Research on global path planning algorithm for mobile robots based on improved A*

Research on global path planning algorithm for mobile robots based on improved A* Author links open overlay panel

Xing Xu [a], Jiazhu Zeng [b], Yun Zhao [a], Xiaoshu Lü [c d]

[a] School of Information and Electronic Engineering, Zhejiang University of Science and Technology, Hangzhou, 310023, China

[b] School of Mechanical and Energy Engineering, Zhejiang University of Science and Technology, Hangzhou, 310023, China

[c] Department of Electrical Engineering and Energy Technology, University of Vaasa, Vaasa 65380, Finland

[d] Department of Civil Engineering, Aalto University, Espoo 02130, Finland

Abstract

In order to shorten searching time and reduce the quantity of redundant nodes in path planning, an improved A* algorithm was proposed. In the novel algorithm, compared with the A* algorithm, the octet neighborhood was replaced a rectangular boundary without obstacles. The map was explored in two directions from the starting point and the target point respectively. The exploration generated fewer nodes. Due to the enlargement of the search neighborhood and the rectilinear passage without obstacles in the rectangular region, the new algorithm used Euclidean distance as distance estimate. And a new operator was designed to seek the best search node to reduce the time complexity, so as to improve the efficiency of the algorithm. For improving the safety of mobile robot, the novel algorithm adopted adaptive cost function to improve the ability of road safety discrimination. A Slide-Rail corner adjustment method was designed to reduce unnecessary corners and improve the smoothness of the path. The simulation results showed that, compared with the traditional A* algorithm and various improved A* algorithms, the proposed algorithm can shorten the path length and searching time, reduce the number of turns, the total turning angles and search nodes. Compared to A*, the searching time was reduced by 64.76%, the total turning angles were reduced by 56.34%, and the search nodes were reduced by 82.76% averagely in test maps of this paper. Moreover, the average interval width in improved A* was 3.07 and was more than 1.5 times the average interval width in Rectangle Expansion A*.

Keywords: Mobile robot; Path planning; Improved A*; Extended neighborhood; Adaptive cost function

1. Introduction

With the rapid development of intelligent manufacturing and commerce economy, the demand for highly adaptable, flexible and intelligent mobile robots is growing in spurts. For example, there are sweeping robots for cleaning tasks indoors, robots for sowing seeds or harvesting fruits in fields, robots for underwater surveys and logistics robots in warehouses. Especially in all kinds of production line and warehousing logistics, the robot is widely used. However, Path planning has always been a core research problem (Li et al., 2023, Lyons et al., 2023, Qin et al., 2023) in the field of mobile robotics, where the first task is to plan a path for the mobile robot to travel efficiently and stably according to the specific conditions and environment. And

Path planning algorithm is mainly divided into two categories in Robot Operating System(ROS): global path planning and local path planning. Global path planning is mainly focused on planning a path without obstacles on a static whole map before starting a job, and local path planning is mainly focused on updating the local map to avoid obstacles while working (Quigley et al., 2009). Traditional global path planning algorithm includes artificial potential field algorithm (Warren, 1989), RRT(Rapidly-exploring Random Tree) algorithm (LaValle, 1998), Dijkstra algorithm (Dijkstra, 1959), RRT* algorithm (Karaman & Frazzoli, 2011), A* algorithm (Hart, Nilsson, & Raphael, 1968), etc. Cluster intelligent global planning algorithm includes genetic algorithm (Hu & Yang, 2004), ant colony algorithm (Brand, Masuda, Wehner, & Yu, 2010), and particle swarm optimization algorithm (Zhang, Gong, & Zhang, 2013).

A large number of researchers carried out path planning improved methods. For example, The proposed GLS(Generalized Laser simulator) (Muhammad et al., 2021) algorithm builds all feasible paths between any arbitrarily selected start and goal locations in a discrete gridded environment within the first stage. It follows by an optimal collision-free path determination stage through reducing the number of path points and selecting the optimum path. However, the path quality is still not great, with redundant nodes and lots of corners. The proposed algorithm (Li, Huang, Ding, Song, & Lu, 2022) includes the introduction of a bidirectional alternating search (BAS) strategy to improve the search path efficiency. Moreover, the path length and turning angle are reduced by filtering the path nodes. But the method of filtering the path nodes was based on the situation without being blocked by obstacles, and the method will be ineffective if the path was blocked by obstacles. Chong et al., 2020, Zhang et al., 2016 introduced the rectangular extended node mode of rectangular symmetry reduction(RSR) algorithm (Harabor & Botea, 2010), and proposed rectangle expansion A* (REA*) algorithm, which took octile distance as h value. Then according to the characteristics of octile distance on the grid map, they proved the path optimal method suitable for octile distance, and designed the relative octile distance iteration strategy. However, there were problems such as long search time and extended redundancy of the rectangle on complex map. Besides, the performance of the algorithm was more purposeful and effective, but it lacked safety considerations. Yangqi Ou et al. (Ou, Fan, Zhang, Lin, & Yang, 2022) proposed an turning weight A* (TWA*) algorithm, which introduced the path smoothing strategy, eliminate redundant points and minimum corner points, effectively improved the smoothness of the path, and introduced the steering cost model and adaptive cost function to improve the search efficiency, made the search more purposeful and effective. However, the effectiveness of its adaptive function is not very obvious. Paper (Xu, Yu, Wang, Zhao, Chen et al., 2023) proposed more neighborhoods A* with Floyd algorithm (NF-A*)that expanded the 8 neighborhoods to a 48 neighborhoods, increasing the search direction and improving the efficiency of path planning. The Floyd algorithm was integrated to smooth the planned path and reduced the turning points in the path. However, the algorithm computation was clearly increased from 8 neighborhoods to 48 neighborhoods and the computation of Floyd algorithm was also large. Paper (Singh & Nagla, 2023) proposed Time Optimized A*(TOA*) approach is a bidirectional technique that select the shortest path between two points that may be from start to goal or from goal to start, based on the number of obstacles present nearby the start or goal node. However, this method is only more efficient on some certain maps. Paper (Chen, Yang, Chen, & Feng, 2023) proposed safety distance A* with Floyd algorithm (SF-A*). The robot safety distance was considered, and the Floyd algorithm was used to improve the A* algorithm. But the computation of Floyd algorithm was too large. Paper (You, Shen, Huang, Liu, & Zhang, 2023) proposed a multi-scenario adaptive A* algorithm based on extended neighborhood priority search (ENMSA-A* algorithm) that designed an improved A* algorithm that adaptively adjusts the search weight of the heuristic function and extended 16 neighborhoods to enhance the guidance and search

speed. Then, a better route can be selected to obtain a relatively smooth path by judging the distance and connectivity between each four turning points. However, the algorithm computation was clearly increased from 8 neighborhoods to 16 neighborhoods and the redundant node deletion strategy would consume a lot of time because there were many path-points in the path.

From the above papers, it can be seen that the improvement methods of A* algorithm mainly include distance optimization of the distance calculation function, neighborhood expansion and path smoothing. The improved A* algorithms in the existing papers will be superior to the traditional A* algorithm in some performances, but they are not guaranteed to be applicable to any map. In addition, some of the improved algorithms have higher requirements on the hardware and software performance of the mobile robot have high requirements, as shown in Table 1.

In this paper, we consider the width of path during robot operation and proposed an improved A* algorithm, which can select wider passages to improve safety and stability of the mobile robot. The main contributions are as follows.

Table 1. Path planning performance metrics of the four algorithms on the jagged map.

Algorithm	**search efficiency**	**path optimization**	**path smoothness**	**node reduction**	**safety**
GLS	weak	weak	weak	weak	weak
BAS-A*	weak	moderate	weak	moderate	weak
REA*	strong	moderate	moderate	strong	weak
TWA*	strong	weak	moderate	moderate	weak
F-NA*	weak	moderate	weak	weak	weak
TOA*	moderate	moderate	moderate	moderate	weak
F-SA*	weak	weak	weak	weak	moderate
ENMSA-A*	weak	moderate	strong	weak	weak

(1) The map is rectangular explored in two directions from the starting point and the target point respectively. The exploration generates fewer nodes to save search time.

(2) The new algorithm uses Euclidean distance as distance estimate and a new operator is designed to seek the best search node to reduce the time complexity, in order to improve the efficiency of the algorithm.

(3) For improving the safety of mobile robot, the novel algorithm designs an adaptive cost function to improve the ability of road safety discrimination.

(4) A Slide-Rail corner adjustment method is designed to reduce unnecessary corners and improved the smoothness of the path to improve efficiency of mobile robots' work.

2. Rectangle expansion A* algorithm

A grid map is one of the most popular types of maps used to path planning in literature (Li, Li, Rong, & Zhang, 2020). If (in 8-connected grid maps) a point looks for a path on a grid map with obstacles consisting of blocked units and passable areas consisting of unblocked units, it can move from any unblocked grid center to another cardinally or diagonally if at least one of two adjacent cardinal directions is unblocked.
REA* algorithm is a improved A* algorithm. The inspired function is

$$F = g + h \tag{1}$$

F is the estimated cost value from the starting point to the goal point via current searching node. g is the actual cost value of the starting point to current searching node. h is estimated cost value of current searching node to the target point.
The octile distance (Kumar, Vámossy, & Szabó-Resch, 2016) is used to calculate the distance between two units heuristically. The length of cardinal move is 1 and the length of diagonal move is 1.414. All the grid points are stored in a matrix of the same size as the map. The point *p(x,y)* is at the intersection of x-th column and y-th row in the grid map. The octile distance between *p(x,y)* and *p'(x',y')* is

$$octile\,(p, p') = 1.414 \times min\,(\triangle x, \triangle y) + |\triangle x - \triangle y| \tag{2}$$

Where $\triangle x = |x - x'|$ and $\triangle y = |y - y'|$

Search nodes in REA* are passable intervals, instead of individual units in the grid map. The main contribution of REA* is that, REA does not explore individual units one by one. Instead, a linear search node will expand an passable rectangle. Only boundary units of the passable rectangle will be visited, and units inside the rectangle will not be visited. Then, boundaries of the rectangle (except the original search node) are pushed outward to generate successor search nodes. Successor search nodes will be inserted into the open list with the minimum f value of units in the search node. The best search node is obtained and the above process is repeated until an optimal path is found.

3. A safety bi-direction rectangle expanded A* algorithm

Due to the increased complexity of the map, REA* made redundant extension nodes when there were many intervals presented. It would lead to longer searching time. Besides, in REA*, octile distances calculated inside unobstructed rectangles would result in longer path with unnecessary corners. And REA* did not consider the width of a path and the robot would not be led to a wider path, which reduces the safety of the robot operation. Moreover, the path planned by REA* was not smooth enough and trajectory optimization can help further smooth it. Thus, in view of the above problems, we proposed the following improvements.

3.1. Bi-direction neighborhood extension

In path planning algorithms, neighborhood extension is indispensable, which produces child nodes so that the algorithm finds the next path point. The extension rule of the traditional A* algorithm was that the eight neighbors around their parent nodes were the child nodes. The

single extension took little time but the process would be repeated on each extended node and much time was taken. In this paper, a rectangular extension method was introduced and improved in the RSR algorithm. The eight-neighbor expansion mode of A* algorithm was converted into bidirectional rectangular expansion, which greatly reduced the number of extensions and the expansion nodes, thus speeded up the algorithm.

Two-way rectangle expansion was executing in the horizontal direction from the starting point S and the target point G, and expanding was not stopped until an obstacle or a map boundary was encountered, as shown in Fig. 1(a). The resulting passable interval (PI) with blue was obtained. Then, the PI was expanding in the vertical direction. Similarly, expanding was stopped until an obstacle or a map boundary was encountered, and a passable rectangle (PR) was obtained, as shown in Fig. 1(b). There were not any obstacles in the rectangle. Because once the PI hit an obstacle, the expanding stopped. The results in Fig. 2 further support this statement. From Fig. 2, the two purple rectangles stop searching early due to the new obstacle, thus proving that there is no obstacle in the rectangles. Then the part of the rectangle boundary which was not the map boundary or the obstacle was regarded as the next PI, which was shown as gray in Fig. 1(c). Similarly, the expansion at point G was shown in Fig. 1(d).

Point S1 and point G1 in the Fig. 1(d) were path points, obtained by the method introduced in 3.2. The PI where the point S1 was located was extended for the next time. The direction of the extension depended on the position of the PI at the rectangular boundary. For example, the PI where point S1 was located was at the upper boundary of the rectangular boundary, so the expansion direction was upward, as shown in Fig. 1(e). Similarly, the PI where point G1 was located was at the left boundary of the rectangular boundary, so the expansion direction was to the left, as shown in Fig. 1(f). At this time, there were overlapping regions of the rectangles that were extended in two directions, then the search was stopped and the path was $S \rightarrow S1 \rightarrow S2 \rightarrow G2 \rightarrow G1 \rightarrow G$. Pseudo-code of Bi-direction neighborhood extension is shown in Algorithm 1.

Algorithm 1 Bi-direction neighborhood extension

```
Input:  S:the start point, G:the target point;
rect1 = original rectangle expanded by S;
rect2 = original rectangle expanded by G;
if rect1 ∩ rect2 ! = ∅ then
   return "path is found";
end if
while Openlist ! = ∅ do
   best1 = the best search node from S to G;
   best2 = the best search node from G to S;
   rect1 = rectangle expanded by best1;
   rect2 = rectangle expanded by best2;
   if rect1 ∩ rect2 ! = ∅ then
      return "path is found";
   end if
end while
return "no path is found";
```

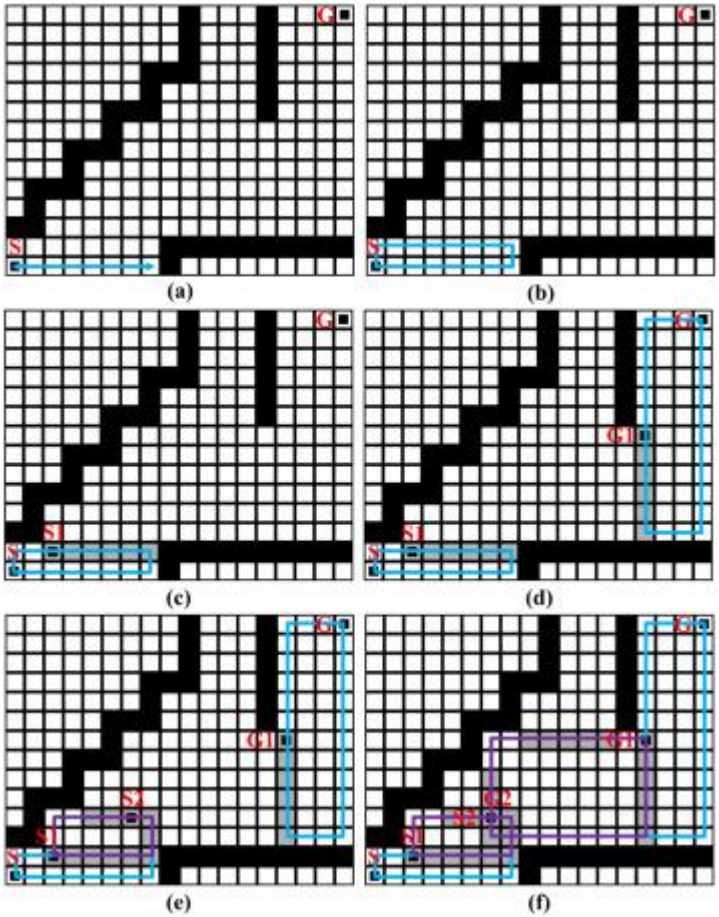

Fig. 1. Bi-direction neighborhood extension on jagged map.

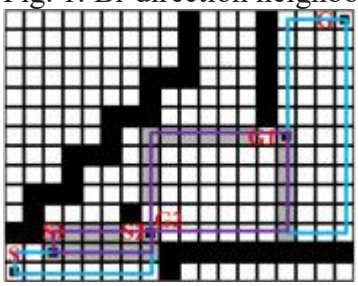

Fig. 2. Bi-direction neighborhood extension on jagged map with a island obstacle.

3.2. Study of the optimal search node operator

Algorithms use raster map as search maps, generally, and their h value was calculated by octile distance. For example, the REA* algorithm was to use octile distance as h value, and the corresponding iterative strategy was designed by the characteristics of octile distance on raster maps. However, the PI would be traversed to find the search node with the minimum F value. The traversal operation increased the time complexity of the operator. And octile distance was not the best choice for the algorithm with rectangular expansion as the neighborhood expansion (RE algorithm) in this paper. Because there are only eight directions in the octile distance. But in order to achieve the new optimal search node operator, more than eight directions were needed. As shown in Fig. 3, there are more directions in Euclidean distance, so the octile distance was replaced by the Euclidean distance in this paper.

To reduce the time complexity of the operator of finding the search node with the minimum F value, an optimal search node operator was designed in this paper. There was a current best search node in each of the two directions because of Bi-direction neighborhood extension.

These two nodes were named best1 and best2 respectively. The line between best1 and best2 was named line m. Besides, the PI was named line n. The two endpoints of the PI were named end1 and end2 respectively.

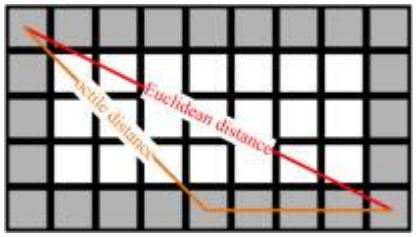

Fig. 3. Comparison of octile distance and Euclidean distance in a PR.

Then, Lines m and n were judged to intersect or not. And if there was an intersection, as shown in Fig. 4(a), the intersection was the node with minimal f value in this PI and its f value was calculated. If there was no intersection, as shown in Fig. 4(b), the distance from end1 and end2 to line m were calculated and named d1 and d2. The smaller of d1 and d2 was named dmin. The endpoint(end1 or end2) with dmin was the node with minimal f value in this PI and its f value was calculated. Fig. 4(c) shows the two cases exist at the same time, the two PIs did not affect each other and found their nodes with minimal f value respectively. Fig. 5 shows the demonstration process of the best search node operator on the complete map. Pseudo-code of the optimal search node operator is shown in Algorithm 2.

Algorithm 2 the optimal search node operator

```
Input:    rect1,best1,best2;
/*only for rect1, and rect2 is omitted here*/
for all PI ∈ rect1 do
   end1,end2 = two endpoints of PI;
   n = the line between end1 and end2;
   m = the line between best1 and best2;
   if m ∩ n ! = ∅ then
      it = the intersection of m and n;
      PI.fvalue = it.fvalue;
   else
      d1 = distance(end1, m);
      d2 = distance(end2, m);
      dmin = min(d1, d2);
      it = end-i with dmin;
      PI.fvalue = it.fvalue;
   end if
end for
```

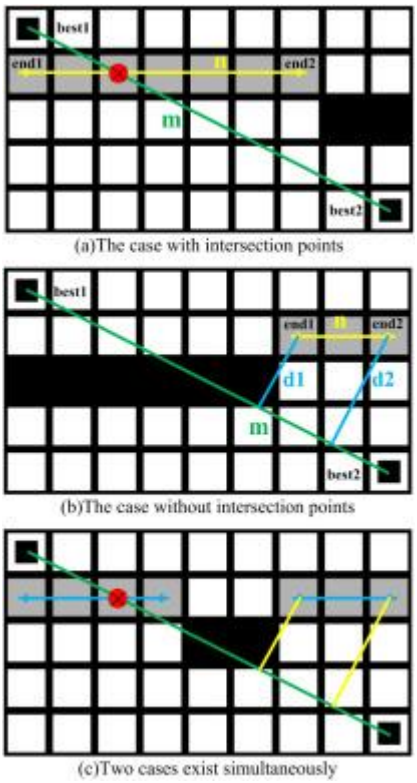

Fig. 4. The optimal search node operator.

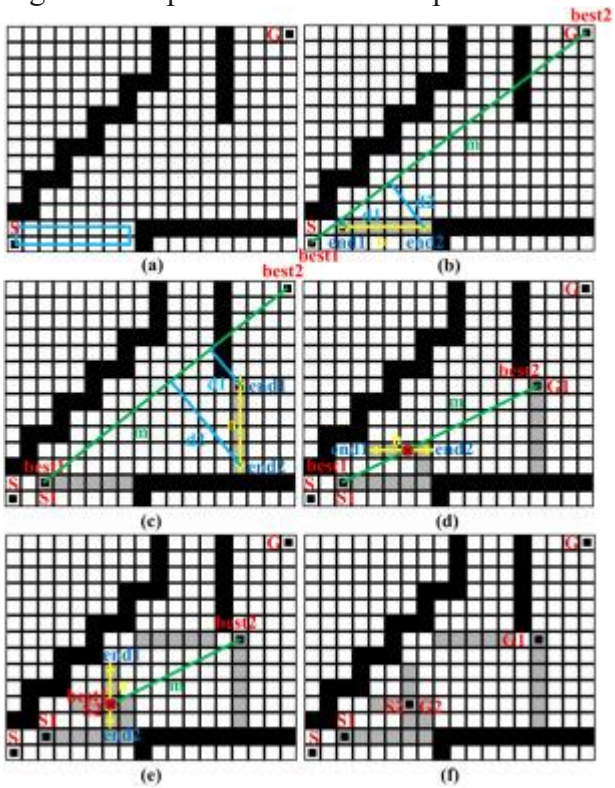

Fig. 5. The demonstration process of the best search node operator on the complete map.

3.3. Adaptive cost function

The inspired function in the traditional A* algorithm is formula (1). In maps with cluttered obstacles, the paths of conventional A* algorithms often present problems. For example, the path passes through narrow passages or even causes the robot to be unable to make turns

(Nakamura et al., 2023a, Nakamura et al., 2023b). For the safety of mobile robots in operation, this paper designed an adaptive cost function to enhance the discrimination of road passability safely and improve the safety of robots in operation.
The width of the current PI and the sum of the widths were calculated. And the main part of the influence coefficient of the cost function were the ratio of the width and the sum. The algorithm with the cost function more inclined to choose wider roads for path planning. In order to avoid the too large influence range of the ratio factor, the weight ratio was tested. Its cost function was

$$F = g + w \cdot h \tag{3}$$

$$w = r + (1 - r) \cdot \frac{\sum_{i=1}^{n} a_i}{n \cdot a_c} \tag{4}$$

where $w>0$ was the weight coefficient, r was the weight ratio and $r \in [0,1]$ because of $w>0$. Where a_i was the width of the ith interval and a_c was the width of the current interval.
When r is 0, the weight coefficient would be completely decided by the interval width, which may lead to too fast convergence and bad path. When r is 1, the adaptive function would not be used. The range from 0 to 1 was divided into ten equal parts, each of which was used to test the raster map and finally find the ratio with the best comprehensive performance. The optimal range of the weight ratio was [0.5, 0.7] in the raster map. Therefore, in the experiments of this paper, r was set to 0.7.

The comparison of the performance of the algorithms was shown in Fig. 6, where Fig. 6(a) was SBREA* without adaptive cost function and Fig. 6(b) was SBREA* with adaptive cost function. Pseudo-code of the adaptive cost function is shown in Algorithm 3.

Algorithm 3 the adaptive cost function

Input: best1,best2,r;
1: **for** all PI **do**
2: $asum = asum$ + PI.$width$;
3: **end for**
4: n = the number of all PI;
5: $w = r + (1 - r) \cdot \frac{asum}{n \cdot a_c}$;
6: h(best1,best2) = $sqrt$(best1^2 + best2^2);
7: F = best1.$gvalue$ + best2.$gvalue$ + $w \cdot h$(best1,best2);
8: **return** F;

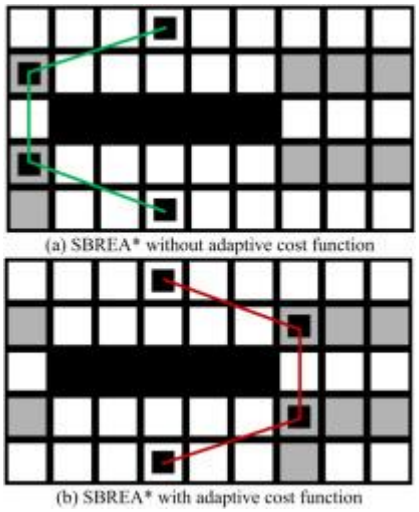

Fig. 6. Comparison of performance tests of local maps.

3.4. Slide-rail corner adjustment method

The path planned by the conventional path planning algorithm were always redundant or containing too large corners. And the conventional measure was simply to make a line between two points and determine whether the line collided with an obstacle.

As shown in Fig. 7, the existing path planning algorithm usually determined whether the line A to C was used as the optimization target for the path quality. If the line did not collide with an obstacle, it was the optimization target and the algorithm continued to find the next optimization target. If a collision occurred, the line was not used as the optimization target and the path ABC was kept without any change. Then the path still had too many corners, which was not good for the actual operation of the mobile robot.

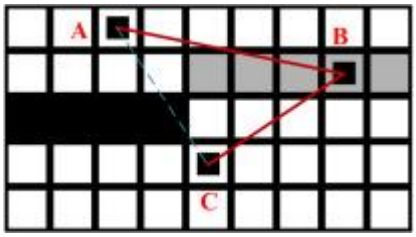

Fig. 7. Adjustment method of conventional corner point.

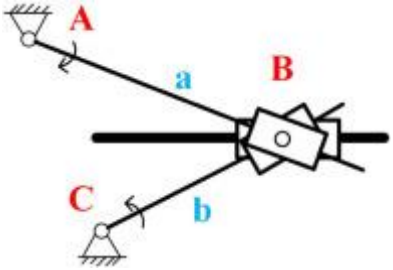

Fig. 8. Tangent mechanism.

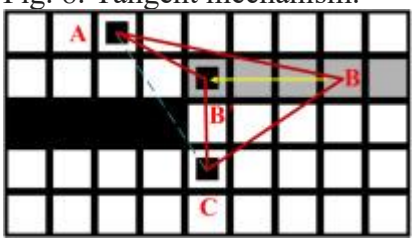

Fig. 9. Slide-Rail corner adjustment method.

In order to overcome the above shortcomings, the paper proposed a Slide-Rail corner adjustment method. The method transformed the problem of excess corners in the path into an angle problem in the tangent mechanism (Zhonghe, Zhaohui, & Smith, 2004). Fig. 8 shows the tangent mechanism. The points A, B and C in Fig. 8 were equivalent to the points A, B and C in Fig. 7, respectively. The point B was regarded as a slider in the tangent mechanism. In this adjustment method, the point B in Fig. 7 can only move in a straight line within the PI, just as the slider B in the tangent mechanism can only move in a straight line in Fig. 8. The larger the ∠ABC in the algorithm, the better the path.

In Fig. 8, Point C was in the clockwise direction of rod a and point A was in the counterclockwise direction of rod b, so the rod a rotated clockwise and the rod b rotated counterclockwise. The slider B was subject to the partial force along the frame from these two rods, so the slider B will move to the left along the frame and the ∠ABC will be larger and larger. For the same reason, as shown in Fig. 9, point C was in the clockwise direction of line AB and point A was in the counterclockwise direction of line BC, so point B will move to the left along the PI. During the movement of point B, the path ABC never hit an obstacle. Because points A and B were in the same PR, and points B and C were also in another PR. And it is clear from 3.1 that there are no obstacles in any PI and PR. Then, the point B stopped moving and the value of ∠AB'C was obtained when the point B got the end of the PI. If ∠AB'C <180∘, the best optimized target point B' was obtained. Therefore, the quality of the path is better than ABC, and the optimization result is AB'C. Otherwise, there must be a ∠AB"C =180∘ because of ∠ABC <180∘ and ∠AB'C >180∘. So point B was abandoned, and the path ABC would become the path AC. Pseudo-code of the Slide-Rail corner adjustment method is shown in Algorithm 4.

Algorithm 4 the Slide-Rail corner adjustment method

Input: path;
1: **for** all pathpoint∈path **do**
2: A = the previous pathpoint;
3: B = the current pathpoint;
4: C = the next pathpoint;
5: Rdir = direction of rotation from $\overline{AB}$ to $\overline{AC}$ in △ABC;
6: $B.dir_{move}$ = Rdir;
7: B' = endpoint in the $B.dir_{move}$ direction of this PI;
8: **if** ∠AB'C < 180° **then**
9: pathpoint = B'
10: **else**
11: delete pathpoint
12: **end if**
13: **end for**
14: **return** new path;

3.5. SBREA* algorithm

At the beginning of the SBREA* algorithm, two pairs of open and close lists were created to hold the nodes searched in the two directions. The start point and the goal point would be placed in two different open lists. The algorithm would determine if the two open lists were empty. If one of the open lists was empty, the path finding failed and the procedure ended. If the open lists were not empty, the best points (Bests) with the f-min value in two open lists were selected and were put into two different close lists. Subsequent rectangular extensions were made with Bests in both directions. Note that, as mentioned in 3.1, Bests were the start

and goal points in the first extension. Then, Bests will be extended as passable intervals (PI) on the boundary of the rectangle. It was determined whether the extended rectangles in the two directions had intersecting regions. If there was an intersecting region, the algorithm stopped source searching and path finding. The path was optimized by the Slide-Rail corner adjustment method proposed in 3.4. But if there was no intersecting region, the algorithm tried to search PIs on the boundary of the extended rectangle were obtained. If the PI searching was success, the PIs would be traversed and a point of a PI with f-min value was found by the method in 3.2. And the g value of the point would be calculated as the g value of PI. If the PI searching failed, no PI was obtained in the extension and the algorithm would return to the open list to reselect another Best.

Subsequently, there were some situations. When the PI was not in the open list nor in the close list, the PI was added directly to the open list. When the PI was in the open list or not in the open list but in the close list and the g value of the PI was less than the g value of the old PI, it was necessary to delete the old PI and then add the PI to the open list. A PI was not added to the open list when it was in the open list or in the close list instead of the open list and the g value of the PI was greater than or equal to the g value of the old PI. For PIs newly added to open list, the adaptive weighting function of 3.3 was used to calculate the f value of them. After traversing PIs, the algorithm continued back to the open list, continued to find the next Best, and started the loop. The general flow of the improved A* algorithm is shown in Fig. 10.
Pseudo-code of the SBREA* algorithm is shown in Algorithm 5. The S is the start point, and the G is the target point. The best1 and the best2 are the points with minimum fvalue in Openlist1 and Openlist2 respectively. The opnodes1 is the node with minimum fvalue in a PI.

The OPnodes1 is the set of all opnodes1.

Algorithm 5 the SBREA* algorithm

```
Input:  map, S, G;
best1 = S , best2 = G;
Openlist1 = [best1], Openlist2 = [best2];
Closelist1 = [], Closelist2 = [];
rect1 = original rectangle expanded by best1;
rect2 = original rectangle expanded by best2;
if rect1 ∩ rect2 ! = ∅ then
   return  "path is found";
end if
while Openlist1 ∩ Openlist2 ! = ∅ do
   /*only for rect1, and rect2 is omitted here*/
   OPnodes1 = the optimal search node operator(rect1, best1, best2);
   for each opnodes1 ∈ OPnodes1 do
      if opnodes1 ∈ Openlist1 or opnodes1 ∈ Closelist1 then
         if g_opnodes1 < old g_opnodes1 then
            old g_opnodes1 is replaced by g_opnodes1 in the Openlist1;
            f_opnodes1 = calculate the f value(opnodes1);
         end if
      else
         g_opnodes1 is added in the Openlist1;
         f_opnodes1 = calculate the f value(opnodes1);
      end if
   end for
   best1 = min(point_fvalue) in Openlist1;
   best2 = min(point_fvalue) in Openlist2;
   Closelist1 = [Closelist1; best1];
   Closelist2 = [Closelist2; best2];
   rect1 = rectangle expanded by best1;
   rect2 = rectangle expanded by best2;
   Openlist1 delect best1;
   Openlist2 delect best2;
   if rect1 ∩ rect2 ! = ∅ then
      return  "path is found";
      Break;
   end if
end while
```

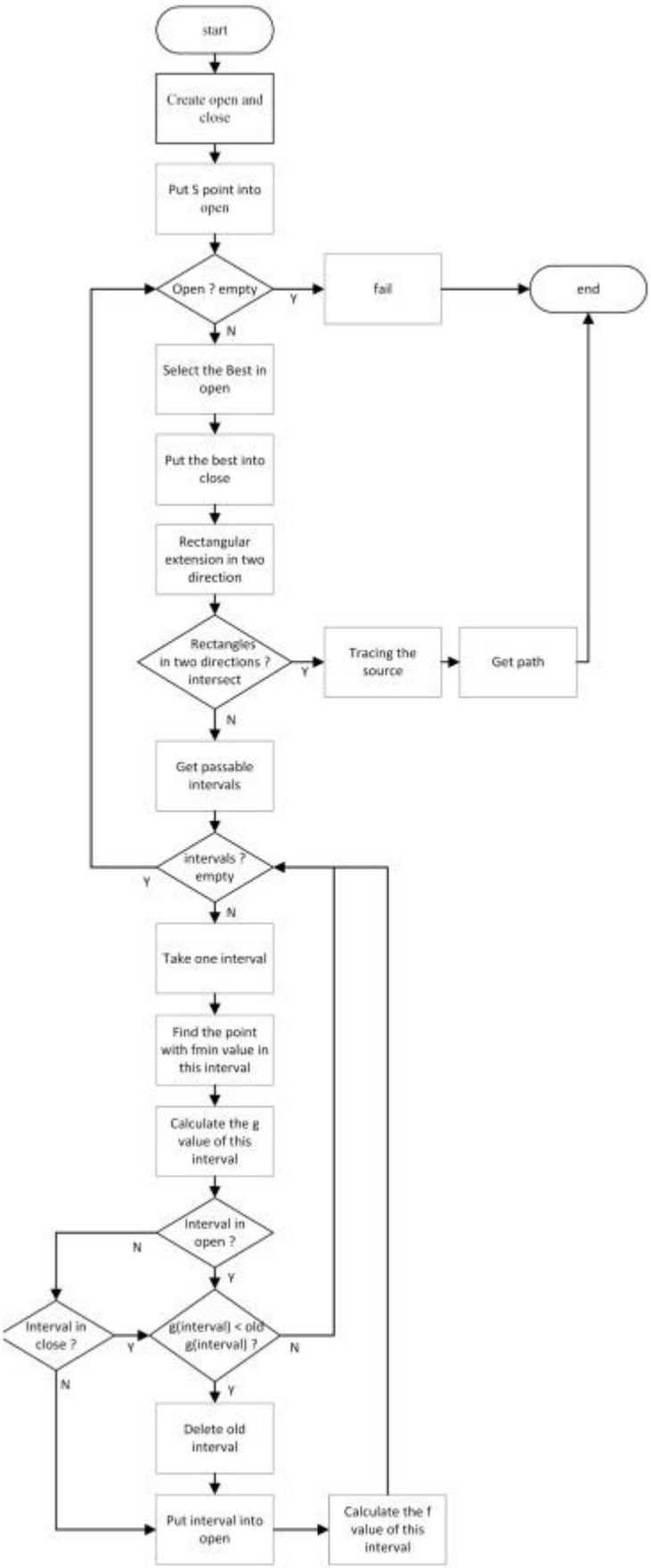

Fig. 10. The general flow of the improved A* algorithm.

After analyzing the SBREA* algorithm, it has the same algorithmic complexity as the A* algorithm. First, when expanding the neighborhood, the A* algorithm searches from the parent node to the surrounding eight neighborhoods, and this process has a complexity of $O(8)$. The SBREA* is expanding outward in a rectangular manner. Assuming that the length and width of the rectangle is $a*b$, the complexity of this process is $O(a*b)= O(1)$. Secondly, the A* algorithm expands $O(n)$ times, so there are $k*n$ nodes added to the openlist. The algorithmic complexity of this process is $O(n)$. The SBREA* is similar to A* and expands n rectangles, which produces $k*n$ PIs to be added to the openlist, so its algorithmic complexity of this process is also $O(n)$.. Finally, both the A* algorithm and the SBREA* algorithm need to find the node with the minimum fvalue from the openlist as the parent node for the next expansion, and the algorithmic complexity of this process is $O(n)$. In summary, the complexity of both the A* algorithm and the SBREA* algorithm is $O(n^2)= O(1)* O(n)* O(n)$. Besides, neighborhood extension of SBREA* in open areas may inadvertently lead to an expanded search range, but the number of its extensions is reduced. In contrast, neighborhood extension of A* lead to a small range, but A* has an increased number of extensions. Thus, neighborhood extension of SBREA* in an open areas expand the search range, which does not affect the efficiency of the SBREA* algorithm.

4. Simulation experiments and result analysis

4.1. Introduction of the simulation map

To verify the effectiveness of the SBREA*, this paper conducts comparison experiments on different types of raster maps on MATLAB R2018b. The more different types of maps an algorithm can work well on, the more effective the algorithm is. Map a~j were Fig. 11(a)~(j) respectively. The size of map a is 14*18 and the others are 40*40. And these maps represent scenarios that mobile robots often encounter in practice. Map a was a jagged map. Jagged maps are detrimental to the algorithm of the rectangular expansion method(RE algorithm). Map b~c was passage-type maps. Activity area of mobile robot was smaller in the passage map, so the gap between traditional A* and RE algorithm became smaller. Map b contained a circular obstacle. And the circular obstacle would become a jagged obstacle that is detrimental to the RE algorithm. There were two passages in map c. The left passage was wider and the right passage was relatively narrow. We prefer to take wider path when the paths are about the same length. Map d~e was maze maps. Maze map is a type of map that has been used in the study of path planning algorithms. Because there were many traps in the maze map. Earlier identification of traps can demonstrate the superiority of an algorithm. Map d was a conventional maze map. Map e was a spiral maze map. Usually, the challenge of quickly searching for a path from one spiral center to another was large.Maps f~h were simple obstacle maps. Simple obstacle maps are the most universal maps in realistic working scenarios. Maps i~j were complex maps with cluttered obstacles. There were a majority of irregularly shaped obstacles in the map, which was detrimental to RE algorithm. And there were many possible paths from the starting point to the target point. Better quality of planned path means better algorithm.

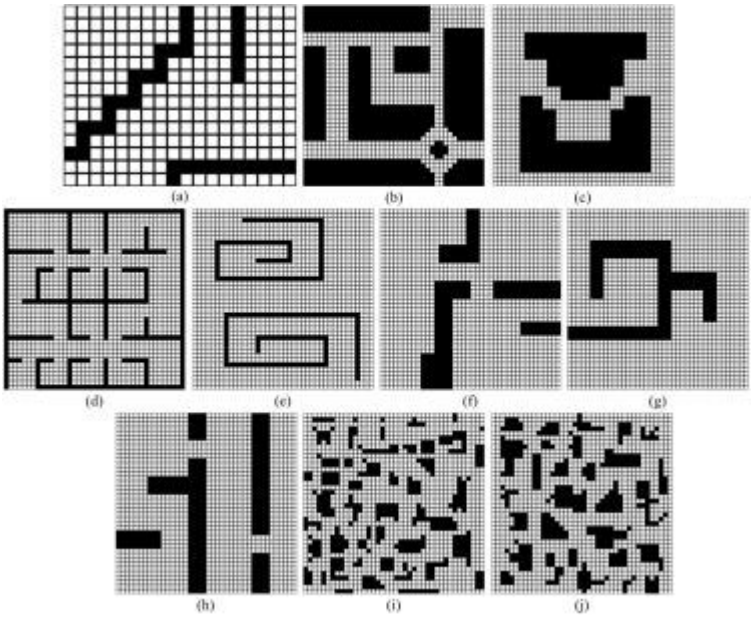

Fig. 11. Different types of raster maps.

4.2. Simulation test of path planning

In this paper, the start point and the goal point were denoted by the blue dot and red dot, respectively. The blue and red line segments represent the planned paths and the gray areas represent the searched nodes. Six performance metrics were set: search time, path length, corners, angles, smoothness, search nodes and average interval width. The search time refers to the time spent on path finding. The search time should not be too long, otherwise it will reduce the efficiency of the algorithm. The path length refers to the total length of the planned path, and it should be the shortest. Liu, Wang, Yang, Liu, Li et al. (2023) The corners and angles respectively refer to the turning times and the total turning angles of the robot during the operation. It is taking time to turn, so the more corners and angles, the more time it takes. The smoothness refers to path smoothness relative to the A* algorithm. The search nodes refers to the number of nodes searched by the algorithm. The process of generating nodes expends search time, so the more nodes there are, the more time is expended. Lai, Wu, Liu, and Zeng (2023) The average interval width refers to the average width of intervals through which the robot will pass. Since we can conclude from the paper (Nakamura et al., 2023a, Nakamura et al., 2023b) that the passage width affects the safety and stability of the mobile robot, we added this performance metric to prove the validity of the adaptive cost function proposed in this paper. In conclusion, the smaller values of search time, path length, corners, angles and search nodes, the better effect of the algorithm. And the larger smoothness rate, the better path smoothness. The larger average interval width, the better safety of the robot.

4.2.1. Tests on a jagged map

Fig. 12 shows the path planning results for the four algorithms A*, TWA*, REA* and SRBEA* on a jagged map respectively. As can be seen from the Fig. 12, A* and TWA* generated a large number of useless search nodes. However, the whole nodes generated by REA* and SBREA* is greatly reduced, thus a large number of useless nodes would be reduced. Further, there are still some corners in Fig. 12(c). The corners did not have a positive effect on the planned path. Too many corners will not only affect the stability of the movement of the mobile robot, but also increase the travel time of the mobile robot. Therefore, the proposed SBREA*

used a bi-directional search method, as shown in Fig. 12(d), which finds smoother paths and generates fewer search nodes.

Table 2 represents the path planning performance metrics of the four algorithms on the jagged map. As can be seen from the Table 2, the number of search nodes in SBREA* was 25 and reduced by 79.7%, 70.9% and 49.0% compared with A*, TWA*, and REA*, respectively. The efficiency of the algorithm was improved greatly. Besides, the angles in SBREA* was 36.87 and reduced by 86.3%, 83.6% and 72.7% compared with A*, TWA*, and REA*, respectively. The path smoothness was improved greatly. Besides, although TWA* had the least search time, the rest of the performance metrics were worse than SBREA*.

Overall, A* and TWA* had the longest path and REA* contained non-essential turns. However, SBREA* not only had the shortest path, but also reduced the non-essential turns and improved the path smoothness. The comprehensive performance of SBREA* is better than A*, TWA*, and REA*.

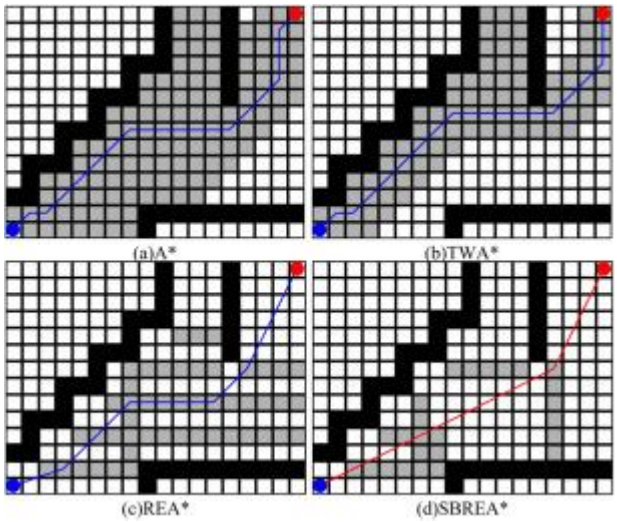

Fig. 12. Path planning results of the four algorithms on the jagged map.

Table 2. Path planning performance metrics of the four algorithms on the jagged map.

Algorithm	search time	path length	corners	angles	smoothness	search nodes
A*(a)	8 ms	24.14	6	270.00°	1	123
TWA*(b)	3 ms	24.14	5	225.00°	1.2	86
REA*(c)	5 ms	23.36	4	135.00°	2	49
SBREA*(d)	4 ms	22.36	1	36.87°	7.32	25

4.2.2. Tests on passage-type maps

Fig. 13 shows the path planning results for the four algorithms A*, TWA*, REA* and SBREA* on two passage-type maps respectively. They represented scenarios where obstacles occupied most of the area and the remainder were roads. Obviously, REA* and SBREA* had far fewer search nodes than A* and TWA*. However, there were noticeable corners at ① and ② in Fig. 13(c), resulting in an unsmooth path. Therefore, the SBREA* proposed in this paper used a slide adjustment method, as shown in Fig. 13(d), to reduce the angle of the corner at ① and

also eliminates the corners at ②. Besides, REA* chose the relatively narrow passage on the right. The narrow passage can affect the stability and safety of the robot's operation. Therefore, the proposed SBREA* used an adaptive cost function, as shown in Fig. 13(h), to select the wider passage on the left.

Table 3 represents the path planning performance metrics of the four algorithms on two passage-type maps respectively. As can be seen from the Table 3, the average number of search nodes in SBREA* was 60 and reduced by 82.1% and 72.7% compared with A* and TWA*, respectively. Besides, the average angles in SBREA* was 136.13 and reduced by 71.2%, 62.2% and 59.1% compared with A*, TWA*, and REA*, respectively. In addition, SBREA* sacrifices only a small amount of time for better overall performance.

Overall, A* had too many corners. TWA* had the longest path. REA* contained non-essential turns and chose a narrow passage. However, SBREA* not only had the shortest path, but also reduced the non-essential turns and selected a wider passage. The comprehensive performance of SBREA* is better than A*, TWA*, and REA*.

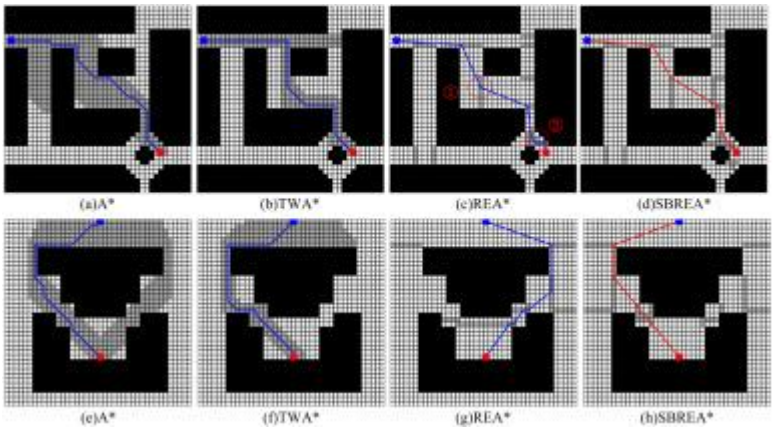

Fig. 13. Path planning results of the four algorithms on passage-type maps.

Table 3. Path planning performance metrics of the four algorithms on passage-type maps.

Algorithm	search time	path length	corners	angles	smoothness	search nodes
A*(a)	21 ms	47.21	10	495.00°	1	280
TWA*(b)	10 ms	51.31	5	315.00°	1.57	187
REA*(c)	12 ms	47.14	6	338.13°	1.46	46
SBREA*(d)	19 ms	44.84	4	162.44°	3.05	60
A*(e)	26 ms	45.87	9	450.00°	1	390
TWA*(f)	12 ms	48.21	7	405.00°	1.11	253
REA*(g)	8 ms	45.48	6	327.43°	1.37	49
SBREA*(h)	14 ms	43.89	2	109.82°	4.10	60

4.2.3. Tests on maze maps

Fig. 14 shows the path planning results for the four algorithms A*, TWA*, REA* and SBREA* on two maze maps respectively. One represented a scene with a large number of traps and another represents a scene with spiral obstacles. As there were many traps or spiral obstacles,

A* and TWA* generated a large number of search nodes. Especially, the search nodes took up almost the entire map in spiral maze map. But REA* and SBREA* are both RE algorithm. The RE algorithm can identify these traps quickly. Thus, the search nodes in REA* and SBREA* were far fewer than A*. In addition, REA* was still an unsmooth path similar to A* or TWA*. The SBREA* proposed in this paper used a bi-directional search method and used Euclidean distance to calculate h-value, as shown in Fig. 14(d), which ultimately selected another path different from the other three algorithms. Besides, a corner was seen at ① in Fig. 14(g). The proposed SBREA* used a slide adjustment method, as shown in Fig. 14(h), to reduce the angle of the corner and make the path smoother.

Table 4 represents the path planning performance metrics of the four algorithms on two maze maps respectively. As can be seen from the Table 4, the average number of search nodes in SBREA* was 127.5 and reduced by 87.7%, 84.1% and 4.9% compared with A*, TWA*, and REA*, respectively. Besides, the average angles in SBREA* was 597.34 and reduced by 30.1%, 35.2% and 13.3% compared with A*, TWA*, and REA*, respectively. In addition, Moreover, although the SBREA*(h) path length is only a little bit shorter than A*(e), the search time and search nodes are much less than A*(e).

Overall, A* and TWA* had too many corners. REA* contained non-essential turns. However, SBREA* not only had the shortest path, but also reduced the non-essential turns. The comprehensive performance of SBREA* is better than A*, TWA*, and REA*.

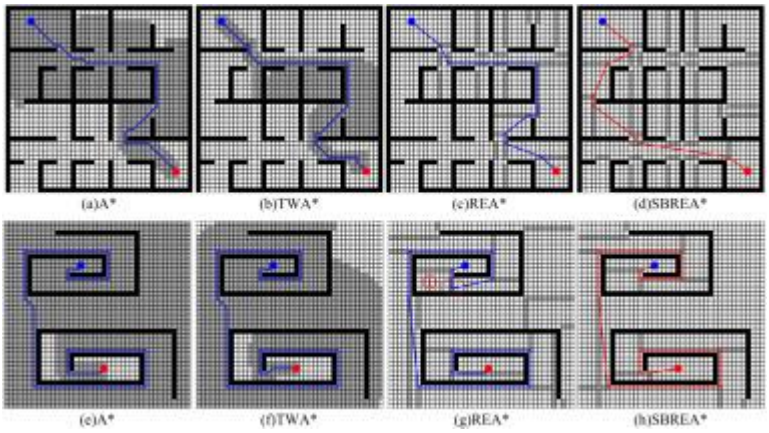

Fig. 14. Path planning results of the four algorithms on maze maps.

Table 4. Path planning performance metrics of the four algorithms on maze maps.

Algorithm	search time	path length	corners	angles	smoothness	search nodes
A*(a)	94 ms	64.70	12	630.00°	1	731
TWA*(b)	24 ms	66.46	12	720.00°	0.86	401
REA*(c)	42 ms	65.47	8	450.95°	1.4	113
SBREA*(d)	58 ms	61.88	7	303.65°	2.07	161
A*(e)	322 ms	130.66	14	1080.00°	1	1336
TWA*(f)	171 ms	130.66	15	1125.00°	0.96	1205
REA*(g)	65 ms	131.90	12	926.57°	1.17	155
SBREA*(h)	23 ms	129.35	12	891.03°	1.21	94

4.2.4. Tests on simple maps

Fig. 15 shows the path planning results for the four algorithms A*, TWA*, REA* and SBREA* on three simple maps respectively. It represents a scenario with less obstacles and a more empty map. Apparently, REA* and SBREA* had far fewer search nodes than A* and TWA*. However, there were obvious corners at ① in Fig. 15(c) and ②~③ in Fig. 15(k), resulting in an unsmooth path. The proposed SBREA* used a slide adjustment method, as shown in Fig. 15(d) and (l), to eliminate the corner at ① and reduce the angle of the corner at ②. And then, SBREA* used a bi-directional search method to avoid the appearance of the corner at ③.

Table 5 represents the path planning performance metrics of the four algorithms on three simple maps respectively. As can be seen from the Table 5, the average number of search nodes in SBREA* was 96.3 and reduced by 86.7% and 83.7% compared with A* and TWA*, respectively. Besides, the average angles in SBREA* was 299.73 and reduced by 54.6%, 41.2% and 16.9% compared with A*, TWA*, and REA*, respectively.

Overall, A* had too many corners. TWA* had the longest path. REA* contained non-essential turns. However, SBREA* not only had the shortest path, but also reduced the non-essential turns. The comprehensive performance of SBREA* is better than A*, TWA*, and REA*.

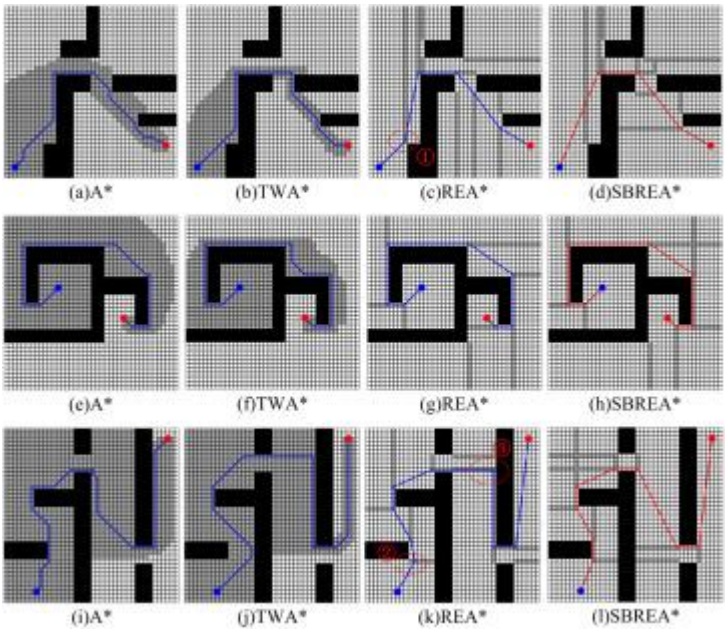

Fig. 15. Path planning results of the four algorithms on simple maps.

Table 5. Path planning performance metrics of the four algorithms on simple maps.

Algorithm	search time	path length	corners	angles	smoothness	search nodes
A*(a)	39 ms	60.53	13	630.00°	1	442
TWA*(b)	15 ms	62.87	7	405.00°	1.56	333
REA*(c)	16 ms	58.23	5	190.28°	3.31	140
SBREA*(d)	12 ms	57.23	3	90.44°	6.97	113
A*(e)	101 ms	74.38	8	540.00°	1	716
TWA*(f)	30 ms	76.73	9	630.00°	0.86	479
REA*(g)	14 ms	73.69	7	450.00°	1.2	75
SBREA*(h)	13 ms	73.69	7	450.00°	1.2	81
A*(i)	197 ms	95.53	15	810.00°	1	1018
TWA*(j)	119 ms	109.18	9	495.00°	1.64	962
REA*(k)	19 ms	98.25	8	442.39°	1.83	75
SBREA*(l)	15 ms	90.83	8	358.75°	2.26	95

4.2.5. Tests on complex maps

In the above simple map, the adaptive cost function was not well represented because of the simple and small number of obstacles. So the following complex obstacle maps were designed in this paper to verify the effect of its adaptive cost function on the path planning algorithm.

Fig. 16 shows the path planning results for the four algorithms A*, TWA*, REA* and SBREA* on two complex maps respectively. It represented scenarios with very many and dense obstacles. As shown in Figures 16(c) and (g), due to the limitations of the rectangular expansion approach on complex maps, the REA* algorithm planed the path with an excessive number of corners, poor smoothness and small interval width. Therefore, the proposed SBREA* used an adaptive weighting function, as shown in figures (d) and (h), to increase the interval width of the path. Further, SBREA* used a slide adjustment method to reduce the corners and improve the smoothness of path.

Table 6 represents the path planning performance metrics of the four algorithms on two maze maps respectively. As can be seen from the Table 6, the average number of search nodes in SBREA* was 177.5 and reduced by 64.5%, 21.3% and 41.9% compared with A*, TWA*, and REA*, respectively. Besides, the average angles in SBREA* was 182.34 and reduced by 77.5%, 76.2% and 79.2% compared with A*, TWA*, and REA*, respectively. Moreover, the average interval width in SBREA* was 3.07 and was more than 1.5 times the average interval width in REA*. Besides, although TWA* had the least search time, the rest of the performance metrics were worse than SBREA*.

Overall, A* had too many search nodes. TWA* and REA* had too many corners. REA* had small interval width. However, SBREA* can break the limitations of the rectangular expansion approach on complex maps. There were the shortest path, the smallest angles and wider interval width in SBREA*. The comprehensive performance of SBREA* is better than A*, TWA*, and REA*.

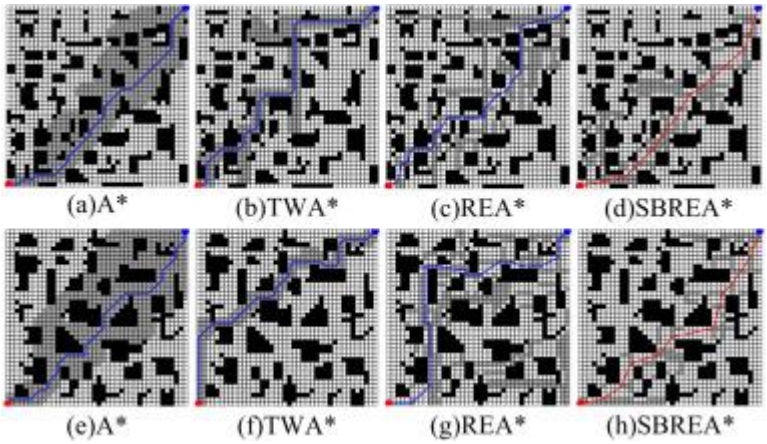

Fig. 16. Path planning results of the four algorithms on complex maps.

Table 6. Path planning performance metrics of the four algorithms on complex maps.

Algorithm	search time	path length	corners	angles	smoothness	search nodes	interval width
A*(a)	52 ms	61.01	15	675.00°	1	443	–
TWA*(b)	29 ms	69.80	17	765.00°	0.88	265	–
REA*(c)	158 ms	66.24	19	949.85°	0.71	272	1.90
SBREA*(d)	114 ms	58.29	6	125.20°	5.39	197	3.06
A*(e)	56 ms	63.36	20	945.00°	1	556	–
TWA*(f)	12 ms	68.87	17	765.00°	1.24	186	–
REA*(g)	302 ms	77.28	15	801.87°	1.18	339	1.78
SBREA*(h)	52 ms	60.64	9	239.48°	3.95	158	3.08

4.3. ROS experiments

To further verify the algorithm in this paper, the SBREA* and comparison algorithm were implemented in Robot Operating System (ROS) on Ubuntu 20.04. The robot is a WHEELTEC four-wheeled car in this experiment, as shown in Fig. 17.

The point cloud data of the obstacle was obtained by LIDAR scanning. The data would be fed back to the ROS controller. The controller calculated the speed information and sent it to the STM32 control board. The control board controlled the four-wheeled car for autonomous navigation.

The ROS provides various interfaces such as global planner and local planner. This experiment only needs to plan the global path according to the target location, so SBREA* path planning algorithm was taken into a new global path planner. The ROS can call the new global path planner to make robot work.

Fig. 17. The WHEELTEC four-wheel car.

Our laboratory was selected as the test environment. The local planner is TEB planner. The 2D construction of the test environment was completed by using the tools such as gmapping and amcl. Rviz was used as a visualization tool to view the navigation paths and set up the navigation tasks. We set the map resolution to 0.05. The results are shown in Fig. 18. REA* was Fig. 18(a) and SBREA* was Fig. 18(b). The green line is the globle path, and the red line is the trajectory of the car. It can be seen that in this navigation task, the path of the REA* had some non-essential turns. Non-essential turns can reduce the efficiency of the robot operation. Besides, REA* passed through narrow areas of the red circle shown in Fig. 18(a), which decreased the stability and safety of the robot during work. However, SBREA* passed through wider areas of the blue circle shown in Fig. 18(b), which effectively avoided the problem of selecting a narrow areas for access in global path planning. To some extent, the problems in paper (Nakamura et al., 2023a, Nakamura et al., 2023b) related to robot stability and safety on narrow roads in locale path planning were alleviated, because the wider load, the safety and stability of robot. Therefore, it is clear that the path planned by SBREA* is superior. The path was not only shorter and wider, but also reduced non-essential turns, which greatly increased the stability, efficiency and safety of the mobile robot.

In Table 7, the running time is travel time of the car from the start point to the goal point. The path length is the total length of the path. The turning angle is the total turning angles. The search node is the number of search nodes searched by the algorithm. It can be seen from Table 7 that although the path lengths of the two algorithms are similar, the car with REA* needs more running time due to the large number of turns. However, the SBREA* selected a better and wider load, so the running time of the car cost less time.

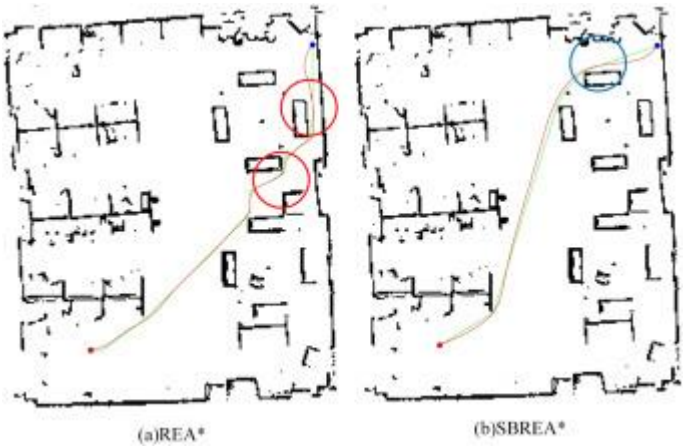

Fig. 18. REA* and SBREA* path planning results on the ROS platform.

Table 7. Performance metrics of the REA* and SBREA* algorithms on the ROS platform.

Algorithm	running time	path length	angles	search nodes
REA*	14.07 s	23.36	135.00°	2848
SBREA*	9.46 s	22.36	36.87°	2669

5. Discussion

This paper proposed an improved A* algorithm. In the novel algorithm, compared with the A* algorithm, the octet neighborhood was replaced a rectangular boundary without obstacles. The map was rectangular explored in two directions from the starting point and the target point respectively. The exploration generated fewer nodes. Due to the enlargement of the search neighborhood and the rectilinear passage without obstacles in the rectangular region, the new algorithm used Euclidean distance as distance estimate. And a new operator was designed to seek the best search node to reduce the time complexity, so as to improve the efficiency of the algorithm. For improving the safety of mobile robot, the novel algorithm adopted adaptive cost function to improve the ability of road safety discrimination. A Slide-Rail corner adjustment method was designed to reduce unnecessary corners and improve the smoothness of the path. Furthermore, we found that the Euclidean distance is more suitable than the octile distance for the A* algorithm that performs neighborhood expansion in a rectangular expansion. Because the inside of extended rectangular was unobstructed and any two points on the rectangular boundary were connectable in a straight line. Also, during the experiments, it was found that the quality of the paths planned by the REA* algorithm was better on map i and map j with Euclidean distance. Moreover, when we implement a bidirectional rectangular extension, the termination condition of searching should be that the extended rectangles in bi-direction have overlapping parts. Otherwise, Two paths would be obtained. By the way, in the experiments, the resolution of map is a property of the raster map, and a proper resolution makes the algorithm fast and accurate in path planning. In general, the resolution of the map is larger and the A* algorithm with rectangle as extended neighborhood is more efficient than the traditional A*.

6. Conclusion

In this paper, an improved A* algorithm was proposed for the long search time and redundant search nodes in the path planning of the traditional A* algorithm in the random obstacle room. In the novel algorithm, compared with the A* algorithm, the octet neighborhood was replaced a rectangular boundary without obstacles. The map was rectangular explored in two directions from the starting point and the target point respectively. The exploration generated fewer nodes. Due to the enlargement of the search neighborhood and the rectilinear passage without obstacles in the rectangular region, the new algorithm used Euclidean distance as distance estimate. And a new operator was designed to seek the best search node to reduce the time complexity, so as to improve the efficiency of the algorithm. For improving the safety of mobile robot, the novel algorithm adopted adaptive cost function to improve the ability of road safety discrimination. A Slide-Rail corner adjustment method was designed to reduce unnecessary corners and improve the smoothness of the path. Simulation results show that the proposed improved algorithm can shorten the time of path finding, reduce corners and total turning angles, and thus search for a better path compared with the traditional A* algorithm and various improved A* algorithms.

CRediT authorship contribution statement

Xing Xu: Methodology, Writing – original draft. **Jiazhu Zeng:** Writing – original draft. **Yun Zhao:** Conceptualization, Methodology, Supervision. **Xiaoshu Lü:** Conceptualization, Investigation, Data curation.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Xing Xu reports financial support was provided by Ministry of Science and Technology of the People's Republic of China.

Acknowledgments

This work was supported by National Key Research and Development Program of China (2019YFE0126100).

Data availability

The data that has been used is confidential.

References

Brand, M., Masuda, M., Wehner, N., & Yu, X. H. (2010). Ant Colony Optimization algorithm for robot path planning. In *Computer design and applications (ICCDA), 2010 international conference on*.

Chen L., Yang H., Chen Z., Feng Z. Research on intelligent disinfection-vehicle system design and its global path planning Electronics, 12 (7) (2023)

Chong L., Jian L., XueQuan L. Static rectangle expansion A* algorithm for pathfinding

IEEE Transactions on Games, 14 (1) (2020), pp. 23-35

Dijkstra E.W. A note on two problems in connexion with graphs Numerische Mathematik, 1 (1959), pp. 269-271

Harabor D., Botea A. Breaking path symmetries on 4-connected grid maps Proceedings of the AAAI conference on artificial intelligence and interactive digital entertainment, vol. 6 (2010), pp. 33-38

Hart P.E., Nilsson N.J., Raphael B. A formal basis for the heuristic determination of minimum cost paths IEEE Transactions on Systems Science and Cybernetics, 4 (2) (1968), pp. 100-107

Hu Y., Yang S.X. A knowledge based genetic algorithm for path planning of a mobile robot

IEEE international conference on robotics and automation, 2004. proceedings. ICRA'04. 2004, vol. 5, IEEE (2004), pp. 4350-4355

Karaman S., Frazzoli E. Sampling-based algorithms for optimal motion planning

International Journal of Robotics Research, 30 (7) (2011), pp. 846-894

Kumar N., Vámossy Z., Szabó-Resch Z.M. Heuristic approaches in robot navigation

2016 IEEE 20th jubilee international conference on intelligent engineering systems, IEEE (2016), pp. 219-222

Lai R., Wu Z., Liu X., Zeng N. Fusion algorithm of the improved a* algorithm and segmented bezier curves for the path planning of mobile robots Sustainability, 15 (3) (2023)

LaValle S.M. Rapidly-exploring random trees: A new tool for path planning Research Report (1998)

Li C., Huang X., Ding J., Song K., Lu S. Global path planning based on a bidirectional alternating search A* algorithm for mobile robots Computers & Industrial Engineering, 168 (2022), Article 108123

Li Z., Li Y., Rong X., Zhang H. Grid map construction and terrain prediction for quadruped robot based on c-terrain path IEEE Access, 8 (2020), pp. 56572-56580

Li Y., Zhao J., Chen Z., Xiong G., Liu S. A robot path planning method based on improved genetic algorithm and improved dynamic window approach Sustainability, 15 (5) (2023)

Liu L., Wang X., Yang X., Liu H., Li J., Wang P. Path planning techniques for mobile robots: Review and prospect Expert Systems with Applications, 227 (2023), Article 120254

Lyons J.B., aldin Hamdan I., Vo T.Q. Explanations and trust: What happens to trust when a robot partner does something unexpected?Computers in Human Behavior, 138 (2023), Article 107473

Muhammad A., Ali M.A., Turaev S., Shanono I.H., Hujainah F., Zubir M.N.M., Faiz M.K., F aizal E.R.M., Abdulghafor R. Novel algorithm for mobile robot path planning in constrained environment Comput. Mater. Contin, 71 (2021), pp. 2697-2719

Nakamura T., Kobayashi M., Motoi N. Local path planning with turnabouts for mobile robot by deep deterministic policy gradient 2023 IEEE international conference on mechatronics, (ICM) (2023), pp. 1-6

Nakamura T., Kobayashi M., Motoi N. Path planning for mobile robot considering turnabouts on narrow road by deep Q-network IEEE Access, 11 (2023), pp. 19111-19121

Ou Y., Fan Y., Zhang X., Lin Y., Yang W. Improved A* path planning method based on the grid map Sensors, 22 (16) (2022), p. 6198

Qin H., Shao S., Wang T., Yu X., Jiang Y., Cao Z. Review of autonomous path planning algorithms for mobile robots Drones, 7 (3) (2023)

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., & Ng, A. (2009). ROS: An open-source Robot Operating System. In *ICRA workshop on open source software, vol. 3* (pp. 1–6).

Singh R.K., Nagla K.S. Enhanced A* algorithm for the time efficient navigation of unmanned vehicle by reducing the uncertainty in path length optimization MAPAN, 38 (2) (2023), pp. 317-335

Warren C.W. Global path planning using artificial potential fields 1989 IEEE international conference on robotics and automation, IEEE Computer Society (1989), pp. 316-317

Xu H., Yu G., Wang Y., Zhao X., Chen Y., Liu J. Path planning of mecanum wheel chassis based on improved A* algorithm Electronics, 12 (8) (2023)

You Z., Shen K., Huang T., Liu Y., Zhang X. Application of A* algorithm based on extended neighborhood priority search in multi-scenario maps Electronics, 12 (4) (2023)

Zhang Y., Gong D.-w., Zhang J.-h. Robot path planning in uncertain environment using multi-objective particle swarm optimization Neurocomputing, 103 (2013), pp. 172-185

Zhang A., Li C., Bi W. Rectangle expansion A* pathfinding for grid maps Chinese Journal of Aeronautics, 29 (5) (2016), pp. 1385-1396

Zhonghe Y., Zhaohui L., Smith M.R. Mechanisms and machine theory

Mechanisms and machine theory (2004)