Robotica (2026), **44**, pp. 1096–1121
doi:10.1017/S0263574726103440

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

Path planning for Ackermann-steered robots based on bidirectional A* and timed elastic band algorithms

Zengshuai Wan[1], Xufei Wang[1,2], Jinbo Han[1] and Cheng Yang[1]

[1]Shaanxi University of Technology, Hanzhong, Shaanxi, China
[2]Shaanxi Provincial Aerospace Special Robot and Intelligent System Proof-of-Concept Center, Hanzhong, Shaanxi, China
Corresponding author: Xufei Wang; Email: wxf@snut.edu.cn

Received: 19 November 2025; **Revised:** 4 February 2026; **Accepted:** 9 April 2026; **First published online:** 28 May 2026

Keywords: Ackerman steering; bidirectional dynamic A* algorithm; TEB algorithm; path planning; ROS

Abstract

To address the issues of path roughness, low search efficiency, and the difficulty of simultaneously considering Ackermann vehicle kinematic constraints in practical path planning, this study developed a coordinated global–local path planning framework tailored for Ackermann-steered robots. In the global planning stage, a bidirectional A* search framework was enhanced with a dynamic heuristic function, improving the geometric quality of the global path while maintaining search efficiency. Path smoothing was applied to further align the path with Ackermann kinematic constraints. In the local planning stage, the Timed Elastic Band (TEB) algorithm was adapted for Ackermann steering by incorporating minimum turning radius and steering-related constraints, ensuring that the generated trajectories were both collision-free and executable. Python-based simulations on constructed maps were used to compare different heuristic strategies in terms of path length and search efficiency. Results showed that the proposed global planning strategy reduced path length by 34.62% compared with the traditional bidirectional A* algorithm and improved search efficiency by approximately 2.1%. Furthermore, Robot Operating System (ROS) simulations and real-robot tests were conducted to analyze curvature characteristics, velocity variations, and path-tracking performance under different global–local planning combinations. The experimental results demonstrated that the proposed planning framework offered strong engineering practicality and provided a feasible solution for real-world Ackermann-steered robots.

1. Introduction

As the warehousing and logistics sector transitions from traditional manual operations to intelligent and automated systems, mobile robots with Ackerman steering capabilities could complete efficient path planning and dynamic obstacle avoidance tasks in complex scenarios, thereby enhancing delivery efficiency. Path planning could be divided into global path planning and local path planning [1]. Global path planning algorithms included A*, Dijkstra, RRT*, etc. Local planning algorithms included Timed Elastic Band (TEB), APF, Dynamic Window Approach (DWA), etc. Gong et al. [2] proposed probabilistic roadmap algorithm, referred to as TL-PRM. By constraining the sampling regions and introducing a path reconnection strategy, the algorithm significantly reduced the number of sampling nodes and the computation time in both static and dynamic environments, while maintaining high path quality and motion safety. Qi et al. [3] proposed a bidirectional optimization algorithm, in which Q-learning was employed to drive an adaptive heuristic mechanism, and a spatial discretization strategy based on LPA* was designed to maintain path feasibility. For instance, the A* algorithm conducts heuristic searched by combining the cost function with estimated costs. Its advantage was that it had global optimality in static environments, but it relied on precise maps and its efficiency decreased as the number of nodes increases, which wasn't conducive to the real-time movement of mobile robots. This study focused on applying global–local path planning methods in practical engineering scenarios, optimizing global

© Shaanxi University of Technology, 2026. Published by Cambridge University Press.

https://doi.org/10.1017/S0263574726103440 Published online by Cambridge University Press

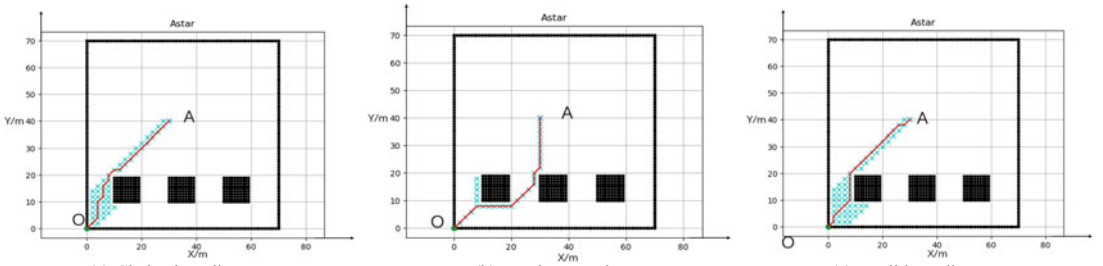
search with a dynamic heuristic strategy, and integrating corresponding local planning to enhance the path smoothness and motion stability of Ackermann-steered robots.

In recent years, many scholars have conducted in-depth research on the A* algorithm. Bhargava et al. [4] proposed a hybrid optimization framework that integrates an improved A* algorithm with the DWA, enabling efficient and accurate navigation in static and dynamic environments. Liu et al. [5] further improved the IGWO algorithm based on A* algorithm, which further improved the number of iterations and path length for parking lot path planning scenarios. Dai et al. [6] improved the LPA-star algorithm to meet the dynamic path planning task. Yang et al. [7] integrated the A* and APF algorithms to achieve the goal of improving vehicle safety and stability, but this increased the time required to generate the path. Wang et al. [8] effectively solved the problem of long search time by generating a topological map for the A* algorithm search. Li et al. [9] introduced the heuristic function of the A* algorithm in an exponentially weighted manner and expanded it to the 24 search domains, thereby improving the search efficiency of the algorithm. Li et al. [10] proposed a rolling optimization algorithm to remove unnecessary search points during the map search process, thereby improving the A* algorithm and enabling dynamic collision avoidance for ships. Yu et al. [11] utilized the potential field function of APF to guide the search of the A* algorithm and proposed the A*-APF new algorithm. Compared with other algorithms, the new algorithm could produce better and smoother path planning results. Wu et al. [12] improved the bidirectional A* algorithm by starting the search from the starting point and the end point separately, thereby improving the search efficiency. Chen et al. [13] addressed the problem of many sharp turns and lack of smoothness in the search of the traditional bidirectional A* algorithm, proposing an improved bidirectional A* algorithm combined with B-spline curves for path planning integration, making the robot's path tracking more stable. Lin et al. [14] utilized redundant safety space to filter narrow roads that were impassable, integrated the predictive planning strategy and redundant turning point elimination strategy, and obtained a better path, but ignored the running efficiency. Tang et al. [15] optimized the A* algorithm by combining the advantages of interpolation algorithms and optimized the irregular path through a screening function, but the algorithm was only applied to static scenes and didn't consider dynamic scenes. Chen et al. [16] combined the improved A* algorithm with APF for path planning and conducted dynamic obstacle avoidance experiments, significantly improving the efficiency of path planning. Bhargava et al. [17] used A* algorithm and PSO algorithm to determine the optimal path for the omnidirectional McNum wheel, overcame the shortcomings of traditional A* algorithm, and compared it with a variety of traditional algorithms.

In summary, the improvement of the A* algorithm could add a weighting function to the heuristic function of the A* algorithm, allocated different weights to different search nodes, so as to find the optimal path. The selection of the weight coefficient in the A* algorithm was one of the research difficulties, and an excessive or insufficient value of the weight function led to different results. Therefore, this study took the selection of the weight coefficient of the bidirectional A* algorithm heuristic function as the research object and explored that it should be a dynamic function rather than a constant. In the early stage of search, the weight function was large, saving the algorithm's computational consumption, and gradually reduced the weight in the middle and later stages, making the algorithm pay more attention to the safety and accuracy of the path.

For TEB algorithms, Li et al. [18] proposed that TEB algorithms had strong real-time performance but may encounter local optima. Houshyari et al. [19] mentioned that it optimized the trajectory of the robot locally and optimized the trajectory execution time. Chen et al. [20] proposed the fuzzy logic control – TEB algorithm for the TEB algorithm, introducing trajectory smoothness constraints to enhance the stability of the speed input. However, it only considered the narrowness of the steering characteristics and the complexity of the turn for trajectory complexity, without considering a more precise model. Yin et al. [21] proposed a fusion algorithm based on RRT* and TEB algorithm and optimized the TEB algorithm. They transformed the TEB algorithm, which considered kinematic constraints, into a nonlinear least squares problem, and introduced penalty functions to adjust the priority of constraints. Wang et al. [22] introduced the shortest distance constraint to make the local path closer to the path edge,

https://doi.org/10.1017/S0263574726103440 Published online by Cambridge University Press



(a) Chebyshev distance (b) Manhattan Distance (c) Euclidean distance

Figure 1. *Schematic diagram of search distance.*

improving the efficiency of local path planning. Wu et al. [23]. enhanced the TEB algorithm by incorporating the shortest distance constraint into the multi-objective optimization problem, thereby reducing the planning time of AGVs by 5%. This improvement also enabled obstacle avoidance in unknown environments.

The above research focused on improving the operational efficiency of the algorithm itself, without taking into account the different movement modes of the mobile robot. Since the Ackerman steering robot has the characteristic of the smallest turning radius, this study proposed a method that integrated global and local planning. The global planner was used to handle the static environment, and the turning constraints were introduced to adjust the robot's movement posture, ensuring that it could not only avoid obstacles but also conform to the kinematic constraints of the Ackerman mechanism in complex environments.

2. Classical A* and TEB algorithms

2.1. Classic A algorithm*

The A* algorithm is a heuristic search method that determines the node with the minimum cost by jointly considering the actual cost from the start node to the current node and the heuristic estimated cost from the current node to the goal. The core idea of A* is to combine the actual cost and the estimated cost to select an optimal path, thereby improving the efficiency of path searching. The cost function was defined as Eq. (2.a):

$$f(n) = g(n) + h(n) \tag{2.a}$$

In Eq. (2.a), $f(n)$ represented the overall priority of a node when it was selected, $g(n)$ represented the cost from the start node to the current node, and $h(n)$ represented the estimated cost from the current node to the goal. When the heuristic cost estimate didn't exceed the true minimum cost from the current node to the goal and maintained a reasonable increasing relationship during path expansion, the A* algorithm could guarantee a globally optimal solution. However, in large-scale or complex environment maps, traditional heuristic functions tended to cause extensive expansion of irrelevant nodes in the early stage of the search, which significantly increased computational burden and consequently degraded search efficiency.

The optimization of the traditional A* algorithm mainly involved modifying the expression of the heuristic function $h(n)$, which typically included the Chebyshev distance, Euclidean distance, and Manhattan distance. A map was created in Python to compare the search paths generated by these three heuristic functions. The obstacles were rectangular regions with a length and width of 10 units, spaced 10 units apart. The start point was O(0, 0), and the goal point was A(30, 40). The results produced by the different heuristic functions were shown in Figure 1.

As shown in Figure 1, it could be observed that the turning point of the Euclidean distance was relatively small. However, for the Ackermann mobile mode robot, the smoother the planned path was,

https://doi.org/10.1017/S0263574726103440 Published online by Cambridge University Press

the more it conformed to its motion constraints. Therefore, in this experiment, the Euclidean distance was selected as the heuristic function for research.

As shown in Figure 1, the A* algorithm based on the Euclidean distance produced smoother paths with more continuous directional changes compared to other heuristic functions. For Ackermann-steering mobile robots, smoother paths could better satisfy their motion constraints; therefore, this study adopted the Euclidean distance as the heuristic function. To improve search efficiency, some researchers have proposed the weighted A* algorithm, whose evaluation function was given in Eq. (2.b) [24]:

$$f(n) = g(n) + w(n) \cdot h(n) \tag{2.b}$$

where $w(n)$ is a weighting function. As the value of $w(n)$ increased, the weighted A* algorithm could rapidly advance toward the goal in the early stages of the search, significantly reducing the number of expanded nodes and improving planning speed. However, the performance of the weighted A* algorithm was highly sensitive to the value of $w(n)$. When $w(n)$ was too large, the algorithm overly relied on the heuristic function, which could lead to a substantial decline in path quality or could even generate unsafe paths close to obstacles. Conversely, when $w(n)$ was too small (e.g., $w(n) = 1$), the algorithm degenerated to the traditional A*, making it difficult to achieve high search efficiency. Therefore, a fixed weighting function struggled to achieve a global balance between search efficiency and path quality.

The characteristics required by the traditional A* algorithm differ across search stages. In the early stage, the primary goal was to quickly narrow the search space and established a rough connection between the start and goal points. A higher heuristic weight at this stage could reduce the expansion of irrelevant nodes and thus improved search efficiency. In the later stage, as the path approaches the goal region, the demand for path safety increased significantly. Appropriately reducing the value of $w(n)$ at this stage could yield higher-quality paths. Therefore, the weighting function $w(n)$ is formulated as a dynamic heuristic. Its value is tuned to be large during the initial search phase and is gradually decreased in the middle to late phases. This method achieves a dynamic balance between the efficiency of the path search and the optimality of the final path.

For the bidirectional A* search framework, the search progressed simultaneously from both the start and goal points, which generally resulted in faster search efficiency compared to the traditional A*. However, since the meeting point of the two search fronts was uncertain, the design of the heuristic function played a more critical role in path quality and search stability. Introducing a dynamic heuristic function into the bidirectional A* algorithm could also improve the quality of the generated paths while maintaining computational efficiency. This, in turn, provided a more reasonable global reference path for subsequent local planning algorithms, reducing local oscillations.

2.2. The basic principle of the TEB algorithm

The TEB algorithm is one of the commonly used local planners. Its core idea is to represent a robot's trajectory as a sequence of discrete poses, each associated with a time interval. This sequence is then optimized under a set of constraints to generate a feasible trajectory that satisfied kinematic and dynamic limits. However, the traditional TEB algorithm was often based on omnidirectional or differential-drive robot models and didn't account for the vehicle's minimum turning radius. In the context of Ackermann-steering mobile robots studied in this work, using the classical TEB algorithm alone couldn't fully reflect the robot's true kinematic characteristics and may generate trajectory segments with excessive curvature or physically infeasible paths. Although the resulting trajectory was geometrically continuous and satisfied obstacle avoidance requirements, in practice it may cause the robot to decelerate frequently or may make in-place adjustments. To address this issue, a minimum turning radius constraint was introduced to account for the Ackermann-steering property, where the vehicle couldn't rotate in place. This constraint addresses the limitations inherent in using A* as a global planner, thereby facilitating the smooth operation of the mobile robot [25].

To ensure that the generated local trajectory conforms to Ackermann-steering kinematics, relevant TEB parameters for trajectory planning and obstacle avoidance were set accordingly. The minimum

https://doi.org/10.1017/S0263574726103440 Published online by Cambridge University Press

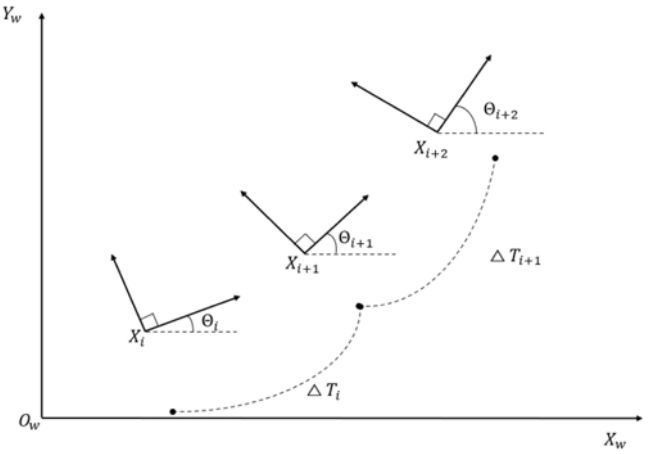


Figure 2. *Pose sequence and time interval diagram of mobile robot.*

turning radius was defined as turning radius = wheelbase/ tan (maximum steering angle), which was then incorporated as a steering constraint. In addition, penalty terms for steering angular velocity and angular acceleration were added to limit the speed and inertia of the steering mechanism. These modifications ensured that the generated local trajectory satisfied obstacle avoidance and dynamic constraints while better matching the actual steering capability of the Ackermann vehicle, thereby improving trajectory smoothness. Figure 2 showed the relationship between the robot's poses and time intervals during motion.

Figure 2 showed the layout of the position x_i, y_i, and azimuth Θ_i of the mobile robot at a certain state. That was, $X_i = (x_i, y_i, \Theta_i)^T$ represented the current position and orientation of the robot, and $\triangle T_i$ represented the time interval between two consecutive positions. From the figure, it could be seen that TEB generated n-1 time differences $\triangle T_i$ in two consecutive position sequences. The pose sequence Q represented a series of discrete poses of the mobile robot over time during its motion. The position sequence Q was as shown in Eq. (2.c):

$$Q = \{X_i\}_{i=0.1.....n-1} \tag{2.c}$$

Each time difference $\triangle T_i$ represented the time required for the robot to move from one pose to the next. The time sequence τ not only accounted for the velocity and acceleration constraints during the robot's motion but also allowed the TEB algorithm to optimize the geometric shape of the trajectory. The set of all time differences was represented by τ as shown in Eq. (2.d):

$$\tau = \{\Delta T_i\}_{i=0,1,...,n-1} \tag{2.d}$$

TEB was defined as a tuple of two sequences: using B to represent the joint optimization of the pose sequence Q and the time sequence τ, enabling the TEB algorithm to generate trajectories that were feasible both spatially and temporally, as expressed in Eq. (**2**.e):

$$B = (Q, \tau) \tag{2.e}$$

When optimizing the TEB algorithm, it could be adjusted in real time based on the pose and time interval, and multiple objectives could be optimized simultaneously. Its solution model was expressed as Eq. (2.f):

$$f(B) == \sum_{k} \gamma_k f_k(B) \tag{2.f}$$

In the above equation, $f(B)$ represented the objective function, B^* was the optimized TEB trajectory, $f_k(B)$ was the constrained objective function, and γ_k was the weight value used to measure the importance of each objective function. Considering that the constraints on linear velocity and linear

https://doi.org/10.1017/S0263574726103440 Published online by Cambridge University Press

acceleration were important steps in the above solution model, the constraints on linear velocity V_i and linear acceleration a_i were respectively Eqs. (2.g) and (2.h):

$$V_i \approx \frac{d_{i+1}}{\Delta T_i} \tag{2.g}$$

$$a_i \approx \frac{2(V_{i+1} - V_i)}{\Delta T_i + \Delta T_{i+1}} \tag{2.h}$$

In the formula, d_{i+1} represented the Euclidean distance between adjacent points, and the corresponding angular velocity ω_i and angular acceleration constraints were α_i as shown in Eqs. (2.i) and (2.j):

$$\omega_i \approx \frac{\theta_{i+1} - \theta_i}{\Delta T_i} \tag{2.i}$$

$$\alpha_i \approx \frac{2(\omega_{i+1} - \omega_i)}{\Delta T_i + \Delta T_{i+1}} \tag{2.j}$$

The curvature $\rho \approx \omega_i / V_i$. The constraint on curvature was expressed by Eq. (2.k) as follows:

$$f_{curv}(B) = \sum_{i=0}^{n-1} \max\left(0, |p_i| - p_{\max}\right)^2 \tag{2.k}$$

The expression $max(0, x)$ indicated that a penalty was imposed when ρ_i was greater than ρ_{max}. The value of k represented the penalty coefficient, and $\rho_{max} = 1/R_{min}$. At this point, the objective function was expanded to Eq. (2.l):

$$f(B) = \gamma_{time} f_{time}(B) + \gamma_{obst} f_{obst}(B) + \gamma_{vel} f_{vel}(B) + \gamma_{curv} f_{curv}(B) \tag{2.l}$$

The value of γ was all greater than or equal to 0, representing the weights of each constraint. γ_{time}, γ_{obst}, and γ_{vel}, respectively, represented the weights for the time, obstacle distance, and velocity constraints. Then the final optimization objective function was Eq. (2.m):

$$B^* = \arg\min_B f(B) \tag{2.m}$$

3. Global path planning method for Ackermann-steered robots

The bidirectional A* algorithm significantly improved the search efficiency compared to the traditional A* algorithm, but it had the problem of having too many path turning points. In literature [26], a 12-domain search was adopted, and the obstacle expansion radius was introduced. The heuristic function was dynamically adjusted based on the Manhattan distance between the unobstructed grid and the expanded radius of the nearest obstacle grid. Finally, cubic spline interpolation was used for smoothing processing. However, it only studied the static environment and didn't consider the dynamic and complex environment.

3.1 Path search strategy based on dynamic heuristics

The goal of the bidirectional dynamic heuristic global planning scheme was to improve search efficiency while finding a more optimal path. In the early stage of the search, the heuristic weight function was set relatively high to reduce computational cost, and it gradually decreased in the middle and later stages, allowing the algorithm to place more emphasis on path safety and accuracy. The weight function $w(n)$ was designed to decrease progressively during the search, with $w(n) = 1.0$ corresponding to the traditional A* global search; therefore, $w(n) \geq 1.0$. The definition of $w(n)$ was given in Eq. (3.a):

$$w(n) = \begin{cases} w_0 - \alpha \cdot \left(1 - \dfrac{d_n}{D}\right), d_n \leq D \\ 1, d_n > D \end{cases} \tag{3.a}$$

https://doi.org/10.1017/S0263574726103440 Published online by Cambridge University Press

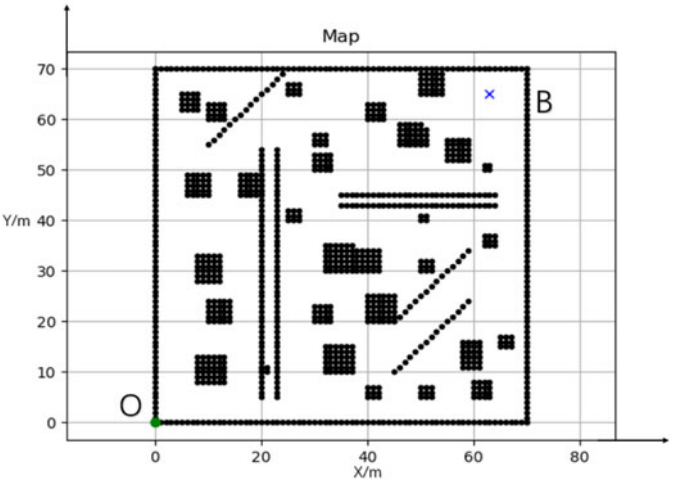


Figure 3. *Schematic diagram of obstacles.*

In the formula, ω_0 represented the base weight, which ensured that the search speed was accelerated in the initial stage and the search nodes were reduced. α was the distance attenuation coefficient, which ensured that the weight function gradually decreased as the search progresses. The range of α was [0,1]. When α was large, the weight of the path in the later stage was low, and the search efficiency was reduced. When α was small, there may be a situation of missing the optimal path. d_n was the Euclidean distance from the current node to the end point. D was the Euclidean distance from the starting point to the end point. The evaluation function of the Bidirectional Dynamic Heuristic A* (BI-Dynamic A*) algorithm was given in Eq. (3.b):

$$f(n) = \begin{cases} g(n) + \left[w_0 - \alpha \cdot \left(1 - \dfrac{d_n}{D} \right) \right] \cdot h(n), d_n \le D \\ g(n) + h(n), d_n > D \end{cases} \tag{3.b}$$

According to the changes in Eq. (3.b), there was the case of Eq. (3.c):

$$f(n) = g(n) + (w_0 - \alpha) \cdot h(n), d_n = 0 \tag{3.c}$$

Based on the content of Literature [27], the heuristic function in the bidirectional dynamic heuristic scheme should make $(\omega_0 - \alpha) \geq 1$. The experimental environment was Ubuntu 20.04 system, Intel i5-12,400 CPU, NVIDIA GeForce RTX3080 graphics card, and VS Code simulation software. The improved algorithm was implemented and visualized using Python 3.8 language. The grid map of 70 m × 70 m as shown in Figure 3 was designed. The starting point and the ending point were (0,0) and (63,65), respectively.

Since the parameters of the dynamic heuristic function only affected the search phase, their values directly influenced factors such as node expansion order, search time, and the number of expanded nodes. In contrast, path smoothing was a post-processing step performed after the search was completed and didn't affect the search process or its computational complexity. Therefore, the optimal parameters of the dynamic heuristic function were determined prior to the path smoothing procedure. During the search phase, different parameter combinations were compared to select the dynamic heuristic parameters that achieved the best balance between search efficiency and the quality of the original path.

According to Eq. (3.a), the range of ω_0 was set to [1.0,3.75] with a step size of 0.25. The range of α was set to [0.1,1] with a step size of 0.2. The experiments were conducted for the parameter values of ω_0 and α to change in the groups as shown in Table I to determine the optimal parameter values.

https://doi.org/10.1017/S0263574726103440 Published online by Cambridge University Press

Table I. *Experimental serial number table with different parameter values*

Experiment number	ω_0	α	Experiment number	ω_0	α
1	1.00	0	29	2.75	0.5
2	1.25	0.1	30	2.75	0.7
3	1.50	0.1	31	2.75	0.9
4	1.50	0.3	32	2.75	1
5	1.75	0.1	33	3	0.1
...	...	...	...	...	...
27	2.75	0.1	55	3.75	0.9
28	2.75	0.3	56	3.75	1

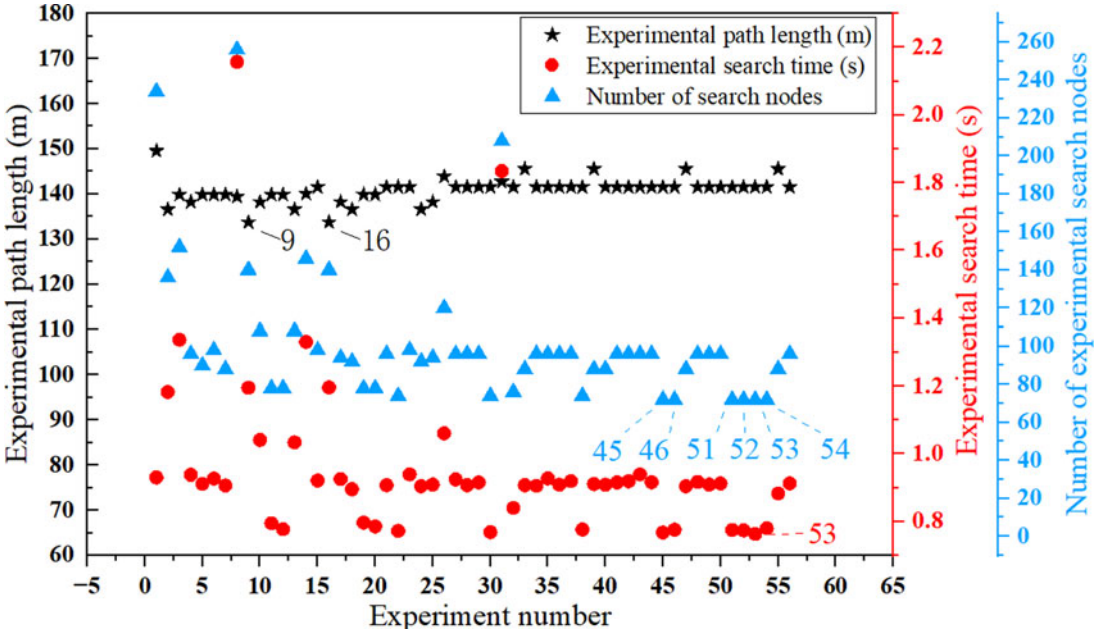


Figure 4. *Comparison of search time and path length for different parameter values.*

Multiple experiments were conducted on the experimental sequence table in Table I, and the results of path length and search time for each group were shown in Figure 4.

From the experimental data in Figure 4, it could be observed that as the parameter values increased, the path length of the BI-Dynamic A* algorithm fluctuated initially but stabilizes once the parameters exceeded a certain threshold. The main reason was that, in the constructed complex obstacle map, narrow passages impose strong constraints, resulting in highly consistent feasible paths. Variations in the dynamic heuristic function primarily affected the number of search nodes and the search time, while having limited impact on the geometric length of the generated path. As the search strategy gradually transitioned to expansions dominated by actual cost, the algorithm converged to a stable path solution, thereby reducing the differences in path length across different parameter combinations.

Experimental Groups 9 and 16 achieved the best path quality, although the differences compared with other groups were relatively small. In contrast, Experimental Group 53 outperformed the others in terms of both search time and the number of expanded nodes. Therefore, under the premise that the path length didn't exhibit a significant disadvantage, parameter combinations with shorter search time and

https://doi.org/10.1017/S0263574726103440 Published online by Cambridge University Press

fewer node expansions were preferred to achieve a reasonable balance between path quality and search efficiency. In this study, Experimental Group 53 was selected, with $\omega_0 = 3.75$ and $\alpha = 0.5$.

3.2. Path smoothing

Since the dynamic heuristic function significantly reduced the heuristic weight in the later stage of the search, the generated global path became smoother in terms of curvature variation and node distribution, providing a better initial solution for subsequent TEB-based local trajectory optimization. However, geometric deficiencies still remained, making further path smoothing necessary.

3.2.1. Introduction and motivation

The global path generated by the A* algorithm was essentially a connection of discrete grid cell centers, which still exhibited notable geometric deficiencies.

1. The resulting path consisted of a large number of polyline segments, leading to abrupt changes in turning angles at corners.
2. During global path generation, the nonholonomic constraints of the vehicle weren't considered, making the path difficult to be smoothly tracked by an Ackermann-steering robot.

As a result, directly using such a polyline path as the reference trajectory often caused frequent steering adjustments in the initial or turning regions during local planning and may even lead to oscillatory behavior. Therefore, an essential technical procedure involves generating an updated reference path through a smoothing method, based on the feasible original global path. This not only enhances the geometric continuity of the path but also ensures its compliance with the kinematic constraints of the Ackermann-steering model.

3.2.2. Selection of the smoothing method

When performing secondary optimization on the path generated by the algorithm, B-spline curve smoothing or Bézier curve smoothing was commonly adopted. The advantages of Bézier curves were mainly reflected in the following aspects:

1. By properly setting control points, continuity in both position and tangential direction could be ensured, which significantly reduced abrupt changes in turning angles along the path.
2. The shape of a Bézier curve was constrained by its control polygon, resulting in relatively smooth curvature variation. This property facilitated subsequent curvature analysis and was well aligned with the minimum turning radius constraint of Ackermann steering.
3. Structurally, a Bézier curve was determined by a finite number of control points, leading to a relatively stable computation process and easy integration into the subsequent Robot Operating System (ROS)-based planning framework. The Bézier curve equation was shown in Eq. (3.d): [28]

$$B(t) = \sum_{i=0}^{n} \binom{n}{i} P_i (1-t)^{n-i} t^i \tag{3.d}$$

In the formula, P_i is expressed as the coordinate value of each point and n represents the order.

3.2.3. Cooperative optimization mechanism of path smoothing and the TEB algorithm

Bezier curve smoothing was a necessary step between global and local path planning, but it does not replace the local planner. First, the BI-Dynamic A* algorithm generates a topologically feasible and collision-free discrete path in complex environments. Next, the Bezier curve smooths and geometrically optimizes this discrete path to produce a continuous global reference path with controlled curvature.

https://doi.org/10.1017/S0263574726103440 Published online by Cambridge University Press

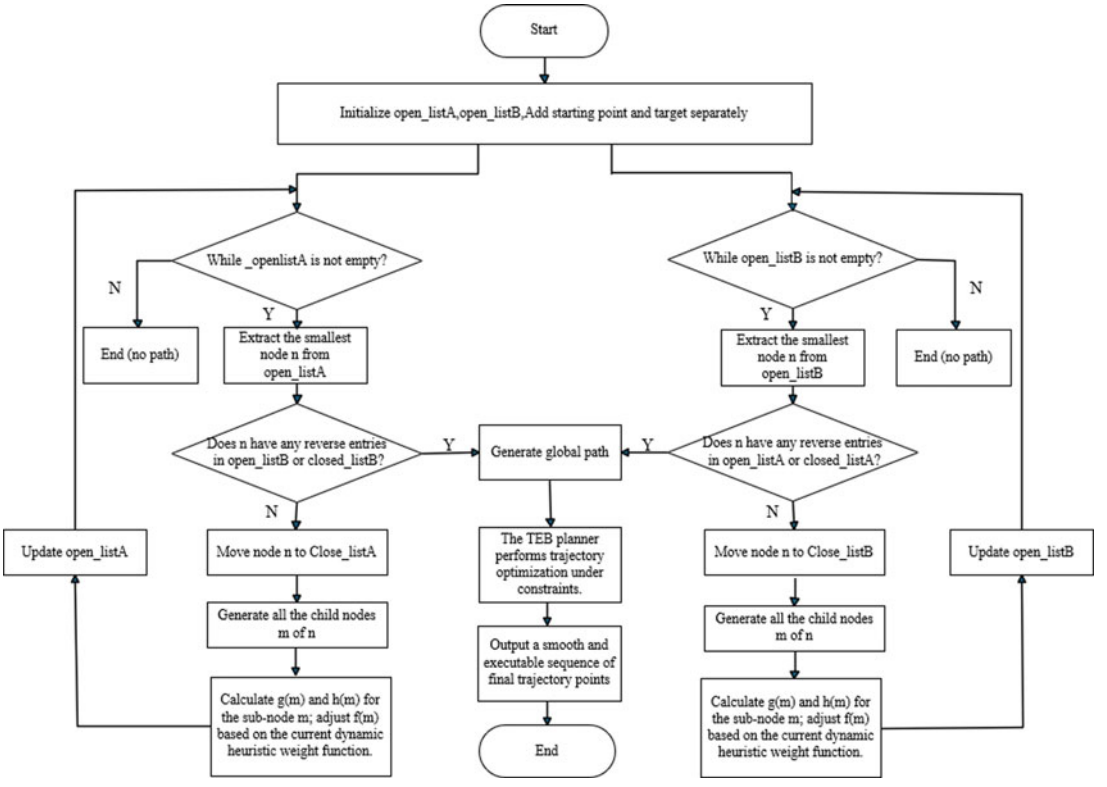


Figure 5. *Global–local path planning flow chart.*

Finally, the smoothed path is fed into the TEB local planner, which optimizes an executable trajectory considering kinematic constraints such as velocity, acceleration, and minimum turning radius. This collaborative mechanism ensures global path optimality while significantly enhancing the motion stability of Ackermann-steering robots.

3.3. Path planning workflow for Ackermann steering

Using a bidirectional A* algorithm could improve path search efficiency and reduced computation time. This study adopted an overall collaborative search workflow combining the BI-Dynamic A* algorithm with the TEB algorithm, as shown in Figure 5.

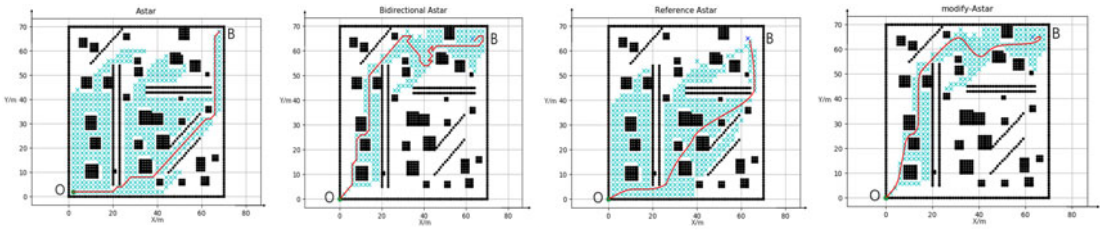
3.4. Theoretical analysis of the BI-Dynamic A algorithm*

The dynamic heuristic function employed in this study adaptively adjusts the weight between the heuristic term and the actual cost. This mechanism enables the search strategy to dynamically tune its expansion behavior, thereby achieving an effective balance between search efficiency and path quality. Therefore, this section analyzed the theoretical properties of the proposed method from two perspectives: the boundedness of the heuristic function and the convergence behavior of the algorithm during the search process.

3.4.1. Boundedness analysis of the dynamic heuristic function

The dynamic heuristic function proposed in this study adaptively adjusted the traditional A* heuristic by introducing a distance-related weight decay mechanism with respect to the goal. Throughout the entire

https://doi.org/10.1017/S0263574726103440 Published online by Cambridge University Press



(a) Traditional A* (b) BI-A* (c) Reference[26] A* (d) BI-Dynamic Heuristic A*

Figure 6. *Algorithm path planning trajectory.*

search process, it satisfied Eq. (3.e):

$$f(n) = g(n) + w(n) \cdot h(n) \leq g(n) + w_0 \cdot h(n) \tag{3.e}$$

In Eqs. (2.b) and (3.a), $\omega(n)$ is the weighting function, which gradually decayed as the distance to the goal decreases. ω_0 is the base weight, which monotonically decreased during the search and eventually approaches 1. This ensured that the total cost of the generated path remained controllable compared to the optimal path produced by the traditional A* algorithm. In other words, the path quality, particularly the path length, was constrained by the initial weight ω_0, preventing uncontrolled degradation of the path. This property provided an important guarantee for the subsequent integration with TEB-based local trajectory optimization.

3.4.2. Analysis of algorithm convergence speed and node expansion scale

In traditional weighted bidirectional A* search, the heuristic function remained constant throughout the search process. While effective for initial guidance, this strategy suffers from premature search bias in local regions as the process continues. This results in two key issues: an increased number of path turns and unstable, oscillatory expansion of waypoints.

To mitigate this issue, this study introduced a dynamic heuristic function that decayed with the distance to the goal, allowing the search strategy to exhibit different behavioral characteristics at different stages. In the early stage of the search, the heuristic weight was relatively high, directing the search toward the goal. As the search progresses, the weight gradually decreased, and the overall search behavior transitioned to a traditional cost-dominated expansion strategy. During the search, the approach didn't solely aim to minimize the number of expanded nodes, but rather sought to balance search efficiency and path quality as much as possible.

From the perspective of path optimization, these characteristics were of significant importance. A relatively smooth global path with more evenly distributed turning points provided a higher-quality initial solution for TEB-based local trajectory optimization. Consequently, subsequent optimization could satisfy velocity, acceleration, and Ackermann kinematic constraints within a smaller adjustment range, thereby improving the overall stability of the planning process. The above analysis regarding behavioral characteristics, node expansion scale, and path quality would be further validated and illustrated through simulation experiments in Chapter 4.

4. Simulation experiments and analysis

4.1. Comparison of global planning methods

The comparison of global path planning trajectories in the experimental map was shown in Figure 6. (The bidirectional A* algorithm will be referred to as BI-A* in the following text; the following text simplifies BI-Dynamic Heuristic A* to BI-Dynamic A*).

It could be observed from Figure 6 that the smoothness of the paths varies across the different algorithms. A timing function was added to each algorithm to calculate the running time for search nodes.

https://doi.org/10.1017/S0263574726103440 Published online by Cambridge University Press

Table II. *Experimental simulation data of different programming algorithms*

Algorithm	Path length/m	Search time /s	Number of nodes
Traditional A*	112.08	32.517	474
BI-A*	149.54	2.00	232
Reference [**26**]A*	96.17	3.815	412
BI-dynamic heuristic A*	97.76	1.9582	234

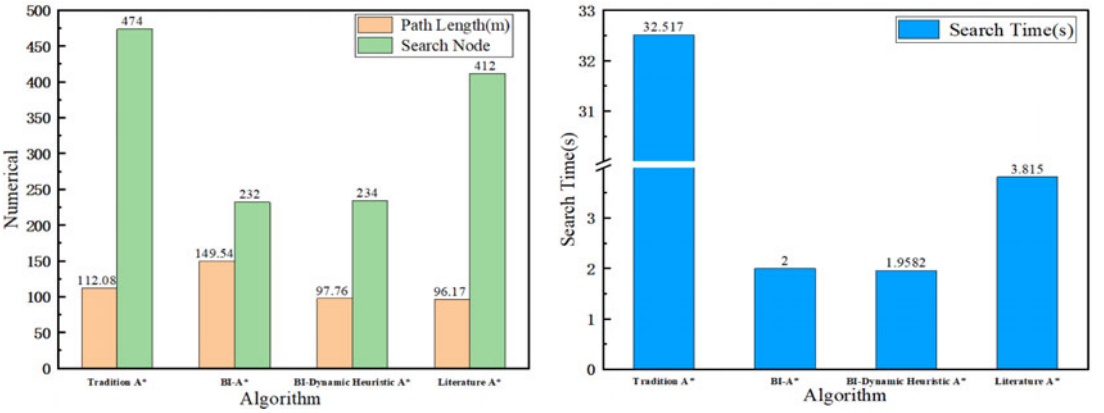


Figure 7. *Comparison of experimental results of different algorithms.*

The distances between adjacent points were then accumulated to obtain the total path length. The simulation data for the three algorithms was shown in Table II. Based on the results from Table II, Figure 7 was plotted to visually display the relevant data for each algorithm.

From Figure 7, it was visually clear that the BI-Dynamic A* algorithm outperforms other algorithms in terms of path length, although the number of search nodes was higher than that of the bidirectional A* algorithm. This could be attributed to the smoothing process applied to the path and the emphasis on selecting a better path, which led to a higher number of nodes being searched. However, the total number of search nodes was still fewer than that of the traditional A* algorithm and the method from Ref. [26]. According to Table II, the path length of the bidirectional dynamic heuristic method was reduced by 34.62% compared to bidirectional A*, and it increased by 1.653% compared to the algorithm from Ref. [26]. The reason for this was that, in order to balance search efficiency, the path generated by this method was not as optimal as that in Ref. [26], but the difference was small.

In terms of search time, the bidirectional dynamic heuristic method improved search efficiency by approximately 2.1% compared to traditional bidirectional A* and by 48.7% compared to the method in Ref. [26]. The number of search nodes decreased by 51.05% compared to traditional A*, increased by 0.85% compared to bidirectional A*, and decreased by 43.2% compared to the method in Ref. [26]. The reason for the increase in search nodes in the method from Ref. [26] was that it attempted 12 neighborhood searches at each intersection in search of the most optimal path, which resulted in an increase in the number of search nodes.

4.2. Model establishment of Ackerman mobile robot

The mobile robot was modeled and assembled in SolidWorks, with its components exported to URDF format. Subsequently, by configuring the xacro files for each robotic joint, the full model was constructed in ROS and imported into Gazebo for simulation. Figure 8 illustrates the state of the robot model during a turn.

https://doi.org/10.1017/S0263574726103440 Published online by Cambridge University Press

Table III. *Kinematic parameters of the robot*

Motion parameters	Numeric value
Maximum forward speed/(m/s)	1.0
Maximum Retraction Speed /(m/s)	1.0
Maximum steering angular velocity /(rad/s^2)	2.5
Maximum linear acceleration /(m/s^2)	2.5
Maximum angular acceleration /(rad/s^2)	2.5
Minimum turning radius /m	0.25

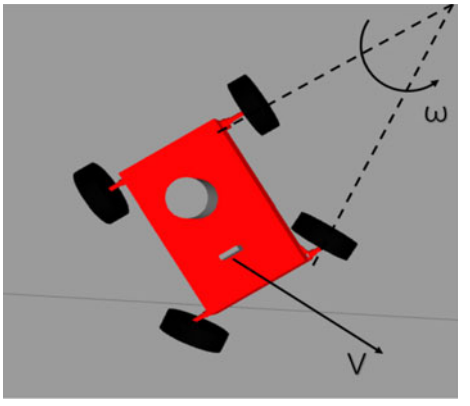


Figure 8. *Robot turning moment motion diagram in Gazebo.*

The kinematic parameters of the mobile robot were set as shown in Table III.

4.3. ROS-based path planning engineering simulation

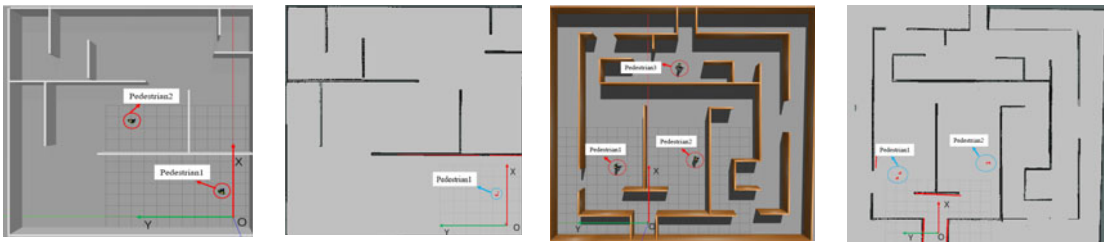
In this experiment, the values of ω_0 and α were set to be the same as those used in the Python-based experiments. The experimental software platform was Ubuntu 20.04 with ROS Noetic. The kinematic model adopted an ideal Ackermann-steering geometry model without considering steering delay. The simulation environment focused on examining the impact of moving obstacles on the path planning scheme, assuming that the environmental perception module could provide reliable cost map inputs. This allowed for the evaluation of the feasibility and practicality of the global–local planning approach in real-world applications.

The environment maps configured in Gazebo had sizes of 15 m × 14 m and 15 m × 18 m, respectively. As shown in Figure 9, the robot started from point O at (0, 0, 0), and fixed target coordinates were issued to the robot using the rostopic command. The target points in the two environments were set to (15, 15) and (18, −12), respectively. Dynamic pedestrians were initialized at the following positions: (2, 1, 0) and (8, 7, 0) in the simple map; (5, 4, 0), (6, −5, 0), and (16, −3, 0) in the complex map. All pedestrians were programmed to follow cyclic motion based on predefined time schedules and coordinate sequences. The grid maps displayed in RViz are shown in Figure 9 (b) and (d), where white regions indicated traversable areas and black regions represented obstacle regions.

4.4. Comparison of global path planning simulation results

The ROS Toolbox in MATLAB provided a bidirectional interface between Simulink and ROS, enabling joint simulation with ROS [29]. In this experiment, the /odom topic was subscribed to MATLAB to display the robot's position changes in real time. A top-down screenshot of the Gazebo simulation environment was loaded into MATLAB, and the robot's motion trajectory in ROS was plotted in real time to

https://doi.org/10.1017/S0263574726103440 Published online by Cambridge University Press



(a) Simple map model in Gazebo (b) Simple raster model in Rviz (c) Complex map model in Gazebo (d) Complex raster map in Rviz

Figure 9. *Map models in a simulation environment.*

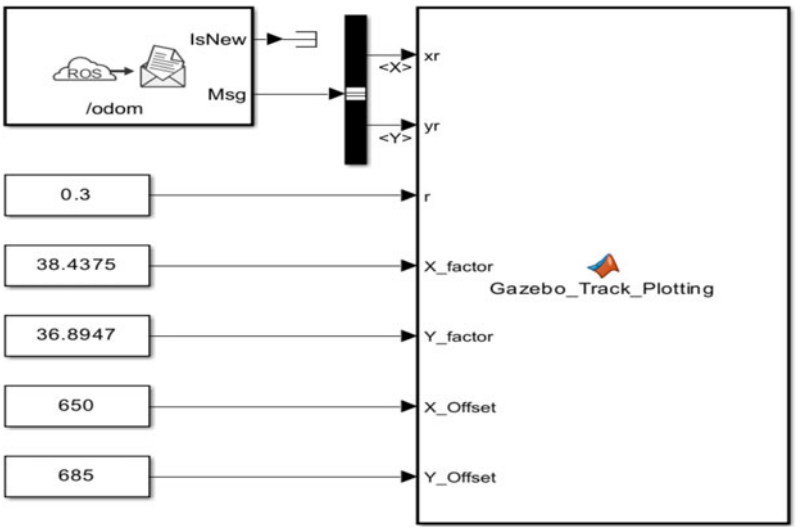


Figure 10. *Design of Simulink trajectory drawing module.*

facilitate the evaluation of the path planning performance. The relationships between the x and y axes in MATLAB and the transformations were defined as follows: w_x and w_y represented the width and height of the displayed image, and l_x and l_y represented the lengths of the map in Gazebo along the x and y axes, respectively. The relationships were given by Eqs. (4.a) and (4.b):

$$X_f = \frac{M_w}{G_y} \tag{4.a}$$

$$Y_f = \frac{M_h}{G_x} \tag{4.b}$$

The X_Offset and Y_Offset represented the offsets of the robot along the x-axis and y-axis, respectively. The Simulink module and the values of each parameter are shown in Figure 10.

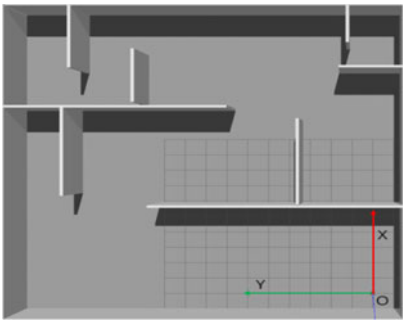
The coordinate system directions of MATLAB and Gazebo were related as shown in Figure 11. By adjusting the value of the offset, ensured that the origin positions of MATLAB and Gazebo were corresponding, and the position of the coordinate system O point was the corresponding position of the Gazebo robot.

In the ROS simulation environment, after the global planner generates a path, discrete path points were automatically extracted by subscribing to the global path topic, and path curvature and other geometric metrics were calculated based on adjacent path points. Multiple independent planning experiments were conducted in the same simulation environment. As shown in Figure 12(a) and (d), the global planning routes of traditional bidirectional A* and bidirectional dynamic heuristic algorithms are represented, respectively.

https://doi.org/10.1017/S0263574726103440 Published online by Cambridge University Press

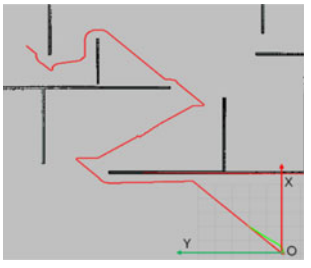


(a) MATLAB-Gazebo Frame Transform



(b) Top-Down Screenshot in MATLAB

Figure 11. *Design of Simulink trajectory drawing module.*



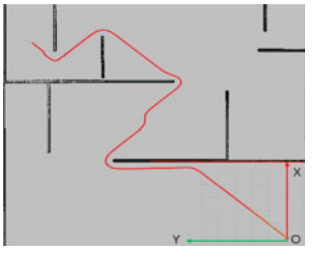
(a) Global path planned by BI-A*



(b) Actual trajectory with traditional TEB



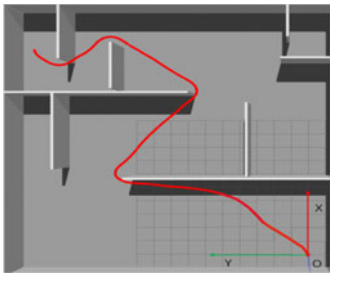
(c) Actual trajectory of the TEB



(d) Global path planned by BI- Dynamic A*



(e) Actual trajectory with traditional TEB



(f) Actual trajectory of the TEB

Figure 12. *Comparison of global and local algorithm trajectories.*

The global path was generated by the BI-Dynamic A* algorithm and output as discrete waypoints. To quantitatively analyze its geometric properties, a discrete curvature model based on three adjacent points was employed to perform a statistical analysis of the path's curvature distribution. The experimental results in Figure 12 showed that the generated path exhibits smoother curvature changes in the turning areas, providing a good initial condition for the subsequent TEB local trajectory optimization. (In this paper, the TEB planning scheme was represented as I-TEB.)

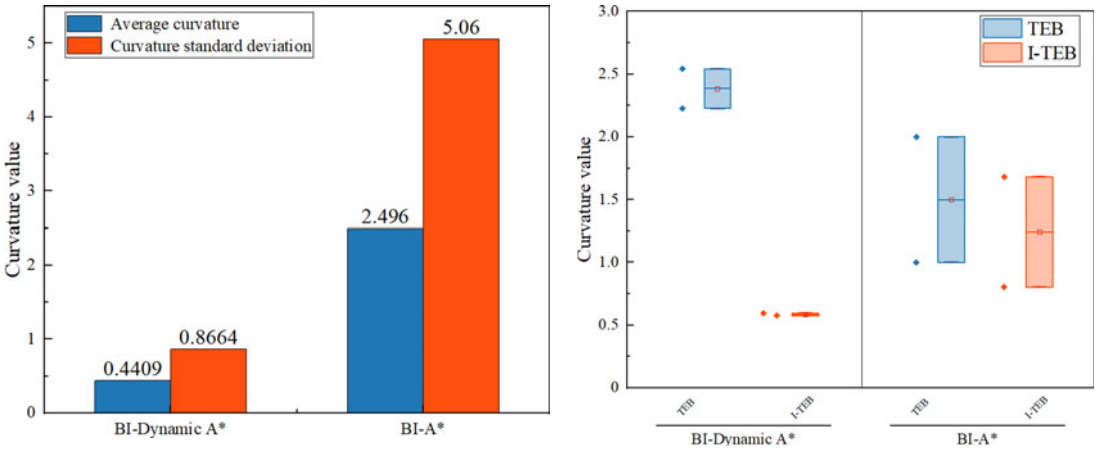
Since global path planning may be affected by path pruning and replanning mechanisms, the number of generated path points could vary at different stages. To ensure the representativeness of the statistics, the analysis was based on the first generated global path, and only paths with a number of points not less than 80% of the original path length were selected for curvature statistics. This approach reflected the overall geometric characteristics of the global path. For TEB-based local planning, the average curvature of the trajectories generated by TEB was calculated as an indicator of overall trajectory smoothness. The experimental results for the entire motion process were summarized in Table IV.

Based on the data in Table IV, Figure 13 was plotted to visually illustrate the curvature variations for different algorithm combinations.

https://doi.org/10.1017/S0263574726103440 Published online by Cambridge University Press

Table IV. *Path curvature statistics for various algorithms*

Global Planner	Local Planner	Global Curvature Mean(m^{-1})	Global Curvature Std. Dev.	Local Curvature Mean (m^{-1})	Local Curvature Std. Dev.
BI-A*	TEB	2.4960	5.0600	0.8453	2.2332
BI-A*	I-TEB	2.4960	5.0600	0.8031	1.6806
BI-Dynamic A*	TEB	0.4409	0.8664	2.5420	2.2270
BI-Dynamic A*	I-TEB	0.4409	0.8664	0.5758	0.5950



(a) Global curvature comparison

(b) local curvature comparison

Figure 13. *Mean and standard deviation of curvature for different algorithm combinations*

In the ROS simulation experiment, the curvature of the local trajectories generated by the TEB local planner during the stable motion phase was statistically analyzed.

To mitigate the impact of trajectory degradation near the goal and local optimization anomalies on the statistical results, this study only analyzed local trajectory segments that contained more than 10 waypoints and where the robot was in a stable motion state. The results show that the BI-Dynamic A* algorithm effectively reduced unnecessary twists in the global path. The generated trajectories had an average curvature of approximately 0.4409 and a curvature standard deviation of 0.8664. Additionally, the TEB planner, as compared to the traditional TEB, further suppressed the curvature fluctuations in the local trajectory. The experimental results showed that the standard deviation of curvature exceeded its mean value. This is attributed to the inherent piecewise-linear nature of the global planner and the presence of certain turning points in the path, both contributing to the increased variance in curvature values. Through the collaborative optimization of the global and local planners, the motion stability and executability of the Ackermann robot have been effectively improved.

4.5. Impact analysis based on TEB trajectory optimization

In Figure 14, the red line represents the global path planning effect displayed by the four algorithms in Rviz. At this point, the robot has not yet started moving.

Simulation data were obtained by averaging 10 experiments on the computation time and optimal path length of the three algorithms, as shown in Table V and Figure 15.

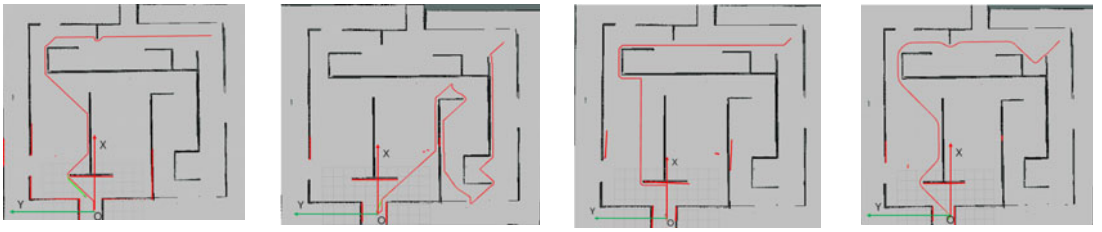
The simulation data of the experiment, as presented in Table V and Figure 14, were obtained by averaging the results of 10 trials. These trials measured the time required for generating global planning paths and the length of the optimal paths produced by the execution files of three distinct algorithms.

Based on Table V, a scatter plot as shown in Figure 15 was drawn.

https://doi.org/10.1017/S0263574726103440 Published online by Cambridge University Press

Table V. *Simulation data of different planning algorithms*

Algorithms	Traditional A*	BI- A*	Literature[26]A*	BI-Dynamic A*
Simple environment search time /(ms)	86.284	65.558	61.009	82.667
Simple environment path length /(m)	35.563	36.901	35.383	41.192
Complex environment search time /(ms)	84.033	51.583	67.212	119.693
Complex environment path length /(m)	36.397	52.514	39.6332	37.442



(a) Traditional A* (b) BI-A* (c) Literature[26]A* (d) BI-Dynamic A*

Figure 14. *Comparison of robot trajectories under simple and complex maps.*

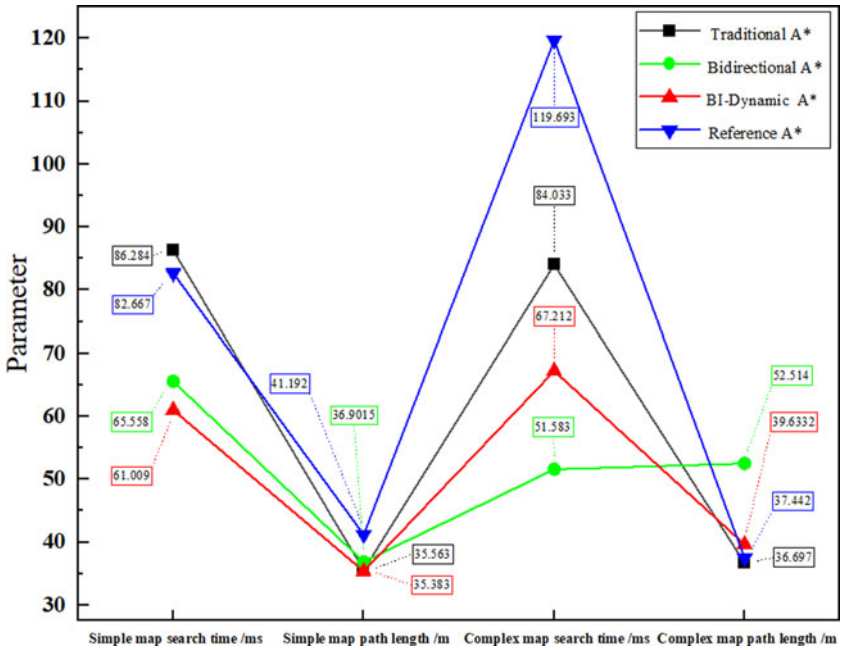
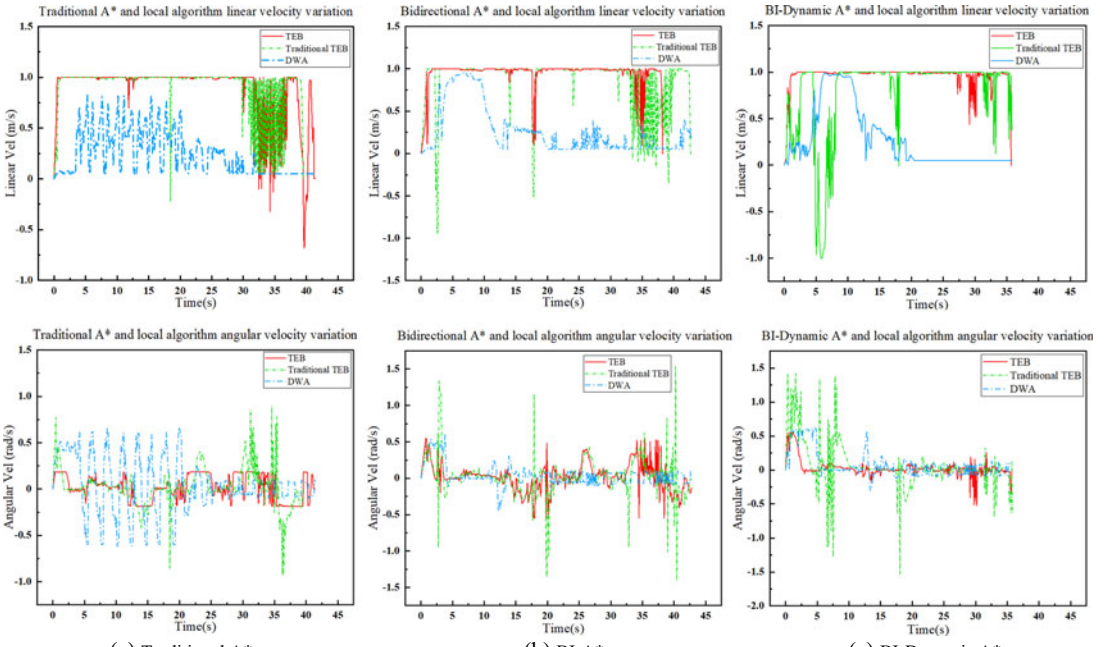


Figure 15. *Simulation data of different planning algorithms.*

From Figure 15, it was clear that the BI-Dynamic A* algorithm outperforms other algorithms in both path length and search efficiency. According to Table IV, in the simple environment group, the BI-Dynamic A* algorithm improved search time by about 29.3% compared to traditional A*, and by 6.93% and 26.19% compared to the bidirectional A* and the algorithm from Ref. [26], respectively. The optimization effect of BI-Dynamic A* in the experiment was slightly improved compared to other

https://doi.org/10.1017/S0263574726103440 Published online by Cambridge University Press



(a) Traditional A* (b) BI-A* (c) BI-Dynamic A*

Figure 16. *Changes in the linear and angular velocities of the robot under simple maps.*

algorithms due to the relative simplicity of the map environment compared to the Python-based environment. However, overall, BI-Dynamic A* performed better than bidirectional A* in terms of path length and better than traditional A* in search time, consistent with the conclusions from Python-based experiments. The path length generated by BI-Dynamic A* was improved by 4.11% and 14.10% compared to bidirectional A* and the algorithm from Ref. [26], respectively, with only a small difference compared to the traditional A* path length. In the complex environment group, the search time of BI-Dynamic A* was improved by 20.01% and 43.84% compared to traditional A* and the algorithm from Ref. [26], but it was slightly higher than bidirectional A*. The reason for this was that BI-Dynamic A* guarantees a more optimal path selection, which led to a longer search time compared to bidirectional A*. The path length generated by BI-Dynamic A* was improved by 24.52% compared to bidirectional A*, but still longer than the paths generated by traditional A* and the algorithm from Ref. [26]. This was because BI-Dynamic A* tended to generate smoother paths, leading to relatively longer paths.

To verify the differences between the TEB planning scheme, traditional TEB algorithm, and DWA algorithm, the robot's linear and angular velocities during motion were output by subscribing to the cmd_vel topic in the ROS environment. Around 40s, the TEB planning scheme group reached the target point, while the DWA algorithm has not yet arrived. The kinematic characteristic curves of the TEB planning scheme group, traditional TEB, and DWA algorithm groups were analyzed by selecting the time range when the TEB planning scheme group reached the target point. (The TEB planning scheme is represented as I-TEB.) The speed variation in both maps was shown in Figures 16 and 18.

Due to dynamic obstacles in the map, the robot's motion trajectory differs, and the robot may deviate from the global planned path. Therefore, the path tracking error index was used to evaluate the global system's stability, not just the local stability. The robot's actual trajectory was obtained from the /odom topic in ROS, while the reference path was provided by the global path topic. The minimum Euclidean distance between the robot's actual position and the global reference path was calculated as the instantaneous tracking error. The average error was then statistically computed to quantify the path's trackability.

The mean values and standard deviations of linear velocity and angular velocity under various experimental combinations on a simplified map were presented in Table VI.

https://doi.org/10.1017/S0263574726103440 Published online by Cambridge University Press

Table VI. *Different algorithms combine linear velocity and angular velocity change*

Global algorithm	**Traditional A***			**BI-A***			**BI-Dynamic A***		
Local algorithm	I-TEB	TEB	DWA	I-TEB	TEB	DWA	I-TEB	TEB	DWA
Tracking error	0.2252	0.2552	0.2765	0.2303	0.2838	0.2924	0.1920	0.2039	0.2537
Average linear vel (m/s)	0.8836	0.9098	0.2365	0.9527	0.8740	0.2702	0.9647	0.7956	0.2825
Average angular vel (rad/s)	0.0180	0.0222	0.0465	0.0186	0.0184	0.0317	0.0214	0.0580	0.0803
Standard deviation of linear vel	0.2925	0.2543	0.2164	0.1569	0.3213	0.2872	0.11922	0.4439	0.3048
Standard deviation of angular vel	0.1364	0.2106	0.3022	0.1932	0.2573	0.1541	0.1279	0.3572	0.1938

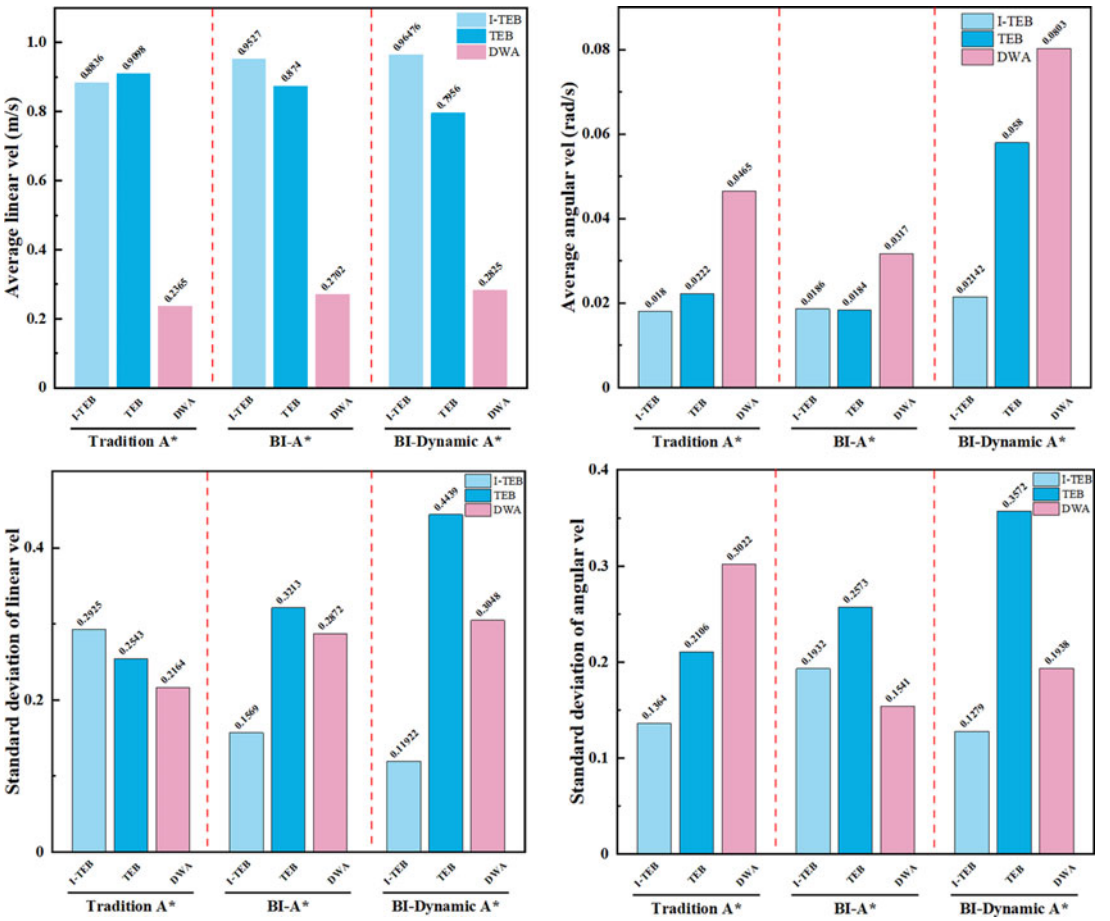
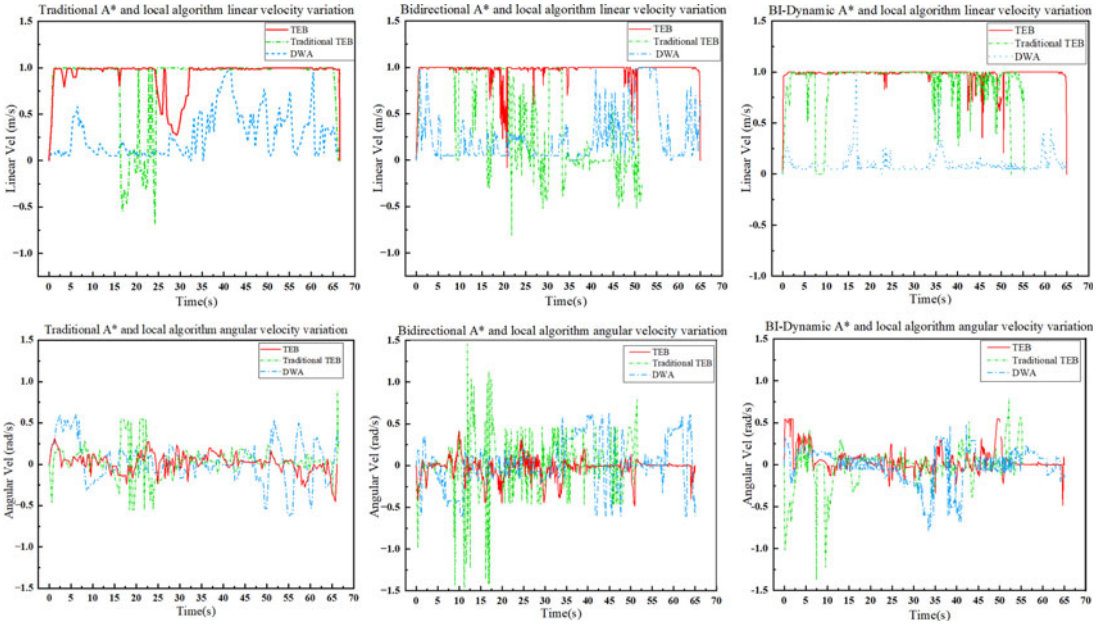


Figure 17. *Variations in linear velocity and angular velocity on a simplified map.*

The data presented in Table VI were graphically illustrated in Figure 17.

As can be observed from Figure 17, the I-TEB group exhibited a lower standard deviation of angular velocity than the other groups, indicating smaller velocity fluctuations and smoother motion. According to the results in Table VI, the average linear velocity of the I-TEB group was higher than that of the DWA algorithm. The reason for this difference was that the DWA algorithm prioritizes optimal local safety. This focus leads it to adopt a lower linear velocity, resulting in longer planning time. In contrast, the

https://doi.org/10.1017/S0263574726103440 Published online by Cambridge University Press



(a) Traditional A* (b) BI-A* (c) BI-Dynamic A*

Figure 18. *Changes in the linear and angular velocities of the robot under complex maps.*

TEB algorithm discretized the global path into multiple pose points and optimizes them. This allows it to select a higher speed while ensuring safety, thereby saving planning time. The I-TEB algorithm achieved efficient linear motion, with an average angular velocity of approximately 0.02 rad/s, indicating smooth steering behavior. Although the traditional TEB algorithm reached a linear velocity of 0.9 m/s in the conventional A* group, its angular velocity standard deviation was as high as 0.21, leading to significant fluctuations and unstable robot motion.

The path tracking error for the BI-Dynamic A* algorithm was slightly lower than that of traditional A* and bidirectional A*, primarily because BI-Dynamic A* emphasizes path smoothness, which results in better curvature variation and geometric continuity. This made the planned path more suitable for vehicle kinematics. During dynamic obstacle avoidance, the robot frequently adjusted its global path, which led to some deviation between the actual motion trajectory and the global path. However, in the local planning algorithm group, the TEB planning scheme outperformed both traditional TEB and DWA, demonstrating better local planning performance.

The variations of linear velocity and angular velocity of the mobile robot under different algorithm combinations in complex maps were shown in Table VII.

Based on the data presented in Table VII, the corresponding graph was illustrated in Figure 19.

From Figure 19, it could be visually observed that in the complex map, the I-TEB group had a lower standard deviation of angular velocity compared to other groups, indicating smaller velocity fluctuations and smoother motion. According to Table VII, in the BI-Dynamic A* group, the DWA group had an average linear velocity of only 0.0987 m/s. This was because the DWA algorithm tends to lower the linear velocity and frequently adjusted the robot's orientation in the velocity space to ensure local safety, which led to a reduction in overall motion efficiency. This phenomenon suggested that when DWA was coupled with high-quality geometric paths, it was difficult to fully leverage the advantages of global planning. The TEB algorithm performed well under most global algorithms, with efficient linear velocity and an average angular velocity of about 0.02 rad/s, resulting in smooth steering. In contrast, the traditional TEB algorithm in the traditional A* group had a linear velocity of 0.845 m/s but a high linear velocity standard deviation of 0.3851, causing large fluctuations and leading to unstable robot motion. In terms of path tracking error, the TEB-based groups generally exhibited lower tracking errors. On the other

https://doi.org/10.1017/S0263574726103440 Published online by Cambridge University Press

Table VII. *Different algorithms combine linear velocity and angular velocity change*

Global algorithm	**Traditional A***			**BI- A***			**BI-Dynamic A***		
Local algorithm	I-TEB	TEB	DWA	I-TEB	TEB	DWA	I-TEB	TEB	DWA
Tracking error	0.2273	0.2395	0.2892	0.2364	0.2529	0.2924	0.2030	0.234	0.2872
Average linear vel (m/s)	0.9230	0.8450	0.2705	0.9669	0.4241	0.2625	0.9714	0.9261	0.0987
Average angular vel (rad/s)	0.0137	0.0449	−0.0065	−0.0246	0.0021	0.0249	0.0459	−0.0052	−0.0101
Standard deviation of linear vel	0.1894	0.3851	0.2559	0.1238	0.5106	0.2841	0.1139	0.1788	0.1018
Standard deviation of angular vel	0.1191	0.2097	0.2804	0.1196	0.3559	0.2885	0.1438	0.2177	0.2097

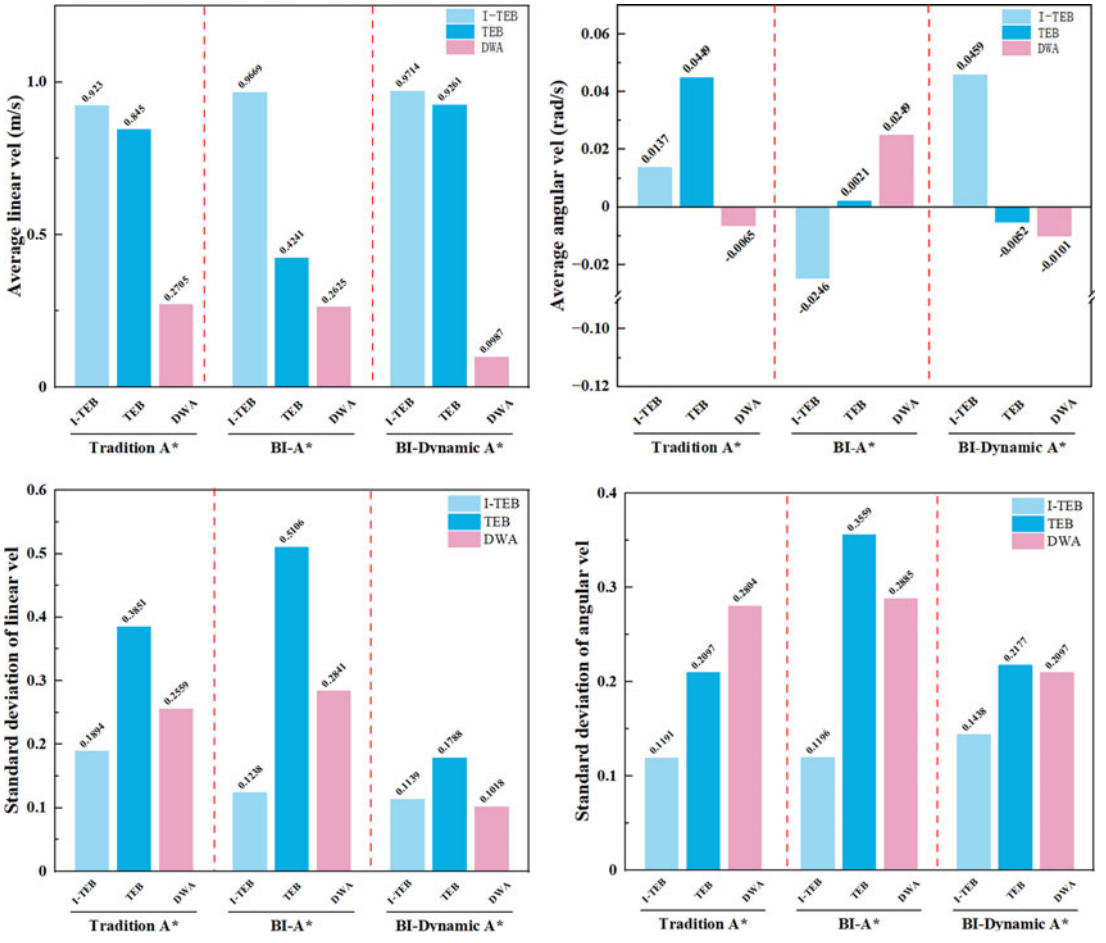
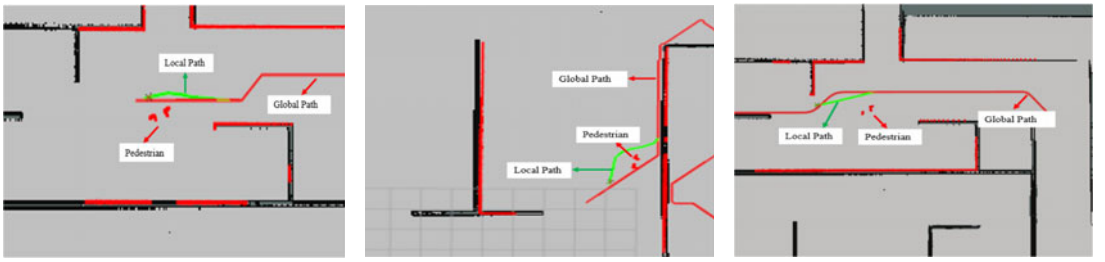


Figure 19. *Changes in linear velocity and angular velocity under complex maps.*

hand, the DWA group, due to frequent orientation adjustments, showed larger velocity fluctuations and an increase in path tracking error.

These results demonstrate that introducing a minimum turning radius constraint, together with high-quality global path geometric properties, was a key factor in reducing path tracking error and enhancing the motion stability of Ackermann-steered robots.

https://doi.org/10.1017/S0263574726103440 Published online by Cambridge University Press



(a) Traditional A* (b) BI-A* (c) BI-Dynamic A*

Figure 20. *Local trajectories in edge-case scenarios.*

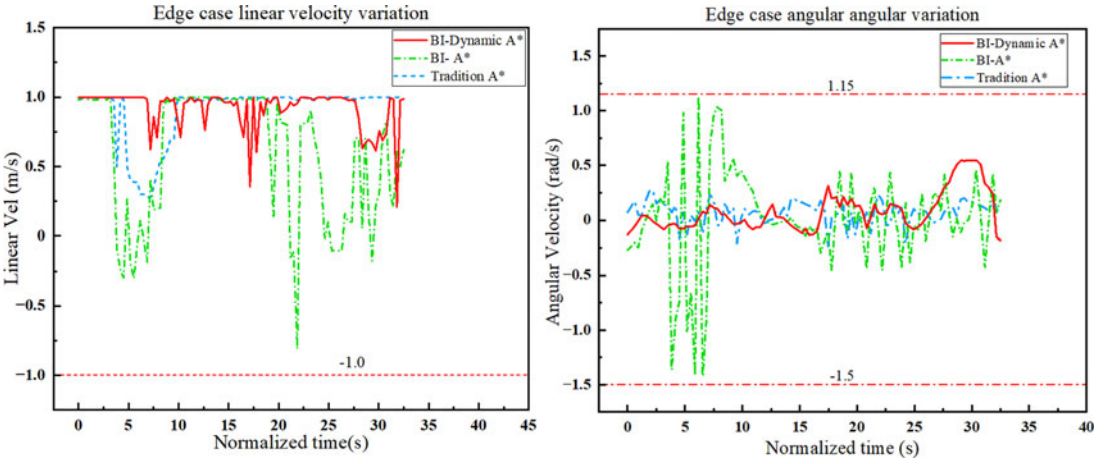
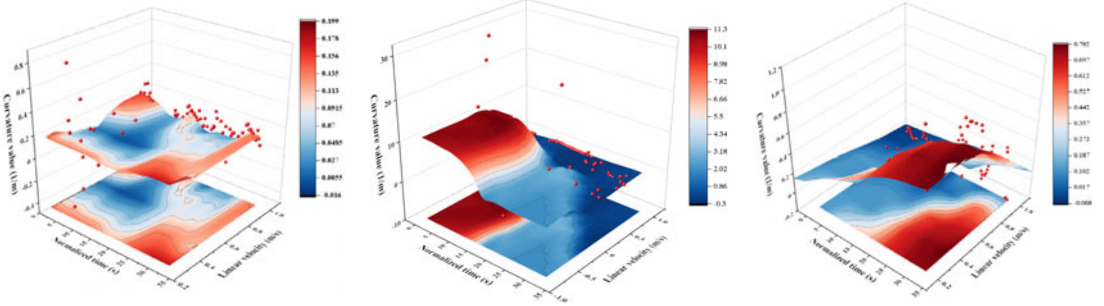


Figure 21. *Edge-case velocity curves.*



(a) Traditional A* (b) BI-A* (c) BI-Dynamic A*

Figure 22. *3D visualization of curvature constraints in edge cases.*

4.6. Trajectory analysis of edge cases in complex maps

To validate the effectiveness of the proposed method, this section analyzed edge cases in complex maps where the robot encountered dynamic pedestrians and sharp turns, which may cause abrupt changes in linear and angular velocities. The variations in linear velocity, angular velocity, and curvature under different algorithm combinations in these edge scenarios were comparatively analyzed, as illustrated in Figure 20.

Based on the overall variation trends in Figure 18, the velocity and angular velocity curves for edge-case scenarios were illustrated in Figure 21, while Figure 22 presents the three-dimensional visualization of the curvature constraints.

https://doi.org/10.1017/S0263574726103440 Published online by Cambridge University Press



(a) Robotic Side View (b) Front Elevation of the Robotic System

Figure 23. *Aerospace light boat robot model diagram.*

From the results in Figure 21, it could be clearly observed that, under edge-case scenarios, the linear velocity mainly remained within 1.0 m/s, while the angular velocity reached a maximum of about 1.15 rad/s and a minimum of approximately −1.5 rad/s. Overall, the motion profiles satisfied the predefined robot constraints.

Further analysis with the three-dimensional surface plot in Figure 22 revealed that the path generated by traditional A* had good smoothness but was limited in search efficiency. Traditional bidirectional A* improved efficiency, but the curvature fluctuated significantly. In contrast, the BI-Dynamic A* algorithm ensured both search efficiency and smoother curvature changed over time and linear velocity. The curvature variation in the time–velocity plane was more gradual, leading to a more continuous curvature with the robot's movement and reflecting better path quality. Compared to bidirectional A*, the curvature in the traditional A* group changed more smoothly over time and velocity, with overall fluctuations being smaller. On the other hand, the bidirectional A* group showed more drastic curvature changes, with noticeable high-curvature areas, indicating poorer path smoothness.

4.7. Real environment experiments

Due to the idealized radar distance measurement errors in the simulation environment, there were some differences when applied to a real-world environment. This study evaluated the obstacle avoidance capability of the path planning scheme in a real environment. The robot used was the Qingzhou YQ01, an Ackermann model, equipped with a multi-sensor fusion system. This mobile robot was equipped with a Jeton NANO controller and LiDAR, enabling dynamic path planning and obstacle avoidance capabilities. The mobile robot model diagram and performance parameter table were shown in Figure 23 and Table VIII.

By utilizing the visualization tool Rviz, the target point for the robot was established, and the robot's obstacle avoidance performance was observed, as illustrated in Figure 24: the global planning route was indicated by the red line, while the local planning route was denoted by the green line.

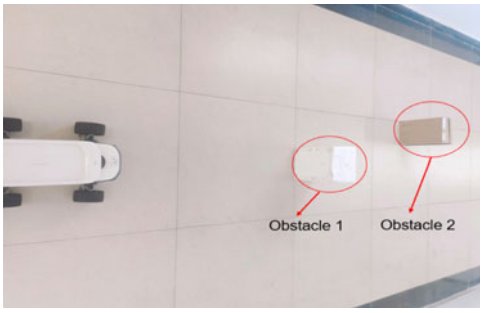
5. Conclusion

This paper presented an engineering solution for global–local path planning tailored to Ackermann mobile robots, achieving executable and stable motion through the coordination of bidirectional dynamic heuristic global planning and the TEB local planner with a minimum turning radius constraint. Python simulation experiments showed that the bidirectional dynamic heuristic global planning effectively generates shorter, smoother paths, improving search efficiency and reducing node count compared to

https://doi.org/10.1017/S0263574726103440 Published online by Cambridge University Press

Table VIII. *Different Robot performance parameter table*

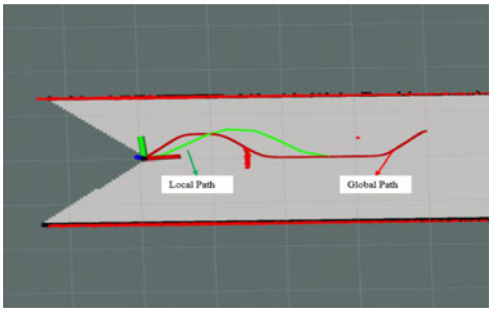
Parameter	Parameter value
Dimensions (L×W×H)	505 mm × 350 mm × 165 mm
Total weight	Approximately 5 kg
Maximum linear vel	0.22 m/s
Maximum angular vel	2.84 rad/s(162.72 deg/s)



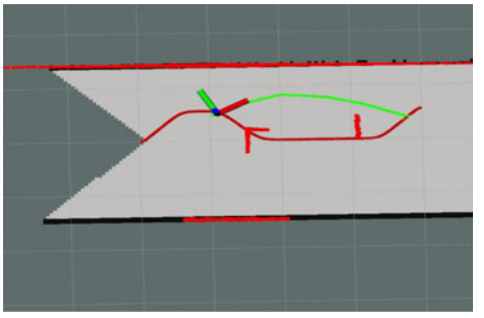
(a) Experimental environment diagram



(b) Schematic diagram of robot obstacle avoidance moment



(c) The route planned by the robot



(d) The obstacle avoidance effect of the robot

Figure 24. *Obstacle avoidance process of mobile robots in real environments.*

traditional A* and other referenced algorithms. ROS simulations with traditional TEB and DWA local planners indicated that the TEB planner with the turning radius constraint outperformed other schemes in velocity and angular velocity stability, enhancing the robot's motion stability and executability in complex environments. MATLAB-ROS co-simulation further confirmed that the proposed scheme produced smoother trajectories in simpler map environments.

Currently, due to limitations in the scale of the experimental platform and environmental complexity, this study only qualitatively verified the algorithm's executability and didn't perform a systematic evaluation of metrics like collision rate. Moreover, scenarios like turning in narrow spaces and real-world dynamic obstacle avoidance were not tested. Future work may proceed along two directions: first, to validate the engineering applicability of the proposed path planning scheme in more complex and dynamic scenarios; second, to conduct tests integrated with the robotic hardware platform, so as to quantitatively evaluate the impact of perception and control delays as well as environmental factors on the overall system performance. Furthermore, adaptive methods could be introduced to optimize the global planning heuristic strategy, enhancing the scheme's practical use in diverse engineering scenarios. Overall, this research provided a feasible global–local planning solution for real-world Ackermann robots, offering valuable insighted into path planning and trajectory execution in complex environments.

https://doi.org/10.1017/S0263574726103440 Published online by Cambridge University Press

Supplementary material. The supplementary material for this article can be found at https://doi.org/10.1017/S0263574726103440.

Author contribution. Conceptualization: Z.W. and X.W.; methodology: X.W.; software: J.H. and Y.C.; formal analysis: X.W.; validation: J.H.; writing – original draft preparation: Z.W.; writing – review and editing: X.W and J.H.; data curation: Y.C.;visualization: Y.C.; funding acquisition: X.W. All authors have read and agreed to the published version of the manuscript.

Funding statement. This research was supported by the Science and Technology Department of Shaanxi Province under the Shaanxi Key Research and Development Program, Grant No. 2025CY-JJQ-20.

Competing interests. The authors declare no competing interests.

References

[1] B. B. K. Ayawli, J. K. Dawson, E. Badu, I. E. B. Ayawli and D. Lamusah, "Comparative analysis of popular mobile robot roadmap path-planning methods," *Robotica*. **43**(4), 1548–1571 (2025). doi: 10.1017/S0263574725000281.

[2] W. Gong, L. Wang, J. Zhang, F. Wang and K. Guo, "Topological discrimination and lazy collision-based multi-stage optimization of probabilistic roadmap algorithm for path planning," *Robotica*. **43**(5), 1885–1909 (2025). doi: 10.1017/S0263574725000529.

[3] S. Qi, H. Lu, Z. Wang, X. Gao, S. Chen, Y. Liu, et al., "A novel path planning algorithm based on synergistic bidirectional optimization," *Robotica*. **44**(1), 52–70 (2026). doi: 10.1017/S0263574725103093.

[4] A. Bhargava, M. Suhaib and A. K. S. Singholi, "A hybrid optimization algorithm combining A* and the dynamic window approach for automated guided vehicle control in static and dynamic environments," *Robotica.* **44**(2), 408–470 (2026). doi: 10.1017/S0263574725102944.

[5] 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.

[6] Y. Dai, W. Lv, S. Li and M. Zong, "Improving the lifelong planning A-star algorithm to satisfy path planning for space truss cellular robots with dynamic obstacles," *Robotica*. **43**(4), 1243–1257 (2025). doi: 10.1017/S0263574725000256.

[7] Y. Yang, X. Luo, W. Li, H. Liu and Y. Li, "AAPF: A safer autonomous vehicle path planning algorithm based on the improved A algorithm and APF algorithm," *Cluster Comput*. **27**, 11393–11406 (2024).

[8] T. Wang, L. Yang, C. Huang, Y. Wang and S. Zeng, "Path planning solution based on topology map and A* algorithm," *Appl. Res. Comput*. **42**(9), 2714–2721 (2025). (in Chinese). doi: 10.19734/j.issn.1001-3695.2025.01.0019.

[9] C. Li, L. Yao and C. Mi, "Fusion algorithm based on improved A* and DWA for USV path planning," *J. Mar. Sci. Appl.* **24**(1), 224–237 (2025). doi: 10.1007/s11804-024-00434-1.

[10] C. Li, X. Zhang, Y. Watanabe and Y. Ohtsuka, "Autonomous collision avoidance of unmanned surface vehicles based on improved A star and minimum course alteration algorithms," *Appl. Ocean Res*. **113**, 102732 (2021).

[11] X. Yu, C. Jiang, S. Duan, H. Wang and Y. Xu, "Path planning for improvement of A* algorithm and artificial potential field method," *J. Syst. Simul*. **36**(3), 782–794 (2024). (in Chinese). doi: 10.16182/j.issn1004731x.joss.23-0255.

[12] X. Wu, L. Xu, R. Zhen and C. G. Soares, "Bi-directional adaptive A* algorithm toward optimal path planning for large-scale UAV under multi-constraints," *IEEE Access*. **8**, 85431–85440 (2020).

[13] M. Chen, G. Qiao, X. Zou, J. Cao and L. Tian, "Research on path smoothing processing of mobile robot based on improved A* algorithm," *J. Syst. Simul.* **37**(11), 2804–2811 (2025). (in Chinese). doi: 10.16182/j.issn1004731x.joss.24-0693.

[14] Z. Lin, K. Wu, R. Shen, X. Ji and Y. Zhang, "An efficient and accurate A-star algorithm for autonomous vehicle path planning," *IEEE Trans. Veh. Technol*. **73**(6), 9003–9008 (2023).

[15] G. Tang, C. Tang, C. Claramunt, X. Hu and P. Zhou, "Geometric A-star algorithm: An improved A-star algorithm for AGV path planning in a port environment," *IEEE Access*. **9**, 59196–59207 (2021).

[16] C. Chen, Y. Lin, L. Zhan, X. Zeng and J. Chen, "Dynamic obstacle avoidance approach based on integration of A-star and APF algorithms for vehicles in complex mountainous environments," *Vehicles*. **7**(3), 65 (2025).

[17] A. Bhargava, M. Suhaib and A. K. S. Singholi, "An omnidirectional mecanum wheel automated guided vehicle control using hybrid modified A* algorithm," *Robotica*. **43**(2), 449–498 (2025). doi: 10.1017/S0263574724001954.

[18] J. Li, F. Zhang and Y. Fu, "A novel human-aware navigation algorithm based on behavioral intention cognition," *Robotica*. **42**(3), 864–890 (2024). doi: 10.1017/S0263574723001832.

[19] H. Houshyari and V. Sezer, "A new gap-based obstacle avoidance approach: Follow the obstacle circle method," *Robotica*. **40**(7), 2231–2254 (2022). doi: 10.1017/S0263574721001624.

[20] L. Chen, R. Liu, D. Jia, S. Xian and G. Ma, "Improvement of the teb algorithm for local path planning of car-like mobile robots based on fuzzy logic control," *Actuators*. **14**(1), 12 (2025). doi: 10.20944/preprints202412.0483.v1.

[21] X. Yin, W. Dong, X. Wang, Y. Liu and Y. Zhang, "Route planning of mobile robot based on improved RRT star and TEB algorithm," *Sci. Rep*. **14**(1), 8942 (2024).

[22] H. Wang, "An improved timed elastic band (TEB) algorithm of autonomous ground vehicle (AGV) in complex environment," *Sensors* **21**(24), 8312 (2021).

https://doi.org/10.1017/S0263574726103440 Published online by Cambridge University Press

[23] J. Wu, X. Ma, T. Peng and H. Wang, "An improved timed elastic band (TEB) algorithm of autonomous ground vehicle (AGV) in complex environment," *Sensors* **21**(24), 8312 (2021).
[24] H. Wang, K. Chen, L. He and J. Wang, "Path planning of robotic fish by combing improved A* algorithm and artificial potential field method, electron," *Meas. Technol*. **48**(13), 58–72 (2025). (in Chinese). doi: 10.19651/j.cnki.emt.2517856.
[25] P. Gui, Y. Jin, G. Liao and W. Zhu, "Ackerman mobile robots based on improved TEB algorithm," *J. China Univ. Metrol*. **35**(3), 432–442 (2024). (in Chinese). doi: 10.3969/j.issn.2096-2835.2024.03.009.
[26] K. Kang and H. Huang, "Research on path planning in complex environment based on improved A* algorithm," *J. Mech. Transm*. **49**(7), 22–30 (2025). (in Chinese). doi: 10.16578/j.issn.1004.2539.2025.07.003.
[27] H. Wang, C. Gan and F. Wang, "Improved A* algorithm for unmanned vehicle path planning, mach," *Des. Manuf*. **1**, 117–120 (2025). (in Chinese). doi: 10.19356/j.cnki.1001-3997.20240725.026.
[28] Y. Guo, D. Wang, X. Wu and S. Sun, "Research on path planning of inspection robot based on improved bidirectional A* algorithm, mach," *Build. Autom*. **54**(2), 233–237 (253, 2025). (in Chinese). doi: 10.19344/j.cnki.issn1671-5276.2025.02.045.
[29] T. Zhang, X. Qi, H. Zhang and Z. Li, "Research on mobile robot simulation based on ROS-Matlab, Mod," *Electron. Tech*. **47**(18), 114–120 (2024). (in Chinese). doi: 10.16652/j.issn.1004-373x.2024.18.019.

Cite this article: Z. Wan, X. Wang, J. Han and C. Yang (2026). "Path planning for Ackermann-steered robots based on bidirectional A* and timed elastic band algorithms", Robotica **44**, 1096–1121. https://doi.org/10.1017/S0263574726103440

https://doi.org/10.1017/S0263574726103440 Published online by Cambridge University Press