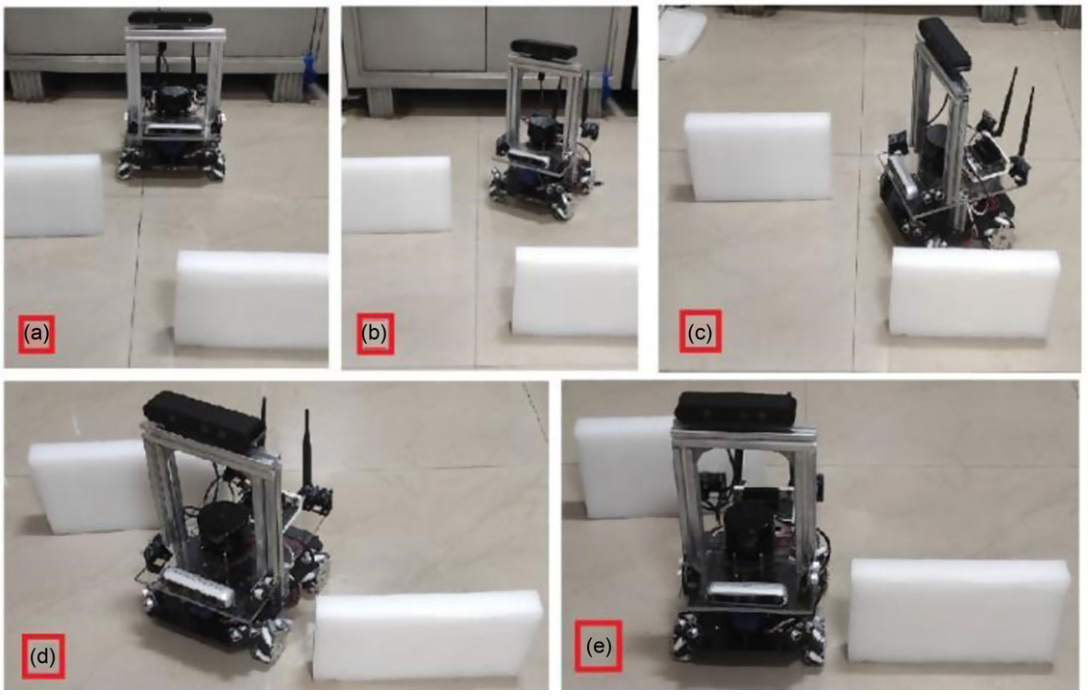


Figure 29. *The avoidance of obstacles in real-time using HMA* algorithm.*

The Hybrid Modified A* (HMA*) approach-based AGV control in random environment, as shown in Figure 29, combines the optimization powers of PSO with the heuristic search efficiency of MA*. In this way, HMA* is able to analyze intricate settings. The AGV is equipped with two lidar sensors to detect obstacles and measure distance accurately, as well as two vision sensors to improve its ability to perceive its surroundings and identify objects. The CPU is a Jetson Nano, that performs complex calculations for the Hybrid Modified A* method, sensor data fusion, as well as making decisions. The mecanum wheels travel effortlessly and precisely because of an STM32 motor controller, which provides precise motor control. To further improve avoiding obstacles and safety in difficult circumstances, six ultrasonic sensors are set all around the AGV to provide additional detection of proximity abilities. For the framework to work, the connections among the sensors and the Jetson Nano controller are crucial. The interfaces between the lidar sensors and the Jetson Nano enable real-time distance measurement processing and high-speed information transmission. The vision sensors are connected by CSI (Camera Serial Interface) ports, which enable the Jetson Nano to receive high-definition video feeds for object detection and environmental mapping. The Jetson Nano's GPIO (General-Purpose Input/Output) links can be used to connect the ultrasonic sensors to transmit proximity data, that is needed to detect obstructions in close proximity. Precise motor controls and feedback have been rendered feasible by the UART (Universal Asynchronous Receiver-Transmitter) connection that connects the STM32 motor control module to the Jetson Nano. With this sophisticated configuration, which guarantees precise navigation, localization, and collision avoidance in a variety of experimental circumstances, the AGV's omnidirectional capabilities may be fully utilized [74]. The main block diagram of the hardware arrangement is as follows (Figure 30):

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

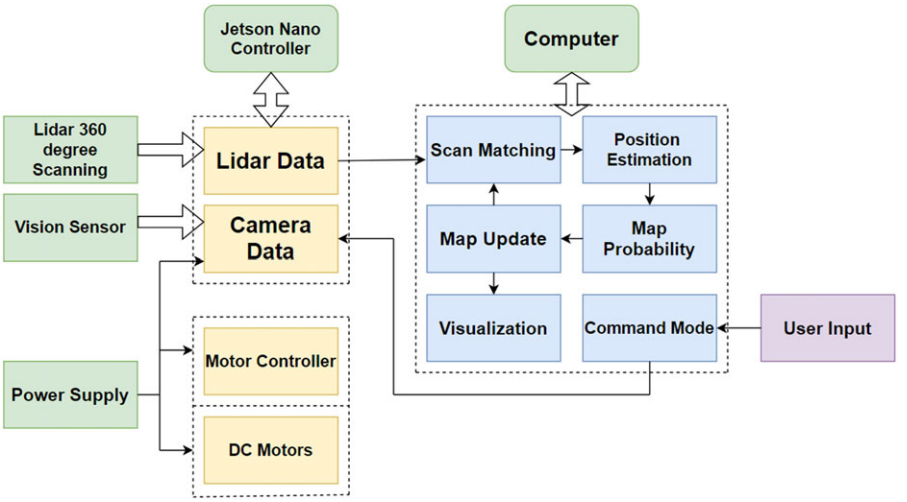


Figure 30. *AGV's overall control structure.*

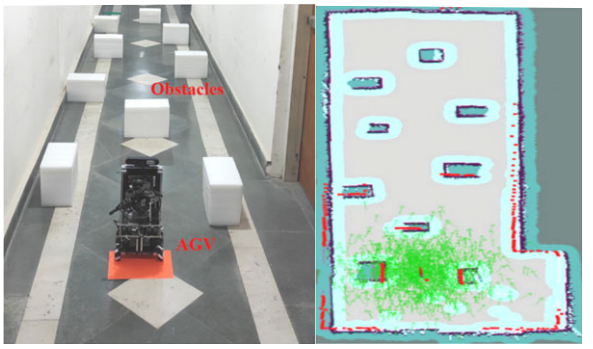
A thorough description of the hardware setup of the AGV to be found in Table XIV. After implementing the HMA* algorithm into the AGV to evaluate its effectiveness the real-time experiment is performed with the following procedures:

1. Create a map illustrating the surroundings. Figure 31(a) shows both the experimental area and the map constructed on the ROS platform.
2. Identify a desired initial position and target location on the map.

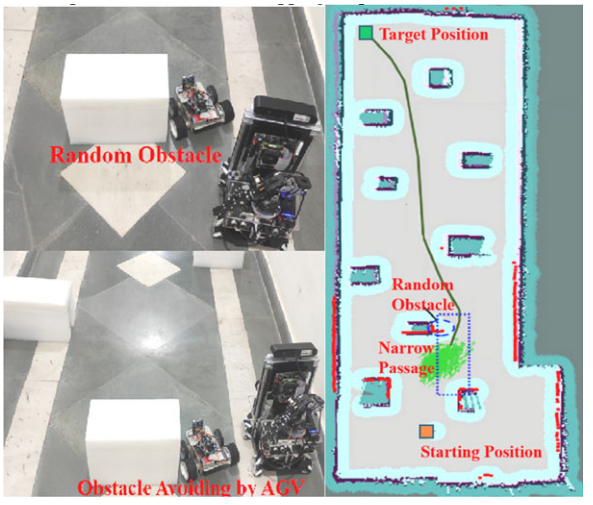
Place arbitrary obstacles along the planned path of the AGV and observe whether the AGV can successfully navigate these random obstacles. This procedure verifies that the HMA* algorithm works as intended when faced with unforeseen obstacles.

The AGV begins global path planning for traveling from its current initial position to the target location, as shown in Figure 31(b). It travels along the route that has been planned globally, moving through tight spaces with ease. The AGV uses sensor data to determine the first random barrier, avoids it, and then re-enters the global path to keep moving in the direction of the goal. As seen in Figure 31(c), the AGV uses sensors to detect and maneuver through new obstacles as it moves, finally arriving at the designated target location. The navigational area is 500 cm by 150 cm in size. Due to the HMA* algorithm, the automated guided vehicle (AGV) successfully traversed a 548 cm course in 55 s, avoiding every obstacle and proving its effectiveness in a matter of seconds. The experimental findings show that when the Modified A* and PSO (HMA*) path optimization methods are combined, the AGV can travel through restricted areas effectively. The AGV is able to avoid random obstacles and plan globally the best paths because of this method. The relationship between linear and angular velocities and different obstacle configurations is also investigated in this study. The surroundings with one obstacle are depicted in Figure 32(a), which shows the changes in angular and linear velocities as the AGV approach and avoids the obstacle. The two-obstacle environment shown in Figure 32(b) illustrates how velocities change when the AGV maneuvers around both obstacles. Lastly, an environment with three obstacles is shown in Figure 32(c), where the graphs illustrate the greatest changes in the AGV's velocities as it navigates around the obstacles.

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press



(a) Experimental Site and Gmapping Map Construction for AGV



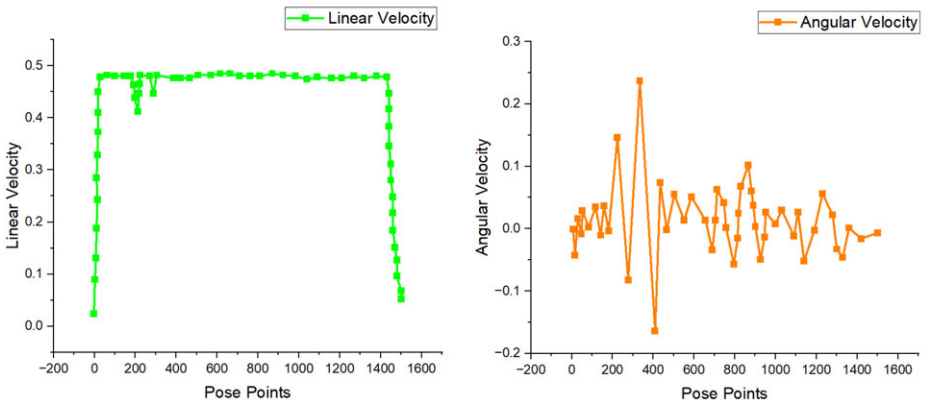
(b) Obstacle avoidance for the random obstacle



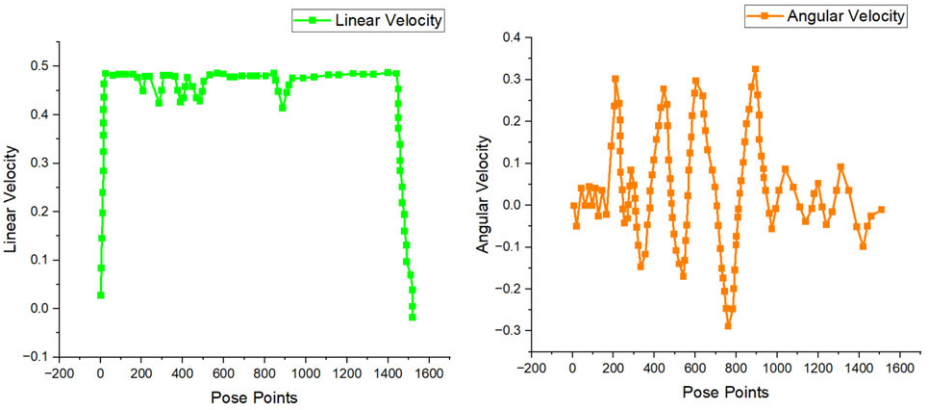
(c) AGV move to target location

Figure 31. *Experimental results with sensor fusion for HMA* validation.*

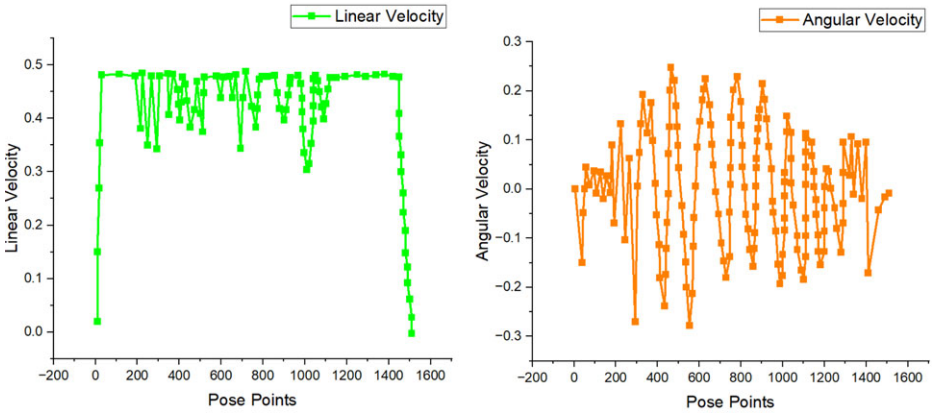
https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press



(a) The surroundings involving one obstacle.



(b) The surroundings involving two obstacles.



(c) The surroundings involving three obstacles.

Figure 32. *Performance comparison of velocity in different environment.*

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

9. Conclusion

This study shows how a Hybrid Modified A* (HMA*) method can find AGV's collision-free shortest path quickly. As seen in Figures 12–23, the different algorithms adopt different paths to reach their goals. In each map, various algorithms are applied to determine the optimal path. The HMA* algorithm consistently produces smoother pathways across the six maps because the routes it selects are generally shorter, have fewer path nodes, and involve fewer turns. This efficiency is primarily due to the algorithm's ability to identify more direct routes. The tables accompanying the maps offer a clear comparison of the algorithm's success in achieving efficient path planning. The HMA* algorithm can avoid static and random obstacles for AGVs as shown in Figure 31. It also adjusts local pathways and provides collision-free navigation in complex settings. The hybrid algorithm can avoid random obstacles, alter local pathways, and navigates in complex environments without colliding and quickly reaching the objective.

- The HMA* algorithm outperforms A*, PRM, RRT, and Bi-RRT in average path length by 12.08%, 10.26%, 7.82%, and 4.69%, respectively, across six maps. Additionally, HMA* demonstrates superior performance in average motion time, achieving improvements of 21.88%, 14.84%, 12.62%, and 8.23% over the same algorithms as shown in Table XII and Table XIII.
- The HMA* algorithm shows promising results with an average deviation of 4.34% in path length and 3% in motion time between simulation and experiments as shown in Table IX and Table X, demonstrating a close approximation to real-world conditions.
- The HMA* method effectively avoids collisions in situations with static and random barriers (Figure 31). Its dynamic local path adjustment ensures reliable navigation in complicated and uncertain environments. Real-world AGVs must be adaptable to quickly changing environmental conditions.
- The HMA* algorithm's collision-free and fast navigation under various conditions makes it practical and versatile for industrial and logistical applications.
- Further research may examine to improve the HMA* algorithm's ability to recognize and avoid obstacles in real-time at extremely dynamic situations by integration of sophisticated sensor fusion methodologies.

Anyone can download the codes and videos created for this purpose by visiting this link: https://github.com/ankurgsb21/Hybrid-Modified-A-star-Algorithm-Modified-A-star-PSO-

Author Contributions. Under the guidance of Prof. (Dr.) Mohd. Suhaib and Prof. (Dr.) Ajay K.S. Singholi, Ankur Bhargava was responsible for the study's conception, design, data collection, methodology, visualization, and investigation. Prof. Mohd. Suhaib and Prof. Ajay K. S. Singholi provide assistance with writing, reviewing, editing, and statistical analysis.

Financial Support. This research received no specific grant from any funding agency, commercial or not-for-profit sectors.

Competing interests. The authors declare no competing interests exist.

Ethical Approval. Not applicable.

References

[1] Z. Yao, W. Zhang, Y. Shi, M. Li, Z. Liang, F. Li and Q. Huang, "RimJump: Edge-based shortest path planning for a 2D map," *Robotica* **37**(4), 641–655 (2019). doi: 10.1017/S0263574718001236.

[2] M. H. Tanveer, C. T. Recchiuto and A. Sgorbissa, "Analysis of path following and obstacle avoidance for multiple wheeled robots in a shared workspace," *Robotica* **37**(1), 80–108 (2019). doi: 10.1017/S0263574718000875.

[3] M. Sharma and H. K. Voruganti, "Multi-objective optimization approach for coverage path planning of mobile robot," *Robotica* **42**(7), 1–25 (2024). doi: 10.1017/S0263574724000377.

[4] H. Khan, S. Khatoon, P. Gaur, M. Abbas, C. A. Saleel and S. A. Khan, "Speed control of wheeled mobile robot by nature-inspired social spider algorithm-based PID controller," *Processes* **11**(4), 1202 (2023). doi: 10.3390/pr11041202.

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

[5] O. A. R. A. Wahhab and S. A., "Al-Araji "Path planning and control strategy design for mobile robot based on hybrid swarm optimization algorithm," *Int J Intell Eng Syst* **14**(3), 565–579 (2021). doi: 10.22266/ijies2021.0630.48.
[6] S. Liu, S. Liu and H. Xiao, "Improved gray wolf optimization algorithm integrating A* algorithm for path planning of mobile charging robots," *Robotica* **42**(2), 536–559 (2024). doi: 10.1017/S0263574723001625.
[7] Z. E. Kanoon, A. S. Al-Araji and M. N. Abdullah, "An intelligent path planning algorithm and control strategy design for multi-mobile robots based on a modified elman recurrent neural network," *Int J Intell Eng Syst* **15**(5), 400–415 (2022). doi: 10.2266/ijies2022.1031.35.
[8] M. A. El Aziz, A. A. Ewees and A. E. Hassanien, "Hybrid Swarms Optimization Based Image Segmentation," **In:** *Hybrid Soft Computing for Image Segmentation*, (S. Bhattacharyya, P. Dutta, S. De and G. Klepaceds.) (Springer, Cham, 2016). doi: 10.1007/978-3-319-47223-2_1.
[9] A. S. Al-Araji, "Development of kinematic path-tracking controller design for real mobile robot via back-stepping slice genetic robust algorithm technique," *Arab J Sci Eng* **39**(12), 8825–8835 (2014). doi: 10.1007/s13369-014-1461-4.
[10] A. A. A. Rasheed, A. S. Al-Araji and M. N. Abdullah, "Static and dynamic path planning algorithms design for a wheeled mobile robot based on a hybrid technique," *Int J Intell Eng Syst* **15**(4), 167–181 (2022). doi: 10.22266/ijies2022.0831.16.
[11] C. Kim, J. Suh and J.-H. Han, "Development of a hybrid path planning algorithm and a bio-inspired control for an omni-wheel mobile robot," *Sensors* **20**(15), 4258 (2020). doi: 10.3390/s20154258.
[12] J. F. S. and S. R., "Self-adaptive learning particle swarm optimization-based path planning of mobile robot using 2D Lidar environment," *Robotica* **42**(4), 977–1000 (2024). doi: 10.1017/S0263574723001819.
[13] T. D. Tolossa, M. Gunasekaran, K. Halder, H. K. Verma, S. S. Parswal, N. Jorwal, F. O. Maria Joseph and Y. V. Hote, "Trajectory tracking control of a mobile robot using fuzzy logic controller with optimal parameters," *Robotica* 1–24 (2024). doi: 10.1017/S0263574724001140.
[14] C. Yuan, Y. Chang, Y. Song, S. Lin and F. Jing, "Design and analysis of a negative pressure wall-climbing robot with an omnidirectional characteristic for cylindrical wall," *Robotica* **42**(7), 2226–2242 (2024). doi: 10.1017/S0263574724000493.
[15] N. Nfaileh, K. Alipour, B. Tarvirdizadeh and A. Hadi, "Formation control of multiple wheeled mobile robots based on model predictive control," *Robotica* **40**(9), 3178–3213 (2022). doi: 10.1017/S0263574722000121.
[16] N. Mao, J. Chen, E. Spyrakos-Papastavridis and J. S. Dai, "Dynamic modeling of wheeled biped robot and controller design for reducing chassis tilt angle," *Robotica*, 1–29 (2024). doi: 10.1017/S0263574724001061.
[17] J. Guo, X. Huo, S. Guo and J. Xu, "A Path Planning Method for the Spherical Amphibious Robot Based on Improved A-star Algorithm," **In:** *2021 IEEE International Conference on Mechatronics and Automation (ICMA)*, Takamatsu, Japan (2021) pp. 1274–1279. doi: 10.1109/ICMA52036.2021.9512805.
[18] K. Wei, Y. Gao, W. Zhang and S. Lin, "A Modified Dijkstra's Algorithm for Solving the Problem of Finding the Maximum Load Path," **In:** *2019 IEEE 2nd International Conference on Information and Computer Technologies (ICICT)*, Kahului, HI, USA (2019) pp. 10–13. doi: 10.1109/INFOCT.2019.8711024.
[19] J. Guo, L. Liu, Q. Liu and Y. Qu, "An Improvement of D* Algorithm for Mobile Robot Path Planning in Partial Unknown Environment," **In:** *2009 Second International Conference on Intelligent Computation Technology and Automation*, Changsha, China (2009) pp. 394–397. doi: 10.1109/ICICTA.2009.561.
[20] K. Karur, N. Sharma, C. Dharmatti and J. E. Siegel, "A survey of path planning algorithms for mobile robots," *Vehicles* **3**(3), 448–468 (2021). doi: 10.3390/vehicles3030027.
[21] X. Wang, J. Wei, X. Zhou, Z. Xia and X. Gu, "AEB-RRT*: An adaptive extension bidirectional RRT* algorithm," *Auton Robot* **46**(6), 685–704 (2022). doi: 10.1007/s10514-022-10044-x.
[22] Q. Li, Y. Xu, S. Bu and J. Yang, "Smart vehicle path planning based on modified PRM algorithm," *Sensors* **22**(17), 6581 (2022). doi: 10.3390/s22176581.
[23] S. He, T. Xing and J. Ma, "Research on Solid Rate Filtering Technique based on Inverse Distance Weighted Interpolation of Navigation Radar," **In:** *2022 IEEE 10th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, Chongqing, China (2022) pp. 838–841. doi: 10.1109/ITAIC54216.2022.9836465.
[24] M. P. Gopika, G. R. Bindu, M. Ponmalar, K. Usha and T. R. Haridas, "Smooth PRM Implementation for Autonomous Ground Vehicle," **In:** *2022 IEEE 1st International Conference on Data, Decision and Systems (ICDDS)*, Bangalore, India (2022) pp. 1–5. doi: 10.1109/ICDDS56399.2022.10037275.
[25] N. Yang, L. Han, C. Xiang, H. Liu, T. Ma and S. Ruan, "Real-time energy management for a hybrid electric vehicle based on heuristic search," *IEEE Trans Veh Technol* **71**(12), 12635–12647 (2022). doi: 10.1109/TVT.2022.3195769.
[26] S. A. Eshtehardian and S. Khodaygan, "A continuous RRT*-based path planning method for non-holonomic mobile robots using B-spline curves," *J Ambient Intell Human Comput* **14**(7), 8693–8702 (2023). doi: 10.1007/s12652-021-03625-8.
[27] C. Li, X. Huang, J. Ding, K. Song and S. Lu, "Global path planning based on a bidirectional alternating search A* algorithm for mobile robots," *Comput Ind Eng* **168**, 108123 (2022). doi: 10.1016/j.cie.2022.108123.
[28] S. Katiyar and A. Dutta, "Comparative analysis on path planning of ATR using RRT*, PSO, and modified APF in CG-space," *Proc Inst Mech Eng Pt C: J Mech Eng Sci* **236**(10), 5663–5677 (2022). doi: 10.1177/09544062211062435.
[29] L. Zhang, X. Shi, Y. Yi, L. Tang, J. Peng and J. Zou, "Mobile robot path planning algorithm based on RRT_Connect," *Electronics* **12**(11), 2456 (2023). doi: 10.3390/electronics12112456.
[30] J. Wang, W. Chi, M. Shao and M. Q.-H. Meng, "Finding a high-quality initial solution for the RRTs algorithms in 2D environments," *Robotica* **37**(10), 1677–1694 (2019). doi: 10.1017/S0263574719000195.
[31] U. A. Umar, M. K. A. Ariffin, N. Ismail and S. H. Tang, "Hybrid multiobjective genetic algorithms for integrated dynamic scheduling and routing of jobs and automated-guided vehicle (AGV) in flexible manufacturing systems (FMS) environment," *Int J Adv Manuf Technol* **81**(9-12), 2123–2141 (2015). doi: 10.1007/s00170-015-7329-2.

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

[32] T. Wang, R. Dong, R. Zhang and D. Qin, "Research on stability design of differential drive fork-type AGV based on PID control," *Electronics* **9**(7), 1072 (2020). doi: 10.3390/electronics9071072.

[33] J. Xiao, X. Yu, K. Sun, Z. Zhou and G. Zhou, "Multiobjective path optimization of an indoor AGV based on an improved ACO-DWA[J]," *Math Biosci Eng* **19**(12), 12532–12557 (2022). doi: 10.3934/mbe.2022585.

[34] X. Cao and M. Zhu, "Research on global optimization method for multiple AGV collision avoidance in hybrid path," *Opti Control Appl Meth* **42**(4), 1064–1080 (2021). doi: 10.1002/oca.2716.

[35] V. Ruiz Molledo and J. E. Sierra Garcia, "Simulation tool for hybrid AGVs based on IEC-61131," *IEEE Lat Am Trans* **20**(2), 317–325 (2022). doi: 10.1109/TLA.2022.9661472.

[36] R. Lin, "Research into the automatic guidance system for AGVs used for logistics based on millimeter wave radar imaging," *Wire Commun Mobile Comput* **2022**, 8 (2022). doi: 10.1155/2022/3104017.

[37] A. G. Gad, "Particle swarm optimization algorithm and its applications: A systematic review," *Arch Computat Methods Eng* **29**(5), 2531–2561 (2022). doi: 10.1007/s11831-021-09694-4.

[38] Y. Qiao, Y. Fu and M. Yuan, "Communication-control co-design in wireless networks: A cloud control AGV example," *IEEE Internet Things J* **10**(3), 2346–2359 (2023). doi: 10.1109/JIOT.2022.3211766.

[39] P. Durst, X. Jia and L. Li, "Multi-Objective Optimization of AGV Real-Time Scheduling Based on Deep Reinforcement Learning," **In:** *42nd Chinese Control Conference (CCC)*, Tianjin, China (2023) pp. 5535–5540. doi: 10.23919/CCC58697.2023.10240797.

[40] X. Yuan, X. Yuan and X. Wang, "Path planning for mobile robot based on improved bat algorithm," *Sensors* **21**(13), 4389 (2021). doi: 10.3390/s21134389.

[41] Q. Si and C. Li, "Indoor robot path planning using an improved whale optimization algorithm," *Sensors* **23**(8), 3988 (2023). doi: 10.3390/s23083988.

[42] M. Jiang, D. Yuan and Y. Cheng, "Improved Artificial Fish Swarm Algorithm," **In:** *2009 Fifth International Conference on Natural Computation*, Tianjian, China (2009) pp. 281–285. doi: 10.1109/ICNC.2009.343.

[43] W.-M. Dai and E. S. Kuh, "Simultaneous floor planning and global routing for hierarchical building-block layout," *IEEE Tran Comput-Aided Des Integr Circuits and Syst* **6**(5), 828–837 (1987). doi: 10.1109/TCAD.1987.1270326.

[44] M. H. Korayem, S. Nosoudi, S. Khazaei Far and A. K. Hoshiar, "Hybrid IPSO-automata algorithm for path planning of micro-nanoparticles through random environmental obstacles, based on AFM," *J Mech Sci Technol* **32**(2), 805–810 (2018). doi: 10.1007/s12206-018-0129-x.

[45] A. Bhargava, M. Suhaib and A. S. Singholi, "A review of recent advances, techniques, and control algorithms for automated guided vehicle systems," *J Braz Soc Mech Sci Eng* **46**(7), 419 (2024). doi: 10.1007/s40430-024-04896-w.

[46] S. Abi, B. Benhala and H. Bouyghf, "A Hybrid DE-ACO Algorithm for the Global Optimization," **In:** *2020 IEEE 2nd International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS)*, Kenitra, Morocco (2020) pp. 1–6. doi: 10.1109/ICECOCS50124.2020.9314533.

[47] Z. Nie and H. Zhao, "Research on Robot Path Planning Based on Dijkstra and Ant Colony Optimization," **In:** *2019 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*, Shanghai, China (2019) pp. 222–226. doi: 10.1109/ICIIBMS46890.2019.8991502.

[48] G. Shial, S. Sahoo and S. Panigrahi, "An enhanced GWO algorithm with improved explorative search capability for global optimization and data clustering," *Appl Artif Intell* **37**(1), 1 (2023). doi: 10.1080/08839514.2023.2166232.

[49] T. M. Shami, A. A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M. A. Summakieh and S. Mirjalili, "Particle swarm optimization: A comprehensive survey," *IEEE Access* **10**, 10031–10061 (2022). doi: 10.1109/ACCESS.2022.3142859.

[50] T. M. Shami, A. A. El-Saleh and A. M. Kareem, "On the Detection Performance of Cooperative Spectrum Sensing using Particle Swarm Optimization Algorithms," **In:** *2014 IEEE 2nd International Symposium on Telecommunication Technologies (ISTT)*, Langkawi, Malaysia (2014) pp. 110–114. doi: 10.1109/ISTT.2014.7238187.

[51] X. Chen, Y. Zhao, J. Fan and H. Liu, "Three-Dimensional UAV Track Planning based on the GB-PQ-RRT* Algorithm," **In:** *42nd Chinese Control Conference (CCC)*, Tianjin, China (2023) pp. 4639–4644. doi: 10.23919/CCC58697.2023.10240961.

[52] W. Guan and K. Wang, "Autonomous collision avoidance of unmanned surface vehicles based on improved A-star and dynamic window approach algorithms," *IEEE Intel Transp Syst Magaz* **15**(3), 36–50 (2023). doi: 10.1109/MITS.2022.3229109.

[53] B. Cao, Z. Yang, L. Yu and Y. Zhang, "Research on the Star Algorithm for Safe Path Planning," **In:** *2023 IEEE International Conference on Control*, Jilin, China (2023) pp. 105–109. doi: 10.1109/ICCECT57938.2023.10141167.

[54] A. Bhargava, A. S. Singholi and M. Suhaib, "A Study on Design and Control of Omni-Directional Mecanum Wheels based AGV System," **In:** *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, Delhi, India (2023) pp. 1–6. doi: 10.1109/ICCCNT56998.2023.10306665.

[55] S. Li, J. Gu, Z. Li, S. Li, B. Guo, S. Gao, F. Zhao, Y. Yang, G. Li and L. Dong, "A visual SLAM-based lightweight multi-modal semantic framework for an intelligent substation robot," *Robotica* **42**(7), 1–15 (2024). doi: 10.1017/S0263574724000511.

[56] Z. Chen, X. Zhang, L. Wang and Y. Xia, "A Fast Path Planning Method Based on RRT Star Algorithm," **In:** *2023 3rd International Conference on Consumer Electronics and Computer Engineering (ICCECE)*, Guangzhou, China (2023) pp. 258–262. doi: 10.1109/ICCECE58074.2023.10135365.

[57] S. Zhang, A. Li, J. Ren and R. Ren, "Kinematics inverse solution of assembly robot based on improved particle swarm optimization," *Robotica* **42**(3), 833–845 (2024). doi: 10.1017/S0263574723001789.

[58] Y.-J. Pak, Y.-S. Kong and J.-S. Ri, "Robust PID optimal tuning of a delta parallel robot based on a hybrid optimization algorithm of particle swarm optimization and differential evolution," *Robotica* **41**(4), 1159–1178 (2023). doi: 10.1017/S0263574722001606.

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press

[59] Y. Liu, X. Wang, Y. Zhang and L. Liu, "An integrated flow shop scheduling problem of preventive maintenance and degradation with an improved NSGA-II Algorithm," *IEEE Access* **11**, 3525–3544 (2023). doi: 10.1109/ACCESS.2023.3234428.
[60] X. Liu, R. Feng, S. Zhou and Y. Yang, "A Novel PSO-SGD with Momentum Algorithm for Medical Image Classification," **In:** *2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, Houston, TX, USA (2021) pp. 3408–3413. doi: 10.1109/BIBM52615.2021.9669876.
[61] H. Jingjing, L. Xun, M. Wenzhe, Y. Xin and Y. Dong. Path Planning Method for Mobile Robot Based on Multiple Improved PSO. **In:** *2021 40th Chinese Control Conference (CCC)*, Shanghai, China (2021) pp. 1485–1489, 10.23919/CCC52363.2021.9550590
[62] M. H. Demir and M. Demirok, "Designs of particle-swarm-optimization-based intelligent PID controllers and DC/DC buck converters for PEM fuel-cell-powered four-wheeled automated guided vehicle," *Appl Sci* **13**(5), 2919 (2023). doi: 10.3390/app13052919.
[63] M. Ding, X. Zheng, L. Liu, J. Guo and Y. Guo, "Collision-free path planning for cable-driven continuum robot based on improved artificial potential field," *Robotica* **42**(5), 1350–1367 (2024). doi: 10.1017/S026357472400016X.
[64] H. Khan, S. Khatoon and P. Gaur, "Stabilization of wheeled mobile robot by social spider algorithm based PID controller," *Int J Inf Tecnol* **16**(3), 1437–1447 (2023). doi: 10.1007/s41870-023-01438-w.
[66] A. Bhargava, A. S. Singholi and M. Suhaib, "Design and Development of a Visual - SLAM based Automated Guided Vehicle," **In:** *2023 IEEE Fifth International Conference on Advances in Electronics, Computers and Communications (ICAECC)*, Bengaluru, India (2023) pp. 1–7. doi: 10.1109/ICAECC59324.2023.10560331.
[67] X. Y. Sandoval-Castro, S. Muñoz-Gonzalez, M. A. Garcia-Murillo, P. D. Ferrusca-Monroy and M. F. Ruiz-Torres, "Four-bar linkage reconfigurable robotic wheel: Design, kinematic analysis, and experimental validation for adaptive size modification," *Robotica* **42**(6), 1–15 (2024). doi: 10.1017/S026357472400078X.
[68] W. Chen, H. Cheng, W. Zhang, H. Wu, X. Liu and Y. Men, "Modeling and invariably horizontal control for the parallel mobile rescue robot based on PSO-CPG algorithm," *Robotica* **41**(11), 3501–3523 (2023). doi: 10.1017/S0263574723001133.
[69] B. Wang, P. Li, C. Yang, X. Hu and Y. Zhao, "Robotica: Decoupled elastostatic stiffness modeling of hybrid robots," *Robotica* **42**(7), 1–19 (2024). doi: 10.1017/S0263574724000675.
[70] Z. Weidong, H. Xianlin, G. Xiao-Zhi and P. Hongjun, "Multi-Objective Longitudinal Trajectory Optimization for Hypersonic Reentry Glide Vehicle based on PSO Algorithm," **In:** *34th Chinese Control Conference (CCC)*, Hangzhou, China (2015) pp. 2350–2356. doi: 10.1109/ChiCC.2015.7260001.
[71] I. A. Raptis, C. Hansen and M. A. Sinclair, "Design, modeling, and constraint-compliant control of an autonomous morphing surface for omnidirectional object conveyance," *Robotica* **40**(2), 213–233 (2022). doi: 10.1017/S0263574721000473.
[72] X. Liu, L. Wang and Y. Yang, "Model-free adaptive robust control based on TDE for robot with disturbance and input saturation," *Robotica* **41**(11), 3426–3445 (2023). doi: 10.1017/S0263574723001078.
[73] G. Gao, D. Li, K. Liu, Y. Ge and C. Song, "A study on path-planning algorithm for a multi-section continuum robot in confined multi-obstacle environments," *Robotica* 1–24 (2024). doi: 10.1017/S0263574724001383.
[75] B. Liu, G. Jiang, F. Zhao and X. Mei, "Collision-free motion generation based on stochastic optimization and composite signed distance field networks of articulated robot," *IEEE Robot Autom Lett* **8**(11), 7082–7089 (2023). doi: 10.1109/LRA.2023.3311357.
[76] T. D. Nguyen, "Kinematic Model and Stable Control Law Proposed for Four Mecanum Wheeled Mobile Robot Platform Based on Lyapunov Stability Criterion," **In:** *2023 International Symposium on Electrical and Electronics Engineering (ISEE), Ho Chi Minh*, Vietnam (2023) pp. 144–149. doi: 10.1109/ISEE59483.2023.10299844.

Cite this article: A. Bhargava, M. Suhaib and A. K. S. Singholi (2025). "An omnidirectional mecanum wheel automated guided vehicle control using hybrid modified A* algorithm", Robotica **43**, 449–498. https://doi.org/10.1017/S0263574724001954

https://doi.org/10.1017/S0263574724001954 Published online by Cambridge University Press