Alexandria Engineering Journal 88 (2024) 91–104

Contents lists available at ScienceDirect

Alexandria Engineering Journal

journal homepage: www.elsevier.com/locate/aej

Intelligent path planning by an improved RRT algorithm with dual grid map

Rui Zhang [a], He Guo [a,*], Darius Andriukaitis [b], Yongbo Li [c], Grzegorz Królczyk [d], Zhixiong Li [d,*]

[a] School of Automobile and Transportation, Tianjin University of Technology and Education, Tianjin 300222, China
[b] Department of Electronics Engineering, Faculty of Electrical and Electronics Engineering, Kaunas University of Technology, 44249 Kaunas, Lithuania
[c] School of Aeronautics, Northwestern Polytechnical University, Xi'an, China
[d] Faculty of Mechanical Engineering, Opole University of Technology, 45-758 Opole, Poland

ARTICLE INFO

Keywords:
Bidirectional RRT
Grid map
A-star algorithm
Obstacle avoidance algorithm
Bezier curve

ABSTRACT

This research addresses the limitations of existing autonomous vehicle path planning algorithms, notably their slow processing speeds and suboptimal route efficiency. We introduce an innovative path planning algorithm that synergizes the A* algorithm with the Rapidly-exploring Random Tree (RRT) approach. This hybrid model significantly enhances route timeliness and reliability, particularly in obstacle avoidance scenarios for driverless vehicles. Our methodology employs a 'two-level map' approach, where a lower-resolution grid map is derived from a high-resolution map. Utilizing the A* algorithm on this framework, we ascertain a preliminary 'coarse path' for the navigation target. The RRT algorithm, modified to reduce the traditional redundancy associated with random uniform sampling, is then applied for probabilistic sampling within this defined area. ovel aspect of our approach is the simultaneous generation of two trees, originating from both the start and end points, guided by a target-biased strategy and dual-direction theory. This method probabilistically expands towards the node of the opposite tree, thereby enhancing both the generation speed and trajectory viability. Further refinements are made through a pruning process, optimizing the path, and employing Bezier curves for smoothing, ensuring compliance with the dynamic constraints of Ackerman chassis vehicles. Comparative analysis in complex environments demonstrates the superiority of our proposed algorithm. It outperforms traditional methods with a 400 % increase in planning speed relative to the RRT-Connect algorithm, and a 30 % reduction in average path length. Additionally, the mean curvature of routes generated by our algorithm is 19 % lower than traditional routes, underscoring significant advancements in both the timeliness and viability of the planned routes.

1. Introduction

Path planning problems are pivotal in determining the level of intelligence in autonomous vehicles. The fundamental requirement of path planning is to swiftly identify a feasible and collision-avoiding route that allows an unmanned vehicle to smoothly transition from its starting area to the target point. Path planning issues are not limited to the transportation sector; with the advancement and development of robotics technology, these problems also have widespread practical applications and demands in industrial production, agriculture, security, and safety rescue domains [1–3]. Common path planning algorithms can generally be summarized into three categories: (3) graph-based algorithms, such as the Dijkstra [4] and A* algorithms [5]; (2) bionics-based algorithms, like ant colony [6] and neural network algorithms; and sampling-based algorithms such as the RRT (Rapidly-exploring Random Tree) [7] and PRM (Probabilistic Roadmap) algorithms [8].

In the realm of sampling-based algorithms, the Rapidly-exploring Random Tree (RRT) stands out as a highly favored and prevalent planning methodology. Distinguished by its probabilistic completeness, the RRT algorithm offers a significant advantage over other planning methods in that it does not depend on pre-existing map data. Nonetheless, this algorithm is not without its challenges. It has been observed to exhibit certain limitations, including a relatively slow planning speed and the generation of routes that do not conform to the kinematic constraints of traditional vehicles. These issues highlight areas for potential refinement and optimization in the algorithm's application.

Many scholars have made improvements to the problem of RRT algorithm. The RRT-connect algorithm, an eminent enhancement of the

* Corresponding authors.
E-mail addresses: 936515285@qq.com (H. Guo), zhixiongli@cumt.edu.cn (Z. Li).

https://doi.org/10.1016/j.aej.2023.12.044
Received 12 October 2023; Received in revised form 23 November 2023; Accepted 21 December 2023
Available online 13 January 2024

RRT (Rapidly-exploring Random Trees) algorithm, was introduced by Kuffner et al. [9]. This innovative approach first utilized a dual-tree strategy to improve the RRT algorithm's efficacy in linking the initial and target states. However, its generation speed and path quality in complex environments are still limited. Karaman et al. [10] proposed an advanced version of the RRT* algorithm, whose essence lies in dynamically optimizing tree connections to enhance path quality, gradually converging to the optimum with algorithm iterations. Yet, this iterative approach significantly diminishes the RRT algorithm's rapid characteristic, and its high computational cost renders it impractical for real-time planning. Li et al. [11] addressed the excessive randomness of the RRT-connect algorithm by introducing a goal-biased strategy to the dual-tree structure of RRT-connect, thereby constraining the randomness of RRT growth and enhancing planning quality. However, this method intensifies computational load in complex maps, manifesting as prolonged algorithm execution time. Kun Hao, Yang et al. [12] proposed a method named Complex Environments Rapidly-exploring Random Tree, incorporating pre-allocated expansion nodes and vertex death mechanisms to surmount traditional algorithmic predicaments, preventing entrapment in uneven terrains. Nonetheless, the generated paths did not account for vehicle dynamics, resulting in non-smooth trajectories, thus impeding practical application. Liu et al. [13] developed a TO-RRT algorithm for robotic arm pick-and-place scenarios, introducing potential fields and attractive step lengths, allowing scenario-specific step length adjustments for rapid algorithm convergence. However, tailored for low-speed scenarios, this method also neglected path feasibility, leading to abrupt curvature changes. Zhang et al. [14] improved RRT's capability to navigate narrow areas and enhance speed in open spaces by incorporating a target attraction force. However, its applicability is limited and may encounter unreachable target issues. Similar concepts were proposed by other scholars; Qureshi et al. [15] introduced a P-RRT* variant, implementing artificial potential fields to enhance the RRT* algorithm. Jeong et al. [16] devised Quick-RRT, accelerating convergence by expanding potential parent vertex sets and refining reconnection techniques. Qie et al. [17] optimized sampling and tree growth mechanisms, combining UAV piloting modes and dynamics for efficient path planning, though such dynamics improvements are not universally applicable to most robots. Lathrop et al. [18] proposed a path planning algorithm based on Wasserstein metrics, demonstrating excellent performance in safety and confidence boundary considerations, albeit increasing computational burden. Gong et al. [19] designed a dual RRT optimization algorithm inspired by RRT-connect and RRT* to address formation shape generation issues but did not resolve high runtime concerns. Wu et al. [20] introduced Fast-RRT, incorporating a Fast-Sampling strategy to enhance search speed and stability significantly. Wang et al. [21] synthesized previous improvements, including goal-biased sampling and adaptive step size strategies, making notable progress in operational speed. Wang et al. [22] tailored the RRT algorithm for underground mine vehicles, employing vectorized maps to constrain kinematics, further improved by dynamic step sizes and steering angle constraints. Finally, Guo et al. [23] integrated a flight cost function into the RRT algorithm, constraining node expansion to meet safety requirements in flight.

The recent trend of integrating various algorithms to leverage their strengths and mitigate their weaknesses has increasingly become a focal point of scholarly attention. Farzad et al. [24] enhanced the RRT algorithm by amalgamating three renowned metaheuristic algorithms. This integration effectively utilized the characteristics of sampling methods and metaheuristic algorithms, resulting in faster route speeds and improved convergence times compared to the RRT* algorithm. However, the effectiveness of the metaheuristic algorithms depends on parameter tuning, making this improvement less robust and thus not the optimal strategy. Li et al. [25] proposed the PQ-RRT algorithm, a blend of the previously mentioned P-RRT and Q-RRT algorithms. This approach accelerates convergence while considering artificial potential fields but may also encounter local minima issues. Yang et al. [26] incorporated ant colony optimization into the expansion process of the random tree, progressively optimizing the planned path. Chen et al. [27] tackled semi-structured road autonomous vehicle problems by introducing a potential field-based RRT planner, enhancing the algorithm's obstacle avoidance capabilities. The subsequent application of a Dijkstra optimizer balanced efficiency and precision, though the algorithm has certain limitations in applicability.

Zammit [28] discussed the performance differences between the A* and RRT algorithms in unmanned aerial vehicle path planning, highlighting their suitability for different scenarios. The A* algorithm surpasses the RRT in both path length and computation time, with scenario complexity affecting the RRT's efficiency. Ben and Kwame [29] developed the ORRT-A* algorithm, which leverages the A* algorithm to identify the shortest path in the RRT and employs strategies like goal bias and spline interpolation for path smoothing.

Most of these improvements primarily focus on enhancing the RRT's expansion directions, neglecting the impact of the sampling domain's size on the RRT algorithm's speed. REZA MASHAYEKHI and associates [30] introduced the Informed RRT* -Connect algorithm, advocating for the inefficiency of searching the entire space and suggesting an elliptical growth space for exploration, significantly enhancing the algorithm's convergence efficiency. However, this elliptical growth space has limited improvements on complex maps.

Drawing on the research experience of other scholars, this article proposes a path planning algorithm that combines A* algorithm and RRT algorithm. This algorithm adopts the dual map concept, first establishing two maps based on the real world data collected by sensors, with different resolutions. Then, on a rough map with lower resolution, the A-star algorithm is used for rapid planning to generate a guidance domain, limiting the sampling domain of the RRT algorithm to the guidance domain to accelerate path generation speed. At the same time, a dual tree strategy is adopted to generate a search tree at both the starting and ending points, and with a certain probability, one tree is directed towards the latest child node of the other tree, further improving the speed and availability of path generation. After generating the path, use pruning optimization to remove redundant points on the path, making it as concise as possible. Finally, use the Bezier curve for secondary optimization of the concise path to improve the smoothness of the planned route. The contributions are given as follows.

(1) Weighted fusion in high-resolution grid maps is employed to extract lower-resolution grid maps. The A* algorithm is then used for coarse path planning in these low-resolution maps. The region planned by the A* algorithm in the low-resolution map is projected onto the high-resolution map, which then serves as the designated planning area for the RRT algorithm.
(2) To accelerate the convergence speed of the RRT algorithm within the defined restricted area, an improved biased tree strategy is proposed, based on the target bias strategy and dual-tree growth tactics. This ensures both the efficiency of the algorithm and the usability of the trajectory.
(3) For optimizing the generated paths, a pruning method is used, followed by smoothing with bezier This process ensures that the path complies with the dynamic constraints of vehicles with Ackermann steering geometry.

The remainder of this paper is structured as follows. Section 2 introduces the relevant prerequisites pertinent to the algorithm discussed in this paper. Section 3 elucidates the proposed algorithm, providing an analysis of its principles. Section 4 presents simulation results, conducting ablation experiments and comparing them with traditional methods and research conducted by other scholars. Finally, Section 5 summarizes the main contributions of this study and discusses potential future research directions.

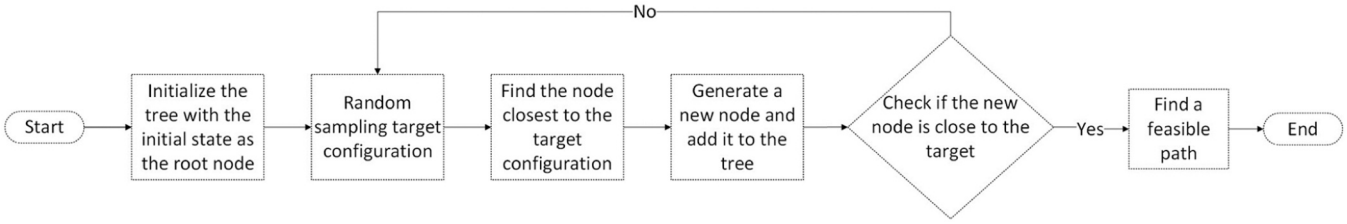

Fig. 1. the execution steps of RRT algorithm.

2. Background

2.1. Problem description

Similar to the approaches of other scholars [23,25,30], our research begins with a detailed elucidation of the definition of path planning. The essence of path planning is to identify the optimal route from a starting point to a destination within a given environment. Let the environment be denoted as S, represented by a set of coordinate points P. Within this environment, there exist obstacles $X_{obs} \subseteq P$, which are the spatial segments that cannot be traversed. The remaining area, defined as the feasible space $X_{free} \subseteq (P \backslash X_{obs})$, represents the traversable sections. We consider a starting point $S \in P$ and a destination $T \in P$. The planned path σ is expressed as an ordered sequence of points from the starting point S to the destination T, i.e., $\sigma = \{p1, p2, \ldots, pn\}$, where $p1 = S$ and $pn = T$. The path must adhere to constraint C. For instance, in autonomous vehicle path planning, the path should comply with vehicle dynamics, ensuring that the angle formed by any three points does not exceed the vehicle's maximum steering angle θ. Let α_i represent the angle formed by the points P_i, P_{i+1}, P_{i+2}, thus $\alpha_i < \theta$, for all $i \in \{1, 2, \ldots, n-2\}$, where $\alpha_i = \arccos\left(\frac{P_i \to P_{i+1} \cdot P_{i+1} \to P_{i+2}}{|P_i \to P_{i+1}||P_{i+1} P_{i+2}|}\right)$. The objective function, defined as the minimum distance $f(\sigma) = \sum_{i=1}^{n-1} distance(P_i, P_{i+1})$ is also established.

2.2. Related work

2.2.1. RRT algorithm

Rapidly-exploring Random Tree (RRT) were first proposed by Steven M. LaValle in 1998. The RRT algorithm is a path planning method based on random sampling, designed specifically to solve robot motion planning problems under high-dimensional and nonlinear constraints. Its main advantages are simplicity, efficiency, and the ability to find feasible solutions in complex environments.

$$x_{new} = x_{near} + s \frac{x_{rand} - x_{near}}{\| x_{rand} - x_{near} \|} 2 - 1$$

The generation process of new nodes can refer to Equation 2–1. Among them, x_ New represents a new node, x_ Near represents the closest tree node, while x_ Rand is a random point. The execution steps of the RRT algorithm are shown in Fig. 1, and Pseudocode in algorithm1.

Algorithm 1. : RRT.

Algorithm 1: RRT

Input: *region,Xstart,Xgoal*
return: *path V*

```
1   V.init()
2   for i=1 to n do
3     xrand ←Sample(region)
4     xrand ←Near(region, V)
5     xnew ←Steer(xrand, xnear, Step)
6     Ei←Edge(Xnew, Xnear)
7     if CollisionFree(Ei) then
8       V.addNode(xnew)
9       V.addEdge(Ei)
10    if Xnew=Xgoal then
11      return V
```

.

The efficacy of path planning is exemplified in Fig. 2, which illustrates the trajectory from the origin at coordinates {5, 35} to the destination at {190, 70}. An analysis of the path planning outcomes, utilizing the Rapidly-exploring Random Tree algorithm for autonomous vehicle navigation, reveals several limitations:

1. Curvature Variability: The planned route exhibits abrupt changes in curvature. Occasionally, this manifests as acute angles between successive segments, which are incongruent with the requirements of autonomous vehicles for smooth, continuous trajectories.
2. Unconstrained Sampling: The RRT algorithm, in its original formulation, employs an unrestricted sampling strategy across the entire map. This approach results in a proliferation of sampling points,

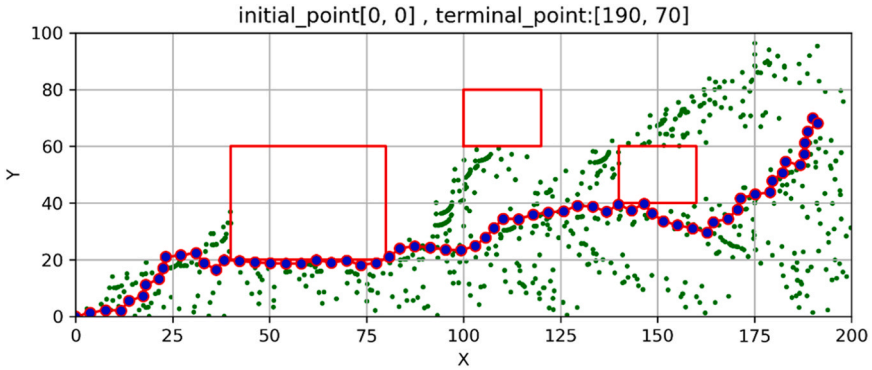

Fig. 2. The efficacy of RRT algorithm.

leading to variability in planning speed. The process oscillates between periods of rapid and slow progression, undermining the stability and efficiency of the path planning process.

3. Path Quality Discrepancy: When compared to an ideal or optimal path, the quality of the path generated by the RRT algorithm is markedly inferior. The route is often encumbered with superfluous segments, contributing to an overall increase in path length and a reduction in practical utility. This divergence from optimal path characteristics highlights a significant area for improvement in the algorithm's application to autonomous navigation.

2.2.2. A* algorithm

The A* algorithm, initially proposed by Hart, Nilsson, and Raphael in 1968, has withstood the test of time, establishing itself as a foundational algorithm in the domains of pathfinding and graph traversal. Grounded in the principles of best-first search, the algorithm strikingly balances two pivotal aspects of algorithmic efficiency: completeness and optimality. The effectiveness of the A* algorithm is primarily dependent on its heuristic function, $f(n) = g(n) + h(n)$, where $f(n)$ represents the total estimated cost of the path through the node n, g(n) denotes the actual cost from the start node to n, and h(n) signifies the estimated cost from n to the goal. This heuristic function is instrumental in estimating the costs from a given node to the goal, thereby guiding the search process towards the most promising paths. The Pseudocode of the A* algorithm is in Algorithm 2.

Algorithm 2. A*.

.

Algorithm 2: A*

Input: *start*, *goal*, *graph*
Output: path
1. *Initialize openSet, closedSet to empty sets*
2. *Add start to openSet*
3. **while** *openSet* **is not** *empty* do
4. *current* ← *node in openSet with the lowest f_score*
5. **if** *current* = *goal* **then**
6. **return** *reconstruct_path(cameFrom, current)*
7. *remove current from openSet*
8. *add current to closedSet*
9. **for** *each neighbor* **of** *current* **do**
10. **if** *neighbor* **in** *closedSet* **then**
11. **continue**
12. *tentative_gScore* ← *g_score[current] + dist_between(current, neighbor)*
13. **if** *neighbor* **not in** *openSet* **then**
14. *add neighbor to openSet*
15. **else if** *tentative_gScore* ≥ *g_score[neighbor]* **then**
16. **continue**
17. *cameFrom[neighbor]* ← *current*
18. *g_score[neighbor]* ← *tentative_gScore*
19. *f_score[neighbor]* ← *g_score[neighbor] + heuristic_cost_estimate(neighbor, goal)*
20. **return** *failure*

Table 1
Path planning speed of A* algorithm under three different resolutions.

map resolution	100 * 100	300 * 300	900 * 900
A* algorithm.	0.01 s	3.31 s	14 s
RRT algorithm	0.02 s	0.5 s-20 s	1 s-60 s

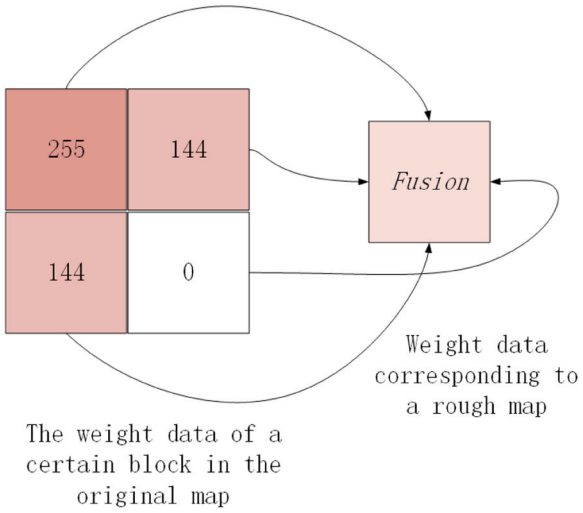

Fig. 3. Coarse sampling process.

3. Design of Improved RRT Algorithm

3.1. Establishment of double layer map and implementation of coarse path guided domain

In pursuit of enhancing the efficiency of path planning while circumventing issues of dimensional catastrophe, it is imperative to construct a grid map utilizing pre-acquired map information prior to the initiation of path planning tasks by intelligent vehicles. Each grid unit encapsulates critical data, encompassing the locational coordinates of the current node and the associated traversal cost. Irrespective of the specific path planning algorithm employed, there exists an inverse relationship between the resolution of the grid map and the algorithm's operational speed. Table 1 elucidates this correlation by presenting comparative data on the planning speeds achieved by various algorithms across maps of differing resolutions.

The results indicate that the A* algorithm, as a graph based search algorithm, has a negative correlation between its planning speed and map resolution, and increases exponentially with the expansion of map resolution. However, as a sampling based method, the RRT algorithm has a relatively small increase in planning speed compared to the A-star algorithm, although its relationship between planning speed and map

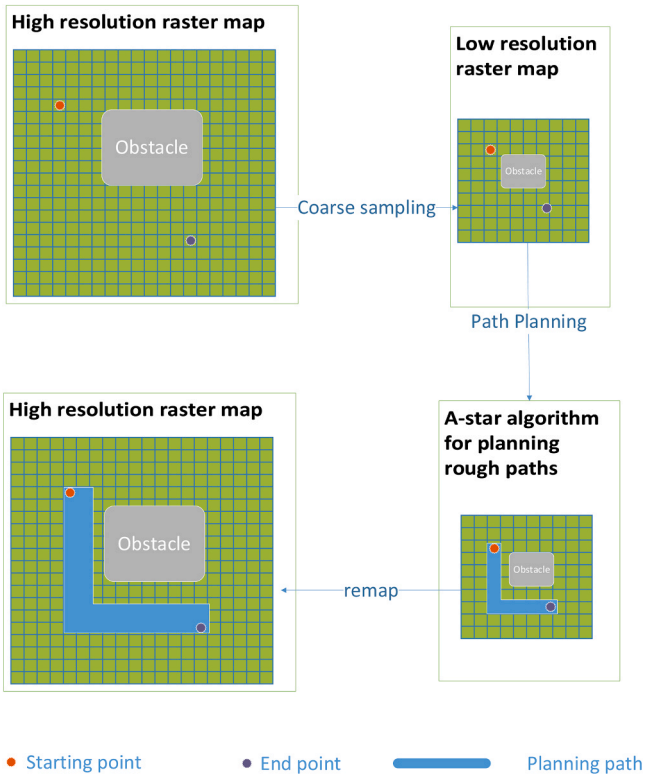

Fig. 4. Dual Map Schematic.

resolution is related. However, it also exhibits an unstable planning speed, which is determined by the unique infinite pure random method of the original RRT algorithm. The higher the resolution of the map, the more redundant nodes the sampling points may collect, If the range of RRT sampling can be limited, the redundant nodes collected can be significantly reduced, which will also stabilize the sampling time.

However, in intelligent vehicle navigation problems, in order to accurately output the path planning of the intelligent vehicle and avoid small obstacles that are difficult to detect, it is often necessary to generate a planned route on a high-resolution map. Therefore, this article proposes a two-layer map structure. On the basis of pre obtained map information to construct a grid map, the high-resolution grid map is further extracted and subjected to coarse sampling. The coarse sampling method for constructing a low resolution grid coarse map is shown in Fig. 3. The original map is added to each map block according to the scale coefficient to obtain the average value of the corresponding block in the coarse map.

The time cost of planning on a low-resolution grid map is very low. Remapping the paths planned on the low-resolution grid map back to the high-resolution grid map will form a restricted area, where the RRT planning algorithm is used to improve the running speed and stability of the path planning algorithm. This method can significantly improve the efficiency of path planning and reduce hardware resource requirements.

The A* algorithm, as a grid-based search algorithm, although its average planning speed cannot meet the planning requirements in complex and high-resolution scenes, it exhibits excellent performance in simple and low resolution scenarios. From the perspectives of path length and average curvature, the planned path quality is excellent. Due to the shortcomings of traditional RRT algorithm in terms of planning speed and path availability, in order to accelerate the planning speed of RRT algorithm, this article uses A-star algorithm to plan and establish coarse paths in low resolution grid maps, and uses the routes planned by A* algorithm as the restricted area of RRT algorithm. The specific process is shown in Fig. 4.

If you want to use the RRT algorithm to generate random points in a restricted area, first discuss whether the point to be generated is in the restricted area, and then discuss whether the point is in the obstacle.

The algorithm in this article first generates random points on a high-resolution map, and then converts the coordinates of the random points in the high-resolution map to a low-resolution map for comparison, determining whether the two are equal to determine whether the generated points are in a restricted area.

Set the coordinates of the generated random points on the real map as (x, y), the resolution of high-resolution maps as resolution, and the resolution of low-resolution maps as low_ Resolution: The length and width coordinates of a real map are Width and Height, respectively. The conversion formula for mapping random points from the real world to high-resolution raster maps is shown in equations 3–1 and 3–2.

$$x\_high = x^{*}resolution/Width3-1$$

$$y\_high = y^{*}resolution/Height3-2$$

The formula for mapping from high-resolution raster maps to low-resolution maps is shown in equations 3–3 and 3–4.

$$x\_low = x\_high^{*}(low\_resolution/high\_resolution)3-3$$

$$y\_low = y\_high^{*}(low\_resolution/high\_resolution)3-4$$

By applying the above formula, it is possible to quickly and efficiently map the point coordinate positions generated from high-resolution images back to the low-resolution map, thereby quickly and efficiently checking whether the point coordinates are within the planned restricted area.

Algorithm 3. Combined A* and RRT with Restricted Areas.

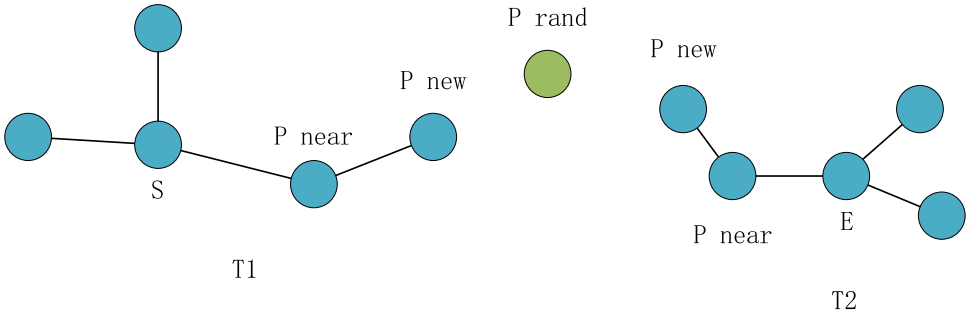

Fig. 5. The Search Process of Bidirectional RRT Algorithm.

Algorithm 3: Combined A* and RRT with Restricted Areas

Input: *High-resolution map, Low-resolution map, Start, Goal*

Output: *Path*

Use A algorithm on Low-resolution map to plan a coarse path*
Define the path from A as the restricted area for RRT*
Initialize RRT on the High-resolution map
while *RRT* **has not reached** *the goal do*
  Generate a random point (x, y) on the High-resolution map
  Convert (x, y) to high-resolution map coordinates:
    *x_high = x * resolution / Width*
    *y_high = y * resolution / Height*
  Convert high-resolution coordinates to low-resolution:
    *x_low = x_high * (low_resolution / high_resolution)*
    *y_low = y_high * (low_resolution / high_resolution)*
  if *(x_low, y_low) is* **in** *the restricted area from A* then*
    continue
  if *(x, y) is in an obstacle then*
    continue
  Add (x, y) to RRT
  if *RRT reaches the goal* **then**
    return *the path from RRT*
return *failure if no path is found*

.

This pseudocode outlines a process where the A* algorithm is initially used to establish a coarse path on a low-resolution map. This path is then used as a restricted area for the RRT algorithm applied on a high-resolution map. The algorithm includes steps for generating random points, converting these points between high and low resolutions, and checking if they fall within the restricted area or an obstacle. The RRT continues until it reaches the goal or fails to find a path.

3.2. Design of an improved bidirectional bias RRT algorithm

The bidirectional RRT algorithm is mainly used to shorten search time, and its core idea is to generate two random trees simultaneously at the starting and ending points. The search process is shown in Fig. 5. Assuming there are two trees T1 and T2, after each round of random point generation, the two trees alternately grow towards that point until the distance between the new node of T1 and the new node of T2 is less than the pre-set threshold, and then stop growing.

Although the bidirectional RRT algorithm has to some extent shortened the search time, the direction of growth in the restricted area

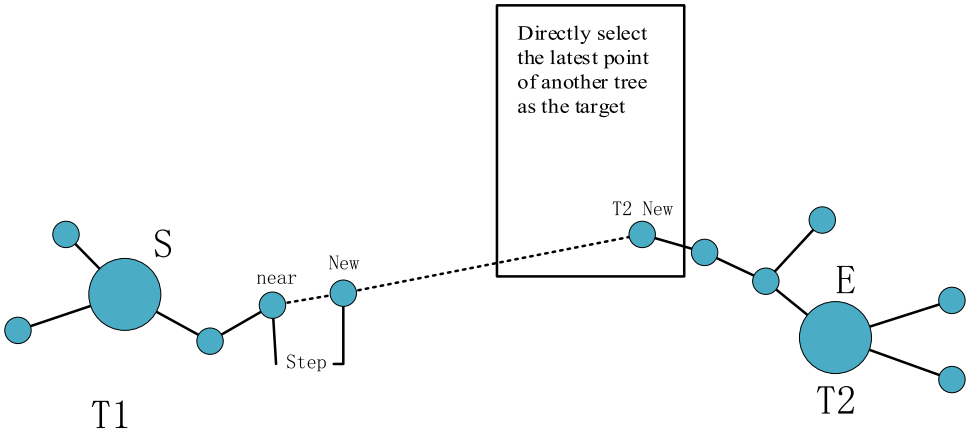

Fig. 6. The search method of the bidirectional RRT algorithm proposed in this article.

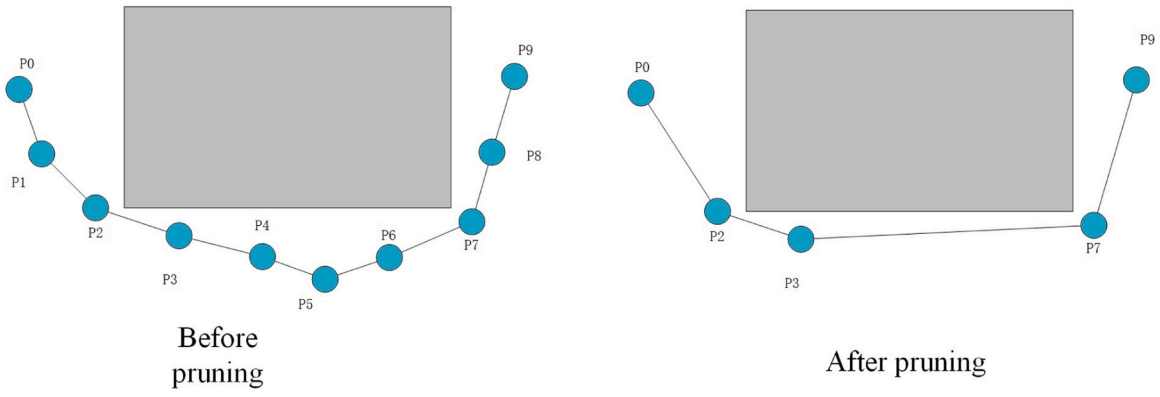

Fig. 7. Planned original path and pruned path.

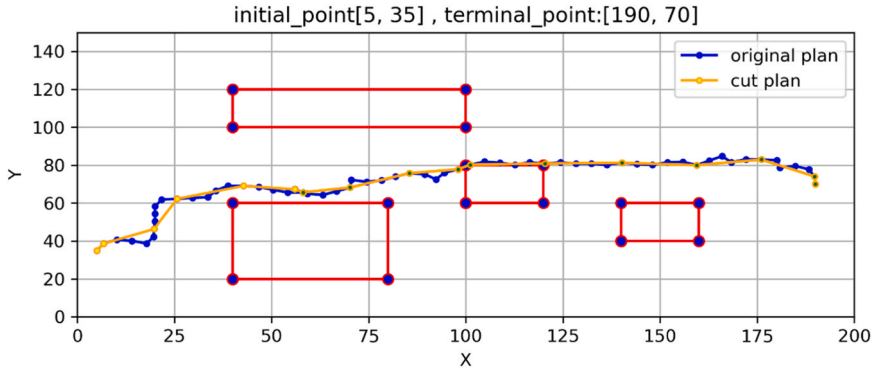

Fig. 8. Planned original path and pruned path.

is still aimless and random, leading to blind search process. This article improves the search method of the bidirectional RRT algorithm, as shown in Fig. 6. During the tree growth stage, the newly generated node of another tree is directly used as the growth direction with a certain probability. The specific implementation method is as follows:

(1)Set a threshold *N*, which can range from $N \in \{0, 1, 2, \ldots, 9, 10\}$.

(2)Set the value range of *U* and randomly generate values.

(3)Compare the sizes of *U* and *N*. If *U* reaches the threshold, directly select the newly generated node from another tree as the growth direction; On the contrary, traditional growth will continue in the direction of randomly generated points.

It should be noted that the values of *U* and *N* directly affect the generation speed of the random tree, and appropriate values need to be selected based on different task objectives. The growth probability *P* of a new node in another *tree_ Other* is represented by equations 3–5, where *U_ Range* represents the range of *U* values, and $N + 1$ represents the number of values less than or equal to N.

$$P_{other} = 1 - (N+1)/(U\_range + 1)3 - 5$$

The specific selection formula is shown in formulas 3–6:

$$Nodes\ (\ rand\ ) = \begin{cases} rand\_point\ (\ )\ , \& U \le N \\ choose\_other\_Tree\_new, \& U > N \end{cases} 3 - 6$$

When selecting the latest node of another tree (*T2*) as the target node, the position of the new node in *T1* tree is shown in equations 3–7 :

$$P_{1new} = P_{1near} + step \frac{P_{2new} - P_{1near}}{\| P_{2new} - P_{1near} \|} 3 - 7$$

Due to the fact that the sampling points of the bidirectional RRT algorithm are within the restricted area planned by the A-star algorithm, the position of the generated points of the bidirectional RRT is restricted

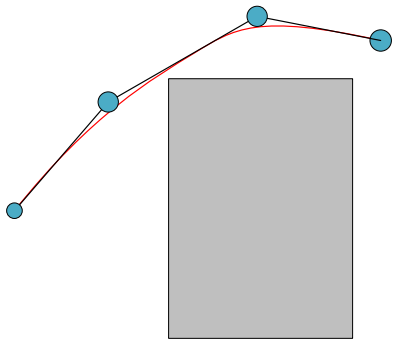

Fig. 9. Bezier curve optimization.

by the restricted area, and its blind scalability is constrained, therefore, selecting P_{2new} point as the target point to expand towards it must have the following advantages compared to the initial point of T2 tree, that is, the module length of two points $P_{2new} - P_{1near}$ must be less than $\| P_{2start} - P_{1near} \|$, as shown in equations 3–8:

$$\| P_{2new} - P_{1near} \| < \| P_{2start} - P_{1near} \| \ 3 - 8$$

Compared to the greedy strategy used by some scholars to generate offset points based on the target endpoint, selecting the latest node of the spanning tree can make the path search speed faster, as shown in Fig. 6.

3.3. Reoptimization of planning paths

Although the use of guidance domain and bidirectional bias strategy can reduce the randomness of generated routes, most of the planned routes still do not meet the requirements of vehicle dynamics. Therefore,

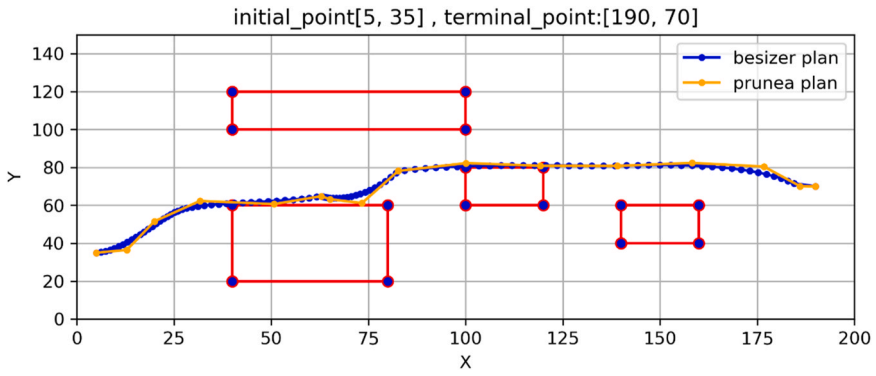

Fig. 10. Planning the final effect.

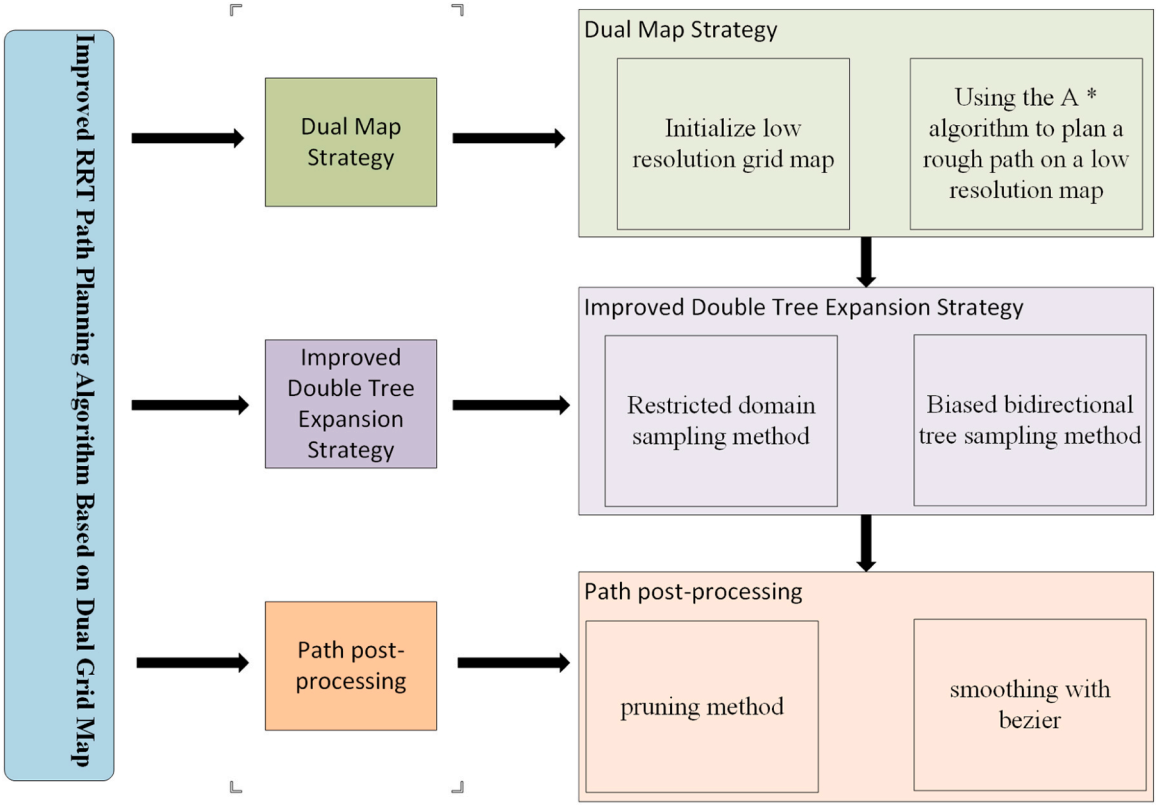

Fig. 11. Improved algorithm flowchart.

we need to perform secondary optimization on the generated planning path adoption.

Firstly, prune and optimize the planned route, and assume that some of the planned paths are as follows

$$\{p0, p1, p2, p3, p4, p5, p6, p7, p8, p9\}$$

The pruning function starts from point *p0* and proceeds to the next node to determine whether there is an intersection between point *p0* and point *p1* with a rectangular obstacle. If there is no intersection, it directly connects point *p0* and point *p2*, and continues to determine whether the line intersects with the rectangle until a node that intersects with the rectangle is found. This point is set as a necessary point, and then pruning is continued based on this point.

The pruning function uses vector cross product to determine whether a line segment intersects with a rectangle, in order to determine whether it intersects with a rectangular obstacle. Firstly, set the coordinates of the two endpoints of the line segment (*P1*, *P2*) and the four vertices of the rectangle (*A, B, C, D*). Using formulas 3–9, calculate the intersection of line segment *P1P2* and the four edges of the rectangle, that is, calculate the vector cross product *cross_product* of each edge separately

$$cross\_product = (P2.x–P1.x)*(B.y–A.y) - (P2.y - P1.y)*(B.x - A.x)3 - 9$$

If the *cross_product* of line segment *P1P2* and any edge vector is not 0, it indicates that the line segment intersects with the rectangle. If the *cross_product* of line segment *P1P2* and all edges is 0, it means that the line segment does not intersect with the rectangle.

As shown in Fig. 7, the required points are $\{p0, p2, p3, p7, p9\}$, respectively. Compared to the tree before pruning, the pruned tree reduces redundant nodes, making the route more direct and the distance shorter. Fig. 8 shows a comparison of planned paths using pruning algorithms and those without pruning algorithms. Compared to the original blue path, the pruned yellow path is more streamlined and has higher direct efficiency.

Although the distance after pruning is shorter, the path has become

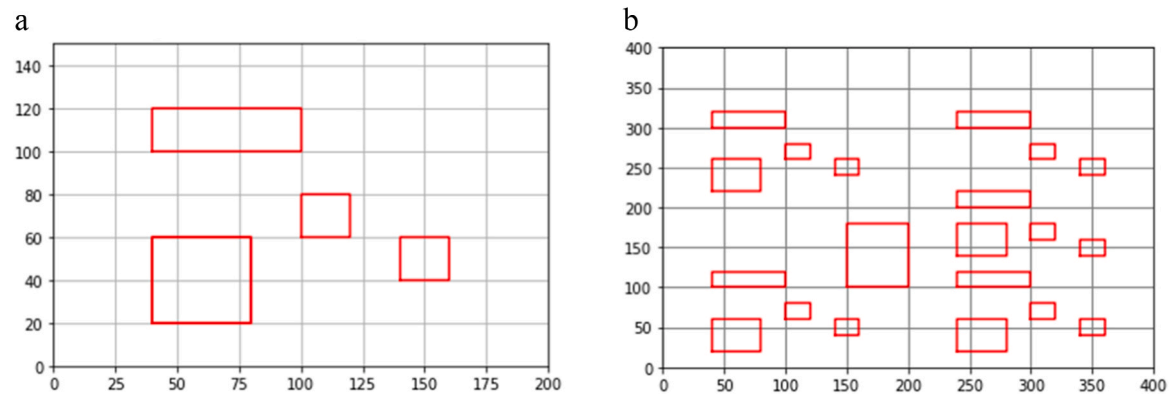

Fig. 12. a Simple simulation scenario. b Simple simulation scenario.

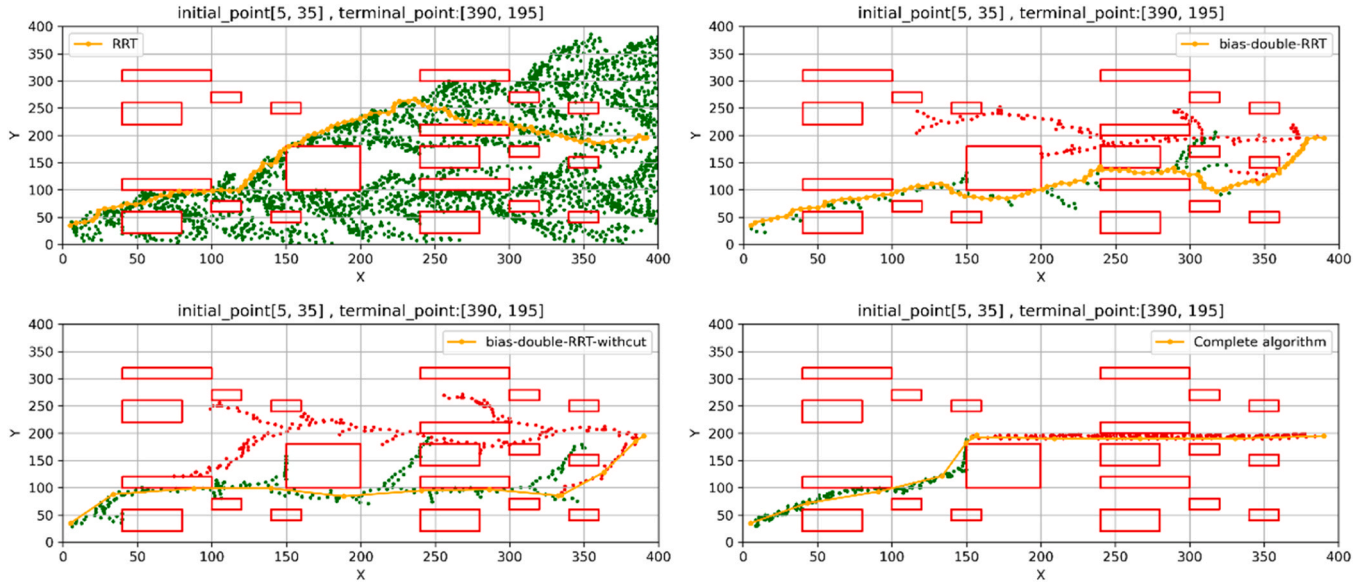

Fig. 13. results of each algorithm in ablation experiments (The algorithms from left to right and from top to bottom are (RRT, bias double RRT, bias double RRT with cut, complete algorithm).

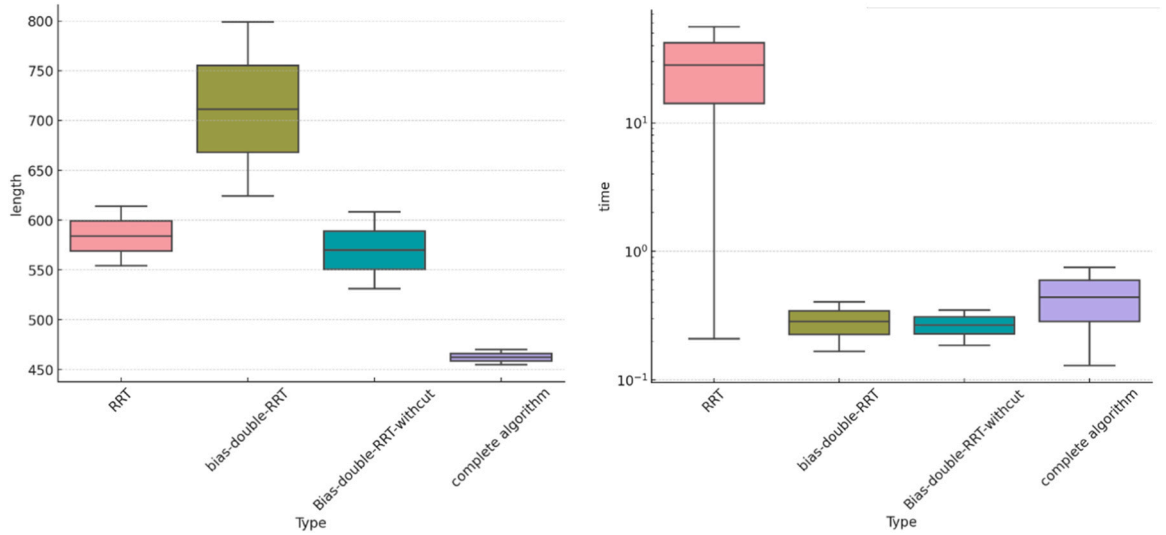

Fig. 14. Comparison of Planning Algorithms in Simple Scenarios.The left figure shows the actual path lengths of various algorithms in ablation experiments from left to right are the " RRT, bias double RRT, bias double RRT with cut, complete algorithm", The figure on the right shows the actual time planned by various algorithms in ablation experiments, from left to right "RRT, bias double RRT, bias double RRT with cut, complete algorithm".

Table 2
Comparison of Algorithms in DIFFERENT Scenarios.

	Result	Simple Map	Complex Map
RRT-Connect	Average time (s)	0.15	2.54
	Average path length	272	581
	Average path curvature	0.15	0.125
A* Algorithms	Average time (s)	0.16	4.11
	Average path length	216	471
	Average path curvature	0.07	0.008
GSRRT-Connect	Average time (s)	0.05	0.33
	Average path length	212	592
	Average path curvature	0.06	0.049
The algorithm proposed in this article	Average time (s)	0.03	0.69
	Average path length	204	463
	Average path curvature	0.02	0.009

basically composed of line segments, which is not smooth for vehicles relying on the Ackermann chassis. Therefore, the next step is to use Bessel curves for path optimization of the pruned route. Bezier curve is an interpolation equation with good local control characteristics, which is widely used in computer graphics and path planning. Its general representation in the two-dimensional plane is shown in equations 3–10, and its performance is shown in Fig. 9:

$$B(t)=\sum_{i=0}^{n}\binom{n}{i}Pi(1-t)^{n-i}t^{i}$$

$$=\binom{n}{0}P0(1-t)^{n}t^{0}+\binom{n}{1}P1(1-t)^{n-1}t^{1}+\ldots+\binom{n}{n-1}Pn-1(1-t)^{1}t^{n-1}3-10$$

In the equation, n represents the degree of the Bezier curve. Since using a third-order Bezier curve can effectively optimize the smoothness between the two line segments between the three points in the pruned path, the optimization effect of using a third-order Bezier curve is shown in Fig. 9. The comparison effect between the path using a Bezier curve (blue) and the path not using a Bezier curve (yellow) is shown in Fig. 10,

Fig. 11 is the flowchart of the algorithm designed in this article.

Fig. 11 summarizes the operational steps of the algorithm presented in this paper. To enhance path planning efficiency and improve route quality, we initially adopt the concept of a dual-map. This involves using the A* algorithm on a low-resolution map to guide the overall algorithm with a coarse path. Searching within this path reduces the randomness of route expansion, thereby increasing both speed and accuracy. The A* algorithm is chosen for its efficiency and quality of planning at low resolution. Subsequently, we apply an improved bidirectional biased tree algorithm for planning within this coarse path. The enhanced biased tree algorithm demonstrates excellent planning speed. Finally, path post-processing techniques like pruning and Bezier curves are used to refine the algorithmically planned paths, reducing their length and smoothing out any abrupt changes.

4. Experimental results

This article uses Python to simulate and validate the path planning algorithm proposed in this article. The processor of the verified device host is 12th Gen Intel (R) Core (TM) i7–12700 H, with a main frequency of 2.30 GHZ and a built-in memory of 40 GB.

In order to accurately verify the planning ability of the algorithm proposed in this article in different scenarios, the experimental validation maps selected for simulation experiments in this article include two types: one is a simple obstacle scene, and the other is a complex obstacle scene. For a simple scenario as shown in Fig. 12.a, a resolution of 150 * 200 is selected, with initial points {5, 35} and target points {190, 70}. There are a total of four rectangular obstacles with varying sizes. For a complex scenario, as shown in Fig. 12.b, a resolution of 400 * 400 is selected, with initial points (5, 35) and target points (390,195). There are a total of 20 rectangular obstacles with varying sizes.

4.1. Ablation experiments

To more clearly demonstrate the functionality of the various components of the algorithm proposed in this paper, we have categorized the improvements suggested herein and conducted ablation experiments accordingly. The experiments were divided into three groups: Group (1) RRT, Group (2) bias-double-RRT, Group (3) bias-double-RRT+cut, and Group (4) Complete Algorithm. The experimental setting was based on the complex environment of Map B mentioned earlier. The planning

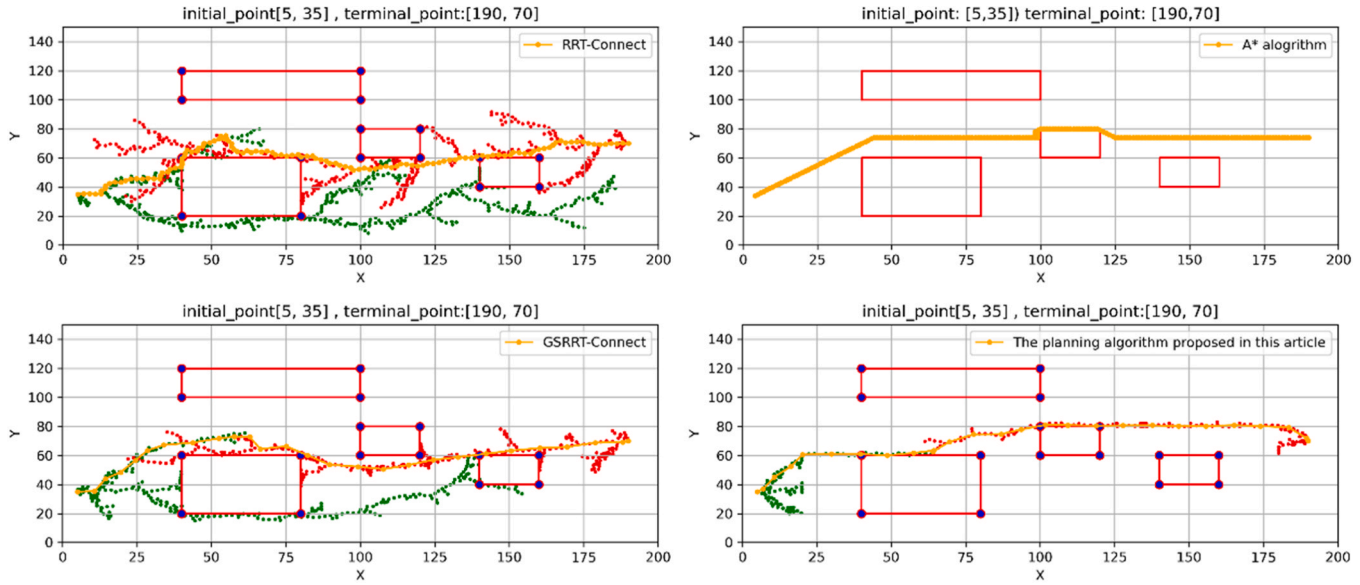

Fig. 15. Comparison of Planning Algorithms in Simple Scenarios (The algorithms from left to right and from top to bottom are (RRT, bias double RRT, bias double RRT with cut, complete algorithm).

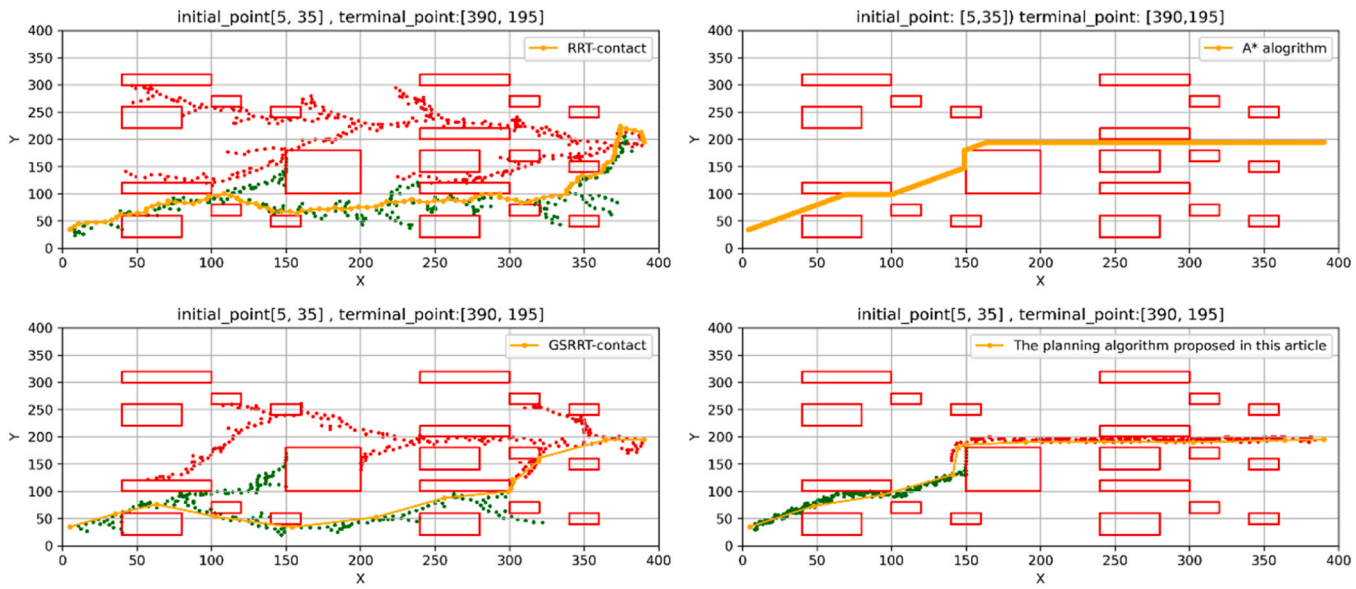

Fig. 16. Comparison of Planning Algorithms in Complex Scenarios (The algorithms from left to right and from top to bottom are (RRT, bias double RRT, bias double RRT with cut, complete algorithm).

results of each algorithm are illustrated in Fig. 13.

The results demonstrate that, across multiple ablation experiments, the operational speeds of the algorithms were as follows: RRT: 8.7 s, bias-double-RRT: 0.24 s, Bias-double-RRT-withcut: 0.22 s, and the complete algorithm: 0.50 s. Concurrently, the path lengths generated by each algorithm were: RRT: 629 units, bias-double-RRT: 759 units, Bias-double-RRT-withcut: 613 units, and the complete algorithm: 509 units. Boxplot XX provides a visual representation of these algorithmic performances across multiple runs.Fig. 14.

The ablation experiments reveal that each module of the algorithm presented in this paper performs as anticipated. The bi-directional biasing approach to the RRT tree generation, in comparison to the original RRT algorithm, demonstrates superior operational speed, with a reduction in planning time by 3600 %. However, this expansion method resulted in excessive growth of the tree, leading to path lengths that were even longer than those produced by the original RRT algorithm. The pruning strategy proposed in this paper effectively reduces the length of the generated path without compromising operational speed. Yet, the randomness inherent in this approach still led to overly extended paths. The incorporation of a dual-grid map concept in the algorithm addressed this issue, reducing the path length by 30 % compared to the versions without the dual-grid map concept.

4.2. *Simulation comparison*

To demonstrate the superiority of the algorithm presented in this paper, we compared it with several established methods, including the classic RRT-connect algorithm, the A* algorithm, and the newly

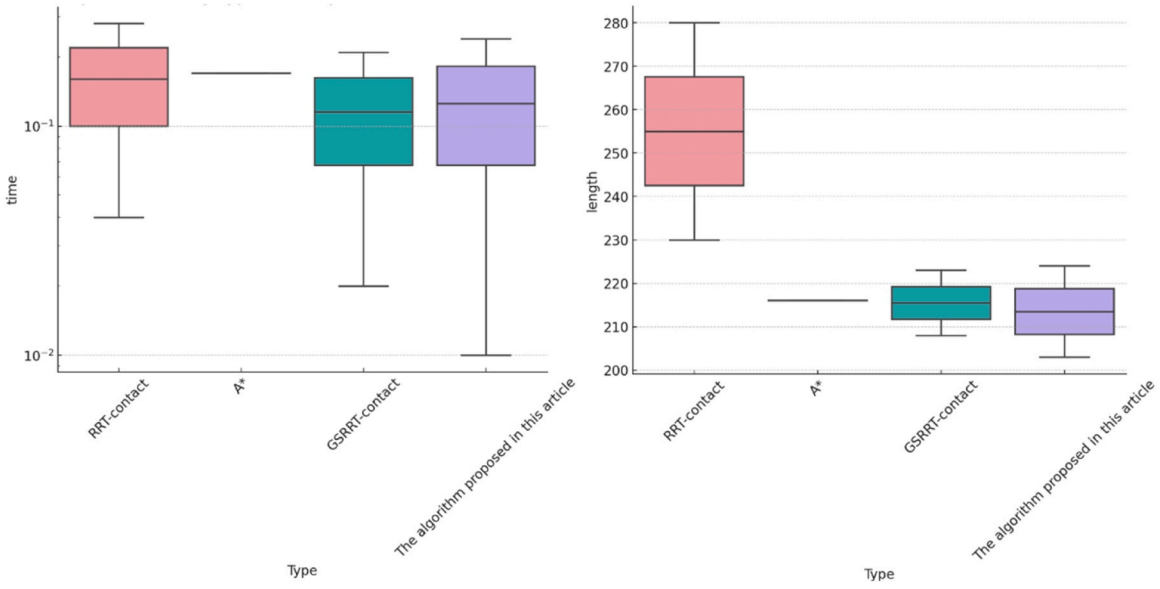

Fig. 17. Comparison of Path Length and Planning Time of Different Algorithms in Simple Scenarios (The left figure shows the running time of various algorithms in Simple scenarios (from left to right are the " RRT-connect algorithm, A * algorithm, GSRRT-connect algorithm, and the algorithm proposed in this article" The figure on the right shows the actual path lengths planned by various algorithms in Simple scenarios, from left to right "RRT connect algorithm, A * algorithm, GSRRT connect algorithm, algorithm proposed in this article").

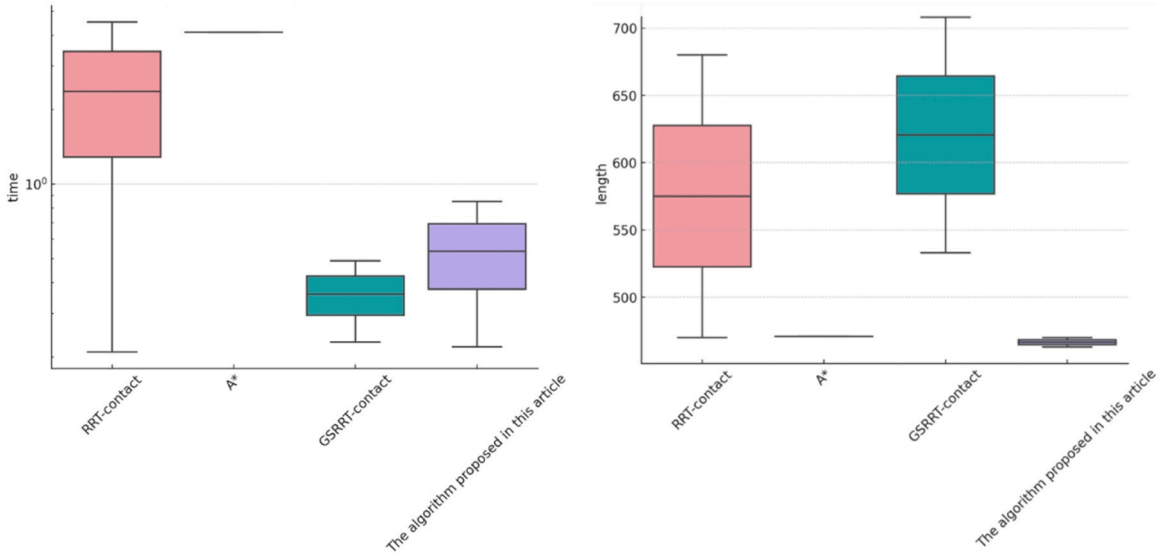

Fig. 18. Comparison of Path Length and Planning Time of Different Algorithms in Complex Scenarios (The left figure shows the running time of various algorithms in complex scenarios (from left to right are the " RRT-connect algorithm, A * algorithm, GSRRT-connect algorithm, and the algorithm proposed in this article" The figure on the right shows the actual path lengths planned by various algorithms in complex scenarios, from left to right "RRT connect algorithm, A * algorithm, GSRRT connect algorithm, algorithm proposed in this article").

Fig. 19. Real vehicle verification scenario.

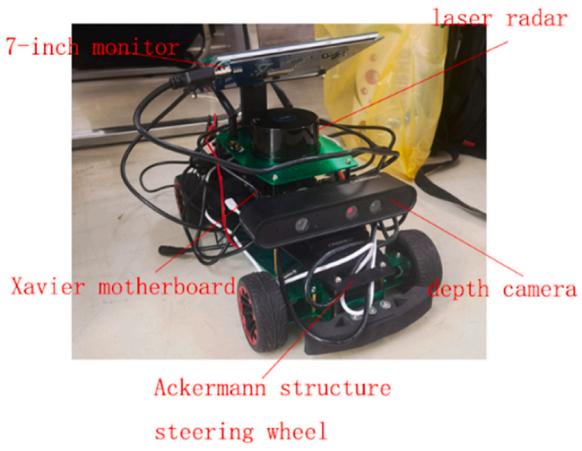

Fig. 20. Structural diagram of experimental vehicle.

published GSRRT-Connect algorithm from other researchers in 2023 [11]. We conducted repeated sampling in both simple and complex scenarios 100 times and calculated the average values for each metric. Table 2 shows the average parameters for each algorithm after 100 tests.

Figs. 15 and 16 illustrate the comparisons of various planning algorithms in simple and complex scenarios, respectively. In the planning graph of the RRT-Connect algorithm, the green and red points denote the generation locations of random points for the two trees of the RRT-Connect, and the yellow line represents the generated path. In the A* algorithm's graph, the yellow color signifies that the generated path is always consistent. In the GSRRT-Connect graph, yellow indicates the generation location of random points for tree 1, red for tree 2, and the yellow line denotes the generated path. In the planning graph of the algorithm proposed in this paper, yellow represents the generation location of random points for tree 1, red for tree 2, and the yellow line signifies the generated path.

Firstly, an analysis of the average planning time in the table reveals notable trends. In both simple and complex scenarios, the RRT-Connect algorithm exhibits the longest average planning time, with 0.15 s in simple scenes and 2.54 s in complex ones. The A* algorithm takes 0.16 s in simple maps, while the GSRRT-Connect shows high computational efficiency in both settings, taking only 0.05 s in simple maps and 0.33 s in complex ones. The algorithm proposed in this paper records the shortest average time in simple maps at 0.03 s, and 0.69 s in complex maps.

Secondly, the average planning lengths in the table were analyzed. In simple environments, the RRT-Connect algorithm's average planning length is 272, and 581 in complex scenes. The A* algorithm plans for 216 in simple maps and 471 in complex ones. The GSRRT-Connect 's planning lengths are 212 in simple scenarios and the longest at 592 in complex ones. The algorithm introduced in this paper outperforms all others in both simple and complex scenarios, with planning lengths of 204 and 463, respectively. This represents a substantial lead over competing algorithms, with approximately a 29 % improvement over advanced peer algorithms and about a 30 % improvement over RRT-connect.

Lastly, the curvature of the planned paths in the table was evaluated. Among all tested algorithms, the one proposed in this article achieved the most optimal average curvature, with 0.02 in simple scenarios and 0.009 in complex ones. The A* algorithm's average curvature is 0.07 in

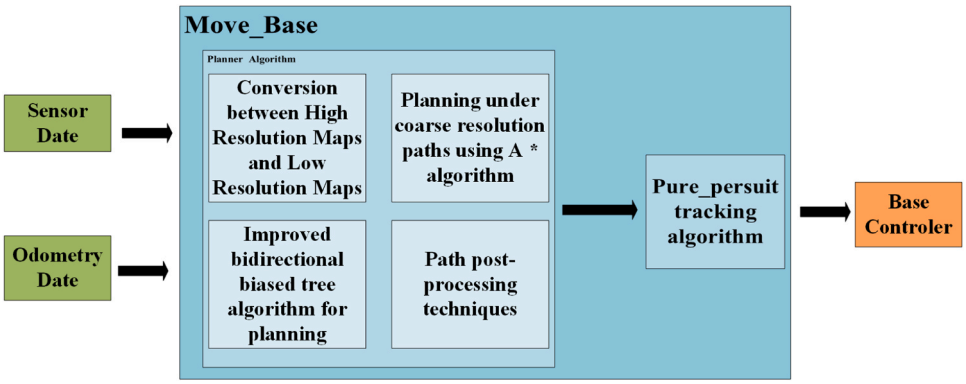

Fig. 21. Real vehicle path planning process.

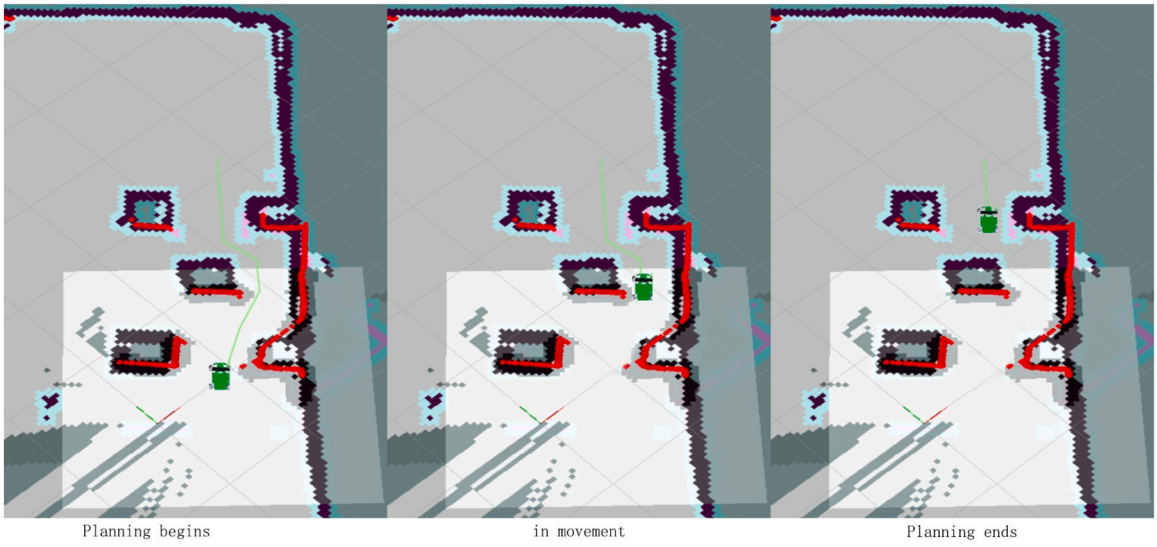

Fig. 22. Real vehicle path planning process.

simple scenarios and 0.008 in complex ones. The GSRRT-Connect algorithm's average curvature is less optimal than both the A* algorithm and the one presented in this paper, with 0.06 in simple scenarios and 0.049 in complex ones. The RRT-connect algorithm has the poorest average curvature, with 0.15 in simple scenarios and 0.125 in complex ones.

Figs. 17 and 18 presents box plots illustrating the performance of various algorithms in repeated experiments. A comprehensive analysis of the table and figure reveals that in simple environments, the algorithm proposed in this paper outperforms both RRT-Connect and A* algorithms in terms of planning speed and route length. Its planning time and quality are comparable to the GSRRT-Connect algorithm. In complex environments, the runtime of our proposed algorithm significantly surpasses that of the RRT-Connect and A* algorithms, and is slightly inferior to the GSRRT-Connect algorithm. However, in terms of the quality of the planned route, our algorithm substantially outperforms all four, including the GSRRT-Connect algorithm, demonstrating a clear lead in complex scenario planning.

4.3. Real vehicle verification

In addition to the simulation experiments mentioned above, this article also designed real vehicle experiments to verify the effectiveness of the algorithm in practical engineering. The actual vehicle adopts an intelligent car with Ackermann structure, as shown in Fig. 20. The components of the intelligent car include a 7-inch display, LiDAR, Xavier motherboard, depth camera, and Ackermann structure steering gear.

The overall structure of the autonomous driving system is shown in Fig. 21 and consists of four parts: perception module, localization module, planning module, and control module. The algorithms proposed in this paper are deployed within the control module's Move_Base framework. The perception module and Odometry localization module transmit vehicle location and map information to the planning module. The algorithm in this article will complete all planning in the planner. Firstly, high-resolution map information will be converted into low-resolution map information. Then, the A * algorithm will be used for planning in the low-resolution map. Then, the improved bidirectional offset tree algorithm will be used for planning in the area output by the A * algorithm. Finally, the path will be provided to the pure tracking algorithm through path post-processing technology. Finally, output the speed command to the vehicle controller to achieve autonomous driving of the vehicle.

The scenario for actual vehicle verification in this article is shown in Fig. 19. A total of 5 obstacles were selected and the map was created using the G-mapping algorithm using LiDAR. Deploy the algorithm in this article to the ROS system for global path planning, and select Pure as the local path planner for the car_ Permit tracking algorithm. The overall actual vehicle experiment process has been visualized by Rviz and shown in Fig. 22. The algorithm proposed in this article has shown excellent navigation performance in actual obstacle avoidance scenarios, which can meet the global navigation requirements of Ackermann structured cars.

5. Conclusion and outlook

This article aims to propose a path planning algorithm that combines the A-star algorithm with the Rapidly-exploring Random Tree (RRT), and analyzes the existing RRT algorithm and other improved methods. Firstly, using the "dual layer map" method, a low resolution coarse map based on a high-resolution grid map is established, and the initial coarse path is obtained using the A-star algorithm in the low resolution coarse map. Remap this path back into a high-resolution grid map as a restricted domain and use an improved RRT spanning tree and target bias strategy within that region. With a certain probability, use the latest tree node of the other party to expand the bidirectional RRT algorithm for sampling to accelerate path generation speed. Finally, the generated path is smoothed through pruning and Bessel curve smoothing to meet the dynamic constraints of Ackermann chassis vehicles.

In this paper, the effectiveness of the proposed improvements to our algorithm was first analyzed through ablation experiments in a simulated environment, confirming the efficacy of our methods. The introduction of the dual-map concept significantly enhances the quality of path planning. Additionally, the modified biased bidirectional tree method substantially increases the speed of path generation. The proposed pruning strategy and Bezier curve optimization further align the paths with vehicle dynamics constraints. The algorithm was then benchmarked against classic algorithms such as the RRT, RRT-Connect, and A* algorithm, demonstrating superior performance across various map types. Notably, in complex maps, it achieved an average speed increase of 1800 % and a 40 % reduction in path length compared to the RRT algorithm. Furthermore, when compared with the latest research by peers, our algorithm showed comparable capabilities in simple maps and achieved a 30 % reduction in path length in complex scenarios at equivalent planning speeds. Ultimately, the proposed algorithm was deployed on a ROS-powered intelligent vehicle and successfully executed path planning tasks, demonstrating its feasibility in vehicles with Ackermann steering dynamics. In summary, this algorithm can be applied to unmanned vehicles, indoor robots, industrial production and other fields.

Despite the algorithm's outstanding performance relative to classical approaches, challenges persist in real-world applications. Optimal parameterization for transitioning between high-precision and coarse maps still requires significant trial and error. Future research will explore machine learning methods to identify optimal transition patterns. Additionally, there is potential for further optimization in the growth step length of the RRT algorithm. Developing an adaptive approach for determining the algorithm's step length will be a primary focus in subsequent research.

Acknowledgments

This work was supported by the Educational Commission Project of Tianjin, China, under grant (No. 2020KJ120).

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Juan Du, Peng Zheng, Zhongyu Xie, Yu Yang, Hongxia Chu, Gaobo Yu. Research on path planning algorithm based on security patrol robot, in: Proceedings of the 2016 IEEE International Conference on Mechatronics and Automation, pp. 1030–1035, IEEE, 2016.

[2] Stefania Pellegrinelli, Andrea Orlandini, Nicola Pedrocchi, Alessandro Umbrico, Tullio Tolio, Motion planning and scheduling for human and industrial-robot collaboration, CIRP Ann. 66 (1) (2017) 1–4.

[3] C. Santos, Luís Filipe N. Santos E.J. Solteiro Pires António Valente Pedro Costa Sandro Magalhães. Path planning for ground robots in agriculture: A short review, in: Proceedings of the 2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), pp. 61–66. IEEE, 2020.

[4] Masato Noto, Hiroaki Sato, A method for the shortest path search by extended Dijkstra algorithm, in: Smc 2000 Conference Proceedings. 2000 IEEE International Conference on Systems, Man and Cybernetics. 'cybernetics Evolving to Systems, Humans, Organizations, and Their Complex Interactions'(cat. no. 0, vol. 3, pp. 2316–2320. IEEE, 2000.

[5] František Duchoň, Andrej Babinec, Martin Kajan, Peter Beňo, Martin Florek, Tomáš. Fico, Ladislav Jurišica, Path planning with modified a star algorithm for a mobile robot, Procedia Eng. 96 (2014) 59–69.

[6] Jianhua Liu, Jianguo Yang, Huaping Liu, Xingjun Tian, Meng Gao, An improved ant colony algorithm for robot path planning, Soft Comput. 21 (2017) 5829–5839.

[7] Steven M. LaValle, J.Kuffner James, B.R. Donald, Rapidly-exploring random trees: Progress and prospects, Algorithm Comput. Robot.: N. Dir. 5 (2001) 293–308.

[8] Gildardo Sanchez, J.-C. Latombe Using a PRM planner to compare centralized and decoupled planning for multi-robot systems, in: Proceedings 2002 IEEE international conference on robotics and automation (Cat. No. 02CH37292), vol. 2, pp. 2112–2119. IEEE, 2002.

[9] James J. Kuffner, M.La.Valle Steven. RRT-connect: An efficient approach to single-query path planning, in: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065), vol. 2, pp. 995–1001. IEEE, 2000.

[10] Sertac Karaman, Emilio Frazzoli, Sampling-based algorithms for optimal motion planning, Int. J. Robot. Res. 30 (7) (2011) 846–894.

[11] Yechen Li, Shaochun Ma, Navigation of apple tree pruning robot based on improved RRT-connect algorithm, Agriculture 13 (8) (2023) 1495.

[12] Kun Hao, Yang Yang, Li Zhisheng, Liu Yonglei, Zhao Xiaofang, CERRT: a mobile robot path planning algorithm based on RRT in complex environments, Appl. Sci. 13 (17) (2023) 9666.

[13] Cheng Liu, Qingchun Feng, Zuoliang Tang, Xiangyu Wang, Jinping Geng, Lijia Xu, Motion planning of the citrus-picking manipulator based on the TO-RRT algorithm, Agriculture 12 (5) (2022) 581.

[14] Zhen Zhang, Defeng Wu, Jiadong Gu, Fusheng Li, A path-planning strategy for unmanned surface vehicles based on an adaptive hybrid dynamic stepsize and target attractive force-RRT algorithm, J. Mar. Sci. Eng. 7 (5) (2019) 132.

[15] Ahmed Hussain Qureshi, Yasar Ayaz, Potential functions based sampling heuristic for optimal path planning, Auton. Robots 40 (2016) 1079–1093.

[16] In-Bae Jeong, Seung-Jae Lee, Jong-Hwan Kim, Quick-RRT*: Triangular inequality-based implementation of RRT* with improved initial solution and convergence rate, Expert Syst. Appl. 123 (2019) 82–90.

[17] Tianqi Qie, Weida Wang, Chao Yang, Ying Li, Wenjie Liu, Changle Xiang, A path planning algorithm for autonomous flying vehicles in cross-country environments with a novel TF-RRT* method, Green Energy Intell. Transp. 1 (3) (2022) 100026.

[18] Paul Lathrop, Boardman Beth, Sonia Martínez, Distributionally safe path planning: wasserstein safe RRT, IEEE Robot. Autom. Lett. 7 (1) (2021) 430–437.

[19] Tianhao Gong, Yang Yu, Jianhui Song, Path planning for multiple unmanned vehicles (MUVs) formation shape generation based on dual RRT optimization, Actuators 11 (7) (2022) 190 (MDPI).

[20] Zhenping Wu, Zhijun Meng, Wenlong Zhao, Zhe Wu, Fast-RRT: a RRT-based optimal path finding method, Appl. Sci. 11 (24) (2021) 11777.

[21] Lina Wang, Xin Yang, Zeling Chen, Binrui Wang, Application of the improved rapidly exploring random tree algorithm to an insect-like mobile robot in a narrow environment, Biomimetics 8 (4) (2023) 374.

[22] Hao Wang, Guoqing Li, Jie Hou, Lianyun Chen, Nailian Hu, A path planning method for underground intelligent vehicles based on an improved RRT* algorithm, Electronics 11 (3) (2022) 294.

[23] Yicong Guo, Xiaoxiong Liu, Xuhang Liu, Yue Yang, Weiguo Zhang, FC-RRT*: An improved path planning algorithm for UAV in 3D complex environment, ISPRS Int. J. Geo-Inf. 11 (2) (2022) 112.

[24] Farzad Kiani, Amir Seyyedabbasi, Royal Aliyev, Murat Ugur Gulle, Hasan Basyildiz, M.Ahmed Shah, Adapted-RRT: novel hybrid method to solve three-dimensional path planning problem using sampling and metaheuristic-based algorithms, Neural Comput. Appl. 33 (22) (2021) 15569–15599.

[25] Yanjie Li, Wu Wei, Yong Gao, Dongliang Wang, Zhun Fan, PQ-RRT*: an improved path planning algorithm for mobile robots, Expert Syst. Appl. 152 (2020) 113425.

[26] Fan Yang, Xi Fang, Fei Gao, Xianjin Zhou, Hao Li, Hongbin Jin, Yu Song, Obstacle avoidance path planning for UAV based on improved RRT algorithm, Discret. Dyn. Nat. Soc. 2022 (2022) 1–9.

[27] Ruinan Chen, Jie Hu, Wencai Xu, An RRT-Dijkstra-based path planning strategy for autonomous vehicles, Appl. Sci. 12 (23) (2022) 11982.

[28] Christian Zammit, Erik-Jan Van Kampen, Comparison of a* and rrt in real–time 3d path planning of uavs, Aiaa scitech 2020 Forum (2020) 0861.

[29] Ben Beklisi Kwame Ayawli, Xue Mei, Moquan Shen, Albert Yaw Appiah, Frimpong Kyeremeh, Optimized RRT-A* path planning method for mobile robots in partially known environment, Inf. Technol. Control 48 (2) (2019) 179–194.

[30] Reza Mashayekhi, Mohd Yamani Idna Idris, Mohammad Hossein Anisi, Ismail Ahmedy, Ihsan Ali, Informed RRT*-connect: an asymptotically optimal single-query path planning method, IEEE Access 8 (2020) 19842–19852.