

Research Article

Open Access



A phase search-enhanced Bi-RRT path planning algorithm for mobile robots

Yuhao Sun¹, Huazhong Zhu¹, Zhaocheng Liang¹, Andong Liu¹, Hongjie Ni¹, Ye Wang²

¹Department of Information Engineering, Zhejiang University of Technology, Hangzhou 310032, Zhejiang, China.

²Faculty of Engineering, Lishui University, Lishui 323000, Zhejiang, China.

Correspondence to: Prof. Hongjie Ni, College of Information Engineering, Zhejiang University of Technology, No. 288, Liuhe Road, Xihu District, Hangzhou 310032, Zhejiang, China. E-mail: zdfynhj@zjut.edu.cn

How to cite this article: Sun, Y.; Zhu, H.; Liang, Z.; Liu, A.; Ni, H.; Wang, Y. A phase search-enhanced Bi-RRT path planning algorithm for mobile robots. *Intell. Robot.* **2025**, *5*(2), 404-18. <http://dx.doi.org/10.20517/ir.2025.20>

Received: 30 Nov 2024 **First Decision:** 13 Mar 2025 **Revised:** 2 Apr 2025 **Accepted:** 21 Apr 2025 **Published:** 9 May 2025

Academic Editor: Simon Yang **Copy Editor:** Pei-Yun Wang **Production Editor:** Pei-Yun Wang

Abstract

The proposed improvement to the Rapidly-exploring Random Tree (RRT) path planning algorithm is aimed at addressing the issue of slow convergence speed caused by boundary information in the original algorithm, by introducing a phase search approach. The initial approach involves employing a three-stage search strategy to generate sampling points that are specifically oriented toward real-time sampling failure rate, thereby significantly reducing the number of redundant nodes. Simultaneously, a balanced exploration strategy is introduced, enhancing the algorithm's convergence speed by constructing two randomly growing trees for searching. Secondly, a path-pruning strategy is implemented, effectively reducing the path length. Finally, the bidirectional exploration technique from the improved algorithm is applied to the traditional RRT algorithm based on boundary information, and comparative experiments are conducted. The experimental results demonstrate that, compared to the traditional boundary-based RRT method, the proposed improved algorithm reduces the running time by 13.4% and decreases the path length by 9.51%.

Keywords: Phase search, mobile robots, RRT, balanced exploration strategy

1. INTRODUCTION

With the rapid advancement of robotics, mobile robots are increasingly employed in military applications, industrial, medical, logistics, service^[1-4] and agriculture fields^[5]. Mobile robots have a wide range of applications thanks to the development of autonomous navigation technology. Path planning and trajectory tracking



© The Author(s) 2025. **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License (<https://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, sharing, adaptation, distribution and reproduction in any medium or format, for any purpose, even commercially, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.



are at the core of autonomous navigation for mobile robots^[6,7]: the former designs obstacle avoidance paths based on criteria such as time and distance, and the latter ensures that the robot accurately tracks the path and responds to dynamic disturbances. In the field of path planning, there are several main approaches, including state-space search algorithms represented by the A*-algorithm^[8], intelligent bionic algorithms represented by particle swarm optimization (PSO)^[9] and intelligent bug swarm algorithms (IBA)^[10], stochastic sampling methods centered around Rapidly-exploring Random Tree (RRT)^[11], and machine learning methods based on reinforcement learning^[12]. Among these methods, the A* algorithm has gained popularity because it enables robots to search for optimal results accurately. Nevertheless, it is computationally demanding, exhibits poor real-time capabilities, and requires extensive computation time, and the algorithm's search efficiency deteriorates as the number of traversed nodes increases^[13]. To reduce the computation time of the A* algorithm, a time-efficient A* improvement algorithm based on time efficiency is proposed by Guruji *et al.*^[14]. This method optimizes the speed of path planning by accepting slightly longer paths; however, these paths have more turning points, making it difficult for robots to meet their kinematic requirements during actual travel. RRT is a path-planning method that employs a sampling-based search strategy. The algorithm begins at the starting point, samples random points, and identifies the closest obstacle-free point to each sample within the random tree. This point is then added to the tree. This process is repeated until the vicinity of the endpoint is thoroughly explored^[15].

The random tree expansion algorithm has been enhanced to boost its capabilities, and researchers have proposed various improved versions. For example, the RRT-Goal Bias^[16,17] algorithm directs random samples to the endpoint with a certain probability, thus increasing the likelihood that a path to the goal will be found by the algorithm. Extend RRT^[18] algorithm further introduces the concept of a collection of path points, which accelerates the convergence of the algorithm and enhances the stability of the generated paths. Additionally, Palmieri *et al.* introduced an RRT-based motion planning algorithm that addresses arbitrary angle path bias^[19]. This method utilizes an arbitrary angle search technique to find the near-optimal path in a discrete search manner. So that, the performance of this method exhibits large differences under different environmental conditions. This algorithm grows two fast-expanding randomized trees simultaneously from the starting point and the endpoint, connecting these two trees through a greedy heuristic method, thereby significantly improving the speed of path search. However, the Bidirectional (Bi)-RRT algorithm has several significant drawbacks: the random sampling of the algorithm results in too many redundant nodes, leading to a low search speed, and the convergence speed needs to be enhanced.

Many researchers have invested significant effort into improving and optimizing the limitations of the Bi-RRT algorithm. Ma *et al.* introduced the probabilistic smoothing bidirectional RRT (PSBi-RRT) algorithm, which enhances the traditional Bi-RRT by incorporating a goal bias strategy, a node correction mechanism, and a θ -cut mechanism^[20]. These innovations effectively address issues such as the poor quality of initial solutions and slow convergence rates. Similarly, Fan *et al.* developed a bootstrap bidirectional Informed-RRT* (BI-RRT*) algorithm, which incorporates extended range exploration, bidirectional search, and trajectory refinement to significantly improve path planning capabilities^[21]. Despite these advancements, the computational efficiency of these methods still requires further optimization to meet the demands of real-time online applications.

In this paper, an improved phase search-based Bi-RRT path planning algorithm for mobile robots has been proposed. Initially, a three-phase search strategy is implemented, whereby sampling points are generated under the guidance of the real-time sampling failure rate, and by balancing the exploration strategy, two randomly growing trees are concurrently established for searching, enhancing the algorithm's convergence speed. Secondly, by introducing a path pruning strategy, the overall path length is efficiently shortened and the quantity of superfluous nodes is substantially decreased. Finally, the superiority of the proposed algorithm is verified through simulation and real comparative experiments on Matlab and Robot Operating System (ROS).

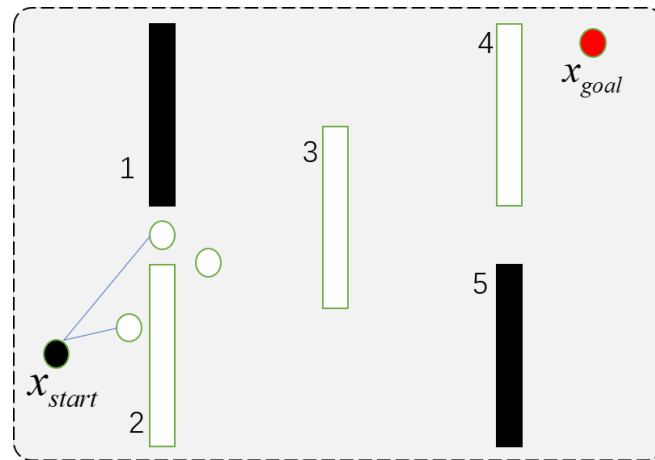


Figure 1. Diagram for detecting the valuable obstacles.

The rest of the paper is organized as follows: Section 2 systematically describes the principles of the fast search random tree algorithm for path planning based on boundary information; Section 3 proposes a multidimensional optimization strategy for the BI-RRT framework, and each subsection discusses specific improvements such as probabilistic bias adaptation and dynamic heuristic constraints; Section 4 validates the algorithm through simulation and physical experiments in a complex obstacle environment effectiveness and compares it with the conventional RRT, its bidirectional variant (Bi-RRT), and the original BI-RRT; finally, the last section summarizes the main findings and outlines future research directions for autonomous navigation systems.

2. BOUNDARY-INFORMED RRT

The current RRT algorithm has disadvantages such as more search nodes and longer planning paths; Wang *et al.* restricted the sampling domain of the RRT algorithm to the environs of value obstacles, thus reducing the number of nodes generated by sampling, improving the efficiency of the algorithm in path planning, and optimizing the path length at the same time^[22].

As shown in Figure 1, since the straight line between the starting point and the target is the optimal path, it is easy to see that it is easier to obtain the optimal path by sampling in the vicinity of the obstacles crossed by this line. In the RRT algorithm, the focus is more on avoiding the barriers, resulting in the planned path needing to be optimized more. At the same time, due to the sampling process's randomness, the path search takes a lot of time.

In the value obstacle method, as shown in Figure 1, obstacles 2, 3, and 4 (marked in green) are regarded as value obstacles because they are crossed by a straight line from the start to the end.

Within the number of iterations, the value obstacle detection function *DetectValObs* generates any point around the obstacle, i.e., the value obstacle, that the start-finish line crosses, denoted as x_{mid} ;

DetectValObs Detect the obstacles located between the starting point x_{start} and the ending point x_{goal} in the first loop; the generated x_{mid} is the point located at the edge of the obstacle closest to x_{start} ; The algorithm will generate multiple points as alternative points for x_{mid} , the number of which depends on the shape of the obstacle, in case of a triangular obstacle, the algorithm will generate three points as alternative points for x_{mid} , while in case of a quadrilateral, it will generate four alternative points; One point is randomly selected as x_{mid} among the generated alternative points. Based on boundary information, the above RRT algorithm can plan

an approximate optimal path offline based on the relative positions of obstacles. Nevertheless, the algorithm's rate of convergence still requires enhancement.

3. PHASE SEARCH BI-RRT

This section introduces a novel approach that integrates and enhances the Bi-RRT algorithm. By harnessing the benefits of boundary information guidance and bidirectional search, this method aims to further optimize the efficiency of path planning and decrease computational expenses.

3.1. Fundamentals

The basic principle of the Bi-RRT algorithm is to construct a randomized tree in space with the target point and the initial point as the root node simultaneously, respectively, and generate connected paths by expanding the two randomized trees alternately in relative directions to save path search time. The pseudo-code of Bi-RRT algorithm is shown in Algorithm 1 as follows.

Algorithm 1 Algorithm of Improved RRT Based on Boundary Information

```

1:  $T_1 \leftarrow \text{Tree1}(x_{\text{start}}); T_2 \leftarrow \text{Tree2}(x_{\text{goal}})$ 
2: for  $i = 1$  to  $n$  do
3:    $x_{\text{rand}} \leftarrow \text{Sample}(M);$ 
4:    $x_{\text{near1}} \leftarrow \text{Near}(x_{\text{rand}}, T_1);$ 
5:    $x_{\text{new1}} \leftarrow \text{Steer}(x_{\text{rand}}, x_{\text{near1}}, \text{stepSize});$ 
6:    $E_1 \leftarrow \text{Edge}(x_{\text{new1}}, x_{\text{near1}});$ 
7:   if  $\text{CollisionFree}(x_{\text{new1}}, x_{\text{near1}})$  then
8:      $T_1.\text{addNode}(x_{\text{new1}});$ 
9:      $T_1.\text{addEdge}(E_1);$ 
10:     $x_{\text{near2}} \leftarrow \text{Near}(x_{\text{new1}}, T_2);$ 
11:     $x_{\text{new2}} \leftarrow \text{Steer}(x_{\text{new1}}, x_{\text{near2}}, \text{stepSize});$ 
12:     $E_2 \leftarrow \text{Edge}(x_{\text{new2}}, x_{\text{near2}});$ 
13:    if  $\text{CollisionFree}(x_{\text{new2}}, x_{\text{near2}})$  then
14:       $T_2.\text{addNode}(x_{\text{new2}});$ 
15:       $T_2.\text{addEdge}(E_2);$ 
16:    do
17:      if  $\text{CollisionFree}(x_{\text{new2}}^*, x_{\text{near2}})$  then
18:         $T_2.\text{addNode}(x_{\text{new2}}^*);$ 
19:         $T_2.\text{addEdge}(E_2)^*;$ 
20:      else
21:        BREAK;
22:      end if
23:    while not  $x_{\text{new1}} = x_{\text{new2}}$ 
24:  end if
25:  if  $|T_1| < |T_2|$  then
26:     $\text{Swap}(T_1, T_2);$ 
27:  end if
28: end if
29: end for

```

The basic steps of the algorithm are as follows:

- (1) In the known state space M , initialize the random tree T_1, T_2 , and set the start point and target point as x_{start} and x_{goal} , respectively. Set the root nodes of the random trees T_1 and T_2 to x_{start} and x_{goal} , respectively. Define and assign the relevant parameters, such as the number of iterations i , the step size is $StepSize$, and so on;
- (2) Use the random function to generate a random point in global space x_{rand} , then denote the point on the random tree T_1 with the smallest distance from x_{rand} as x_{near1} , then start from x_{near1} and extend a point along the direction pointing to x_{rand} in steps of $StepSize$, denoted as x_{new1} , and indicate the path connecting x_{near1} and x_{new1} as E_1 ;
- (3) Detect whether there is an obstacle between x_{near1} and x_{new1} through the collision detection function $CollisionFree()$, if it passes the detection, add x_{new1} to the set of nodes of the random tree T_1 , and add E_1 to the set of edges of the tree T_1 ;
- (4) Similar to the above process, generate x_{new2} and E_2 and add them to the node set and edge set of T_2 ;
- (5) When the spacing between two nodes on the random tree T_1, T_2 is less than the set value and passes the collision detection, it indicates that the path search is successful and the expansion of the random tree will be stopped;
- (6) Perform path backtracking to find and connect its parent node forward from the goal point, repeat this process until it connects to the start point, and generate a feasible path connecting the goal and start points.

3.2. Balancing exploration strategies with three stage search strategies

Aiming at the problem that the Bi-RRT algorithm has a long sampling time and produces too many redundant nodes, this paper introduces a balanced exploration strategy that effectively improves the efficiency of path search by generating the search tree from two directions simultaneously.

This paper uses a three-stage search strategy based on the real-time sampling failure rate. The novel search methodology adjusts the extension of x_{rand} and x_{near} within the Bi-RRT framework in response to environmental data, enabling the algorithm to accommodate environments of varying complexity. The choice of the search strategy is given in

$$Sample = \begin{cases} ASS, & \text{if } p \leq p_1; \\ DRAS, & \text{if } p_1 < p < p_2; \\ TBRRT, & \text{if } p \geq p_2. \end{cases} \quad (1)$$

where p_1 and p_2 are the search failure rate thresholds (SFRTs), $p_1, p_2 \in (0, 1)$, and p is the real-time sampling failure rate. SFRT is obtained by dividing the number of failed samples by the total number of samples. The search strategy for each sample is determined by p . When $p \leq p_1$, the adaptive sector search is selected for the sampling points; When $p_1 < p < p_2$, the dynamic right angle search is run for the sampling points. Otherwise, switch to the target biased RRT search strategy.

The creation of the reference points is illustrated in Figure 2. The light gray point labeled n_{ref} represents the reference point, which is generated by expanding outward near each vertex of the obstacle. The dark gray point n_{ref} in Figure 3A indicates the returned reference point, which is located around the square centered at $n_{nearest}$ and n_{goal} . Here, O marks the center of the square, d signifies the distance from $n_{nearest}$ to O , and a denotes the side length of the square area. The fan-shaped sampling area is depicted in Figure 3B. The line segment connecting the vertices of the sector at $n_{nearest}$ and the endpoint at n_{goal} determines the central angle of the sector, denoted as α . Both the central angle α and the radius r are functions of the real-time sampling failure rate p . As the failure rate increases, the central angle widens and the radius diminishes. α and r are

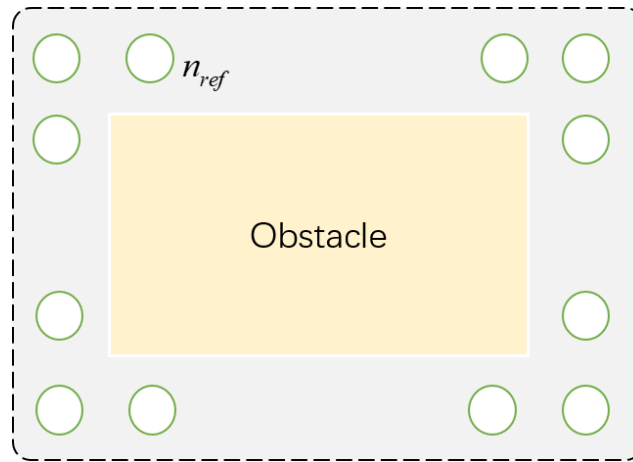


Figure 2. Reference point extension diagram.

respectively calculated by

$$\alpha = \pi(p + p_1)^{1/2} \quad (2)$$

$$r = kL(1 - p)^{1/2} \quad (3)$$

where α is the center angle, r is the radius, k is the scale factor, and L is the robot's extended step size.

The Dynamic Right Angle Search (DRAS) strategy and the Adaptive Sector Search (ASS) strategy exhibit identical algorithmic flows. The distinction lies in the DRAS policy, which substitutes the sector sampling region with a dynamic right-angle sampling region. This right-angle sampling region propels itself toward n_{goal} , with $n_{nearest}$ serving as the vertex of the right angle. An acute angle is defined as the angle formed between any right-angled edge and the line segment connecting $n_{nearest}$ and n_{goal} . When n_{goal} is situated in the upper right quadrant relative to $n_{nearest}$, the configuration of the right-angle sampling region is illustrated in Figure 3C. The right-angle region comprises two congruent rectangles, mutually perpendicular and superimposed, with the longer side b_1 and the shorter side b_2 of these rectangles calculated by

$$b_1 = k_1 L \ln(e - p) \quad (4)$$

$$b_2 = k_2 L \ln(e - p) \quad (5)$$

where k_1 , k_2 are the gain coefficients, L is the AGV expansion step, e is the Euler number, and p is the expansion failure rate. With the increase of p , the sample area in Figure 3C gradually shrinks from solid to dashed areas.

3.3. Path pruning strategy

The RRT and Bi-RRT algorithms generate an excessive number of redundant nodes in narrow channel environments, leading to the algorithms' high path cost. In this paper, we optimize the node redundancy problem in the RRT algorithm from two aspects. First, referring to the idea of the RRT* algorithm, the paths can be pruned by comparing the costs of the paths to remove the unnecessary high-cost parts to obtain a better path. Secondly, a triangular pruning strategy will be applied after receiving the feasible paths.

Upon identifying a feasible path, the algorithm employs a triangular pruning strategy, which hinges on the principle that the sum of the lengths of any two sides of a triangle exceeds the length of the third side. The path pruning process is depicted in Figure 4. When the non-collision point n_2 in the original path is pruned, a new path is yielded, namely, $n_1 - n_3 - n_4$.

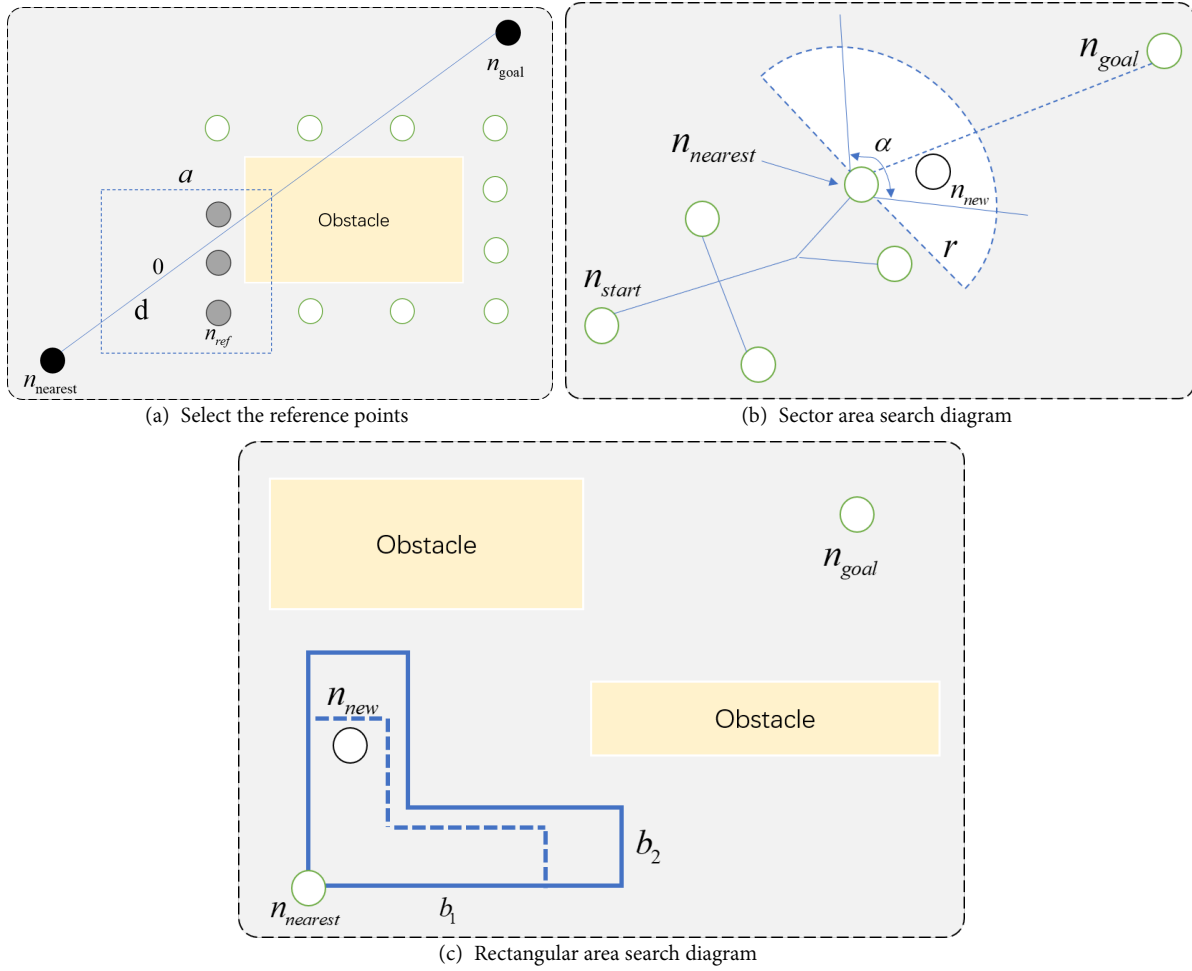


Figure 3. Diagram of reference point selection and search area.

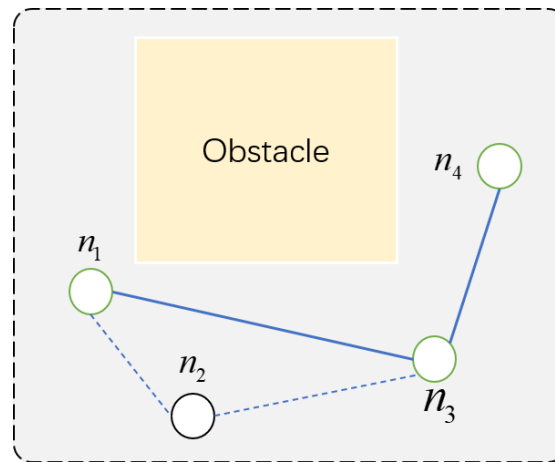


Figure 4. Triangle pruning diagram.

3.4. Improvement of Bi-RRT

The RRT algorithm is based on boundary information, and the improved Bi-RRT algorithm is fused to ensure the overall operational efficiency and stability of the algorithm. First, a two-dimensional grid map is created by Light Detection and Ranging (Lidar), and the value obstacles in the map are labeled using the RRT algo-

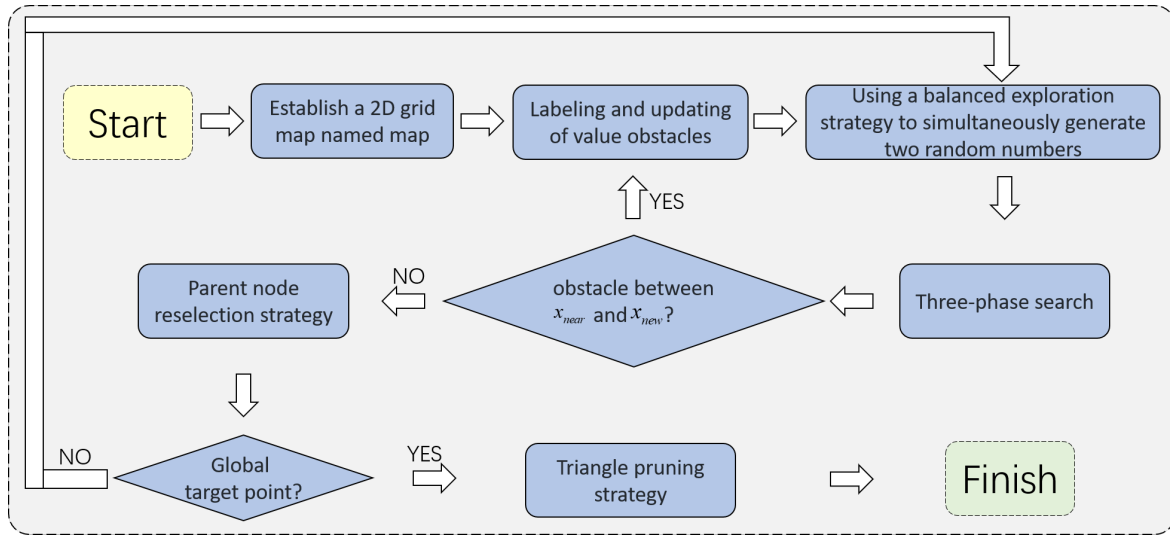


Figure 5. Algorithm flowchart for boundary-information RRT with NCB-RRT. RRT: Rapidly-exploring random tree; NCB-RRT: nonholonomic constraints-based rapidly-exploring random tree.

rithm based on boundary information. Second, the fusion of the improved Bi-RRT algorithm improves the convergence speed of the algorithm by reducing the generation of redundant nodes through the balanced exploration strategy and the three-stage search strategy. Further, the path pruning strategy is used to optimize the generated paths, effectively improving the overall efficiency of path planning. The algorithmic flow of the convergence algorithm is shown in Figure 5. The specific steps are summarized as follows:

(1) Add the initial point x_{start} to $\tau.init$ as the growing root node of the tree, set the value of the iteration number i to 0, and set the exploration point x_{temp} of the value obstacle detection function $DetectValObs$ to the starting point x_{start} . Define two Boolean variables $collisionFree1$ and $collisionFree2$ to record the return value of the collision detection function $Checkcollision(x_{near}, x_{new})$ (Algorithm 2, lines 1-4).

(2) The maximum number of iterations the user can set max iteration. If the algorithm fails to find a feasible path within the specified maximum number of iterations, it will return to failure (Algorithm 2, line 27). Within the number of iterations, the value obstacle detection function $DetectValObs$ generates any point around the obstacle (value obstacle) that the start/end line crosses, denoted as x_{mid} . Proximity node function $NearNeighbor$ will be generated from the tree τ to search for a distance of x_{mid} the nearest node as x_{near} . At this point, it grows along x_{mid} to x_{near} in a specific step to generate a new node x_{new} . The collision detection function $Checkcollision$ detects whether there is any obstacle between x_{near} and the newly generated node x_{new} and stores the result as a boolean value in the variable $collisionFree1$. (Algorithm 2, section 5, paragraph 5) (Algorithm 2, lines 5-9).

(3) The three-stage search strategy is run. First, the algorithm finds the closest point to the endpoint in the search tree denoted as $n_{nearest}$. At line 10, Algorithm 2 returns the set of reference points between $n_{nearest}$ and n_{goal} . Then, the reference point in the set that does not collide with $n_{nearest}$ is labeled n_{new} (line 12), and n_{new} is added to the search tree. If the result returned by $collisionFree1$ is true, x_{new} is added to the tree nodes. The new branch generated between x_{near} and x_{new} is added to the tree (Algorithm 2, lines 14-16); If x_{new} falls within the end range x_{goal} , the search can be terminated and the path planning task is completed. If the line between the newly generated node and the endpoint does not cross any obstacle, then x_{new} can be assigned to x_{temp} to update the value obstacle (Algorithm 2, lines 18-22).

Algorithm 2 Pseudocode of Improved Bi-RRT algorithm

```

1:  $i = 0, x_{temp} = x_{start}$ 
2:  $\tau.Init(x_{start})$ 
3: CollisionFree1 = false
4: CollisionFree2 = false
5: while  $i < maxIteration$  do
6:    $x_{mid} = DetectValObs(x_{temp}, x_{goal});$ 
7:    $x_{near} = NearNeighbor(x_{mid}, r);$ 
8:    $x_{new} = Steer(x_{mid}, x_{near});$ 
9:    $n_{nearest} \leftarrow n_{nearest}; p \leftarrow p;$ 
10:   $a, r \leftarrow Angle(n_{nearest}, n_{goal}, p);$ 
11:   $S'_{sec} \leftarrow S_{sec}(a, r);$ 
12:   $n_{new} \leftarrow Sample(S'_{sec});$ 
13:   $collisionFree1 = Checkcollision(x_{near}, x_{new});$ 
14:  if collisionFree1 == true then
15:     $\tau.AddNode(x_{new});$ 
16:     $\tau.AddEdge(x_{near}, x_{new});$ 
17:  end if
18:  if  $(x_{new} \in X_{goal})$  then
19:    return  $\tau;$ 
20:     $collisionFree2 = Checkcollision(x_{new}, x_{goal});$ 
21:  else if (collisionFree2 == true) then
22:     $x_{temp} = x_{new};$ 
23:  else
24:     $i++;$ 
25:  end if
26: end while
27: return failure

```

(4) Repeat the above process until a collision-free path is found from the start to the target point.

4. SIMULATION RESULTS AND ANALYSIS

4.1. Algorithm simulation and analysis

To assess the efficacy of the proposed algorithm, it was compared against the RRT algorithm, the Bi-RRT algorithm, and the boundary information-based RRT algorithm. The superior performance of the proposed algorithm was evaluated in terms of path length, the number of nodes generated, and the computational time required.

The search component of the proposed algorithm in this paper consists of three distinct stages. The SFRT serves as the critical indicator for delineating these stages. To optimize the algorithm's performance across varying environments, it is essential to select appropriate SFRT values accordingly. Therefore, we systematically traverse the SFRT parameter space from 0 to 1 with a step size of 0.1. A cost function is specifically designed to select the optimal threshold configuration by identifying the group with the lowest average cost in each environment, as given in

$$Cost = 0.3node + 0.3len + 0.2iter + 0.2t \quad (6)$$

where *Cost* denotes the total cost, *node* represents the average number of nodes, *node* indicates the average path length, *len* corresponds to the average iteration count, and *iter* signifies the average runtime.

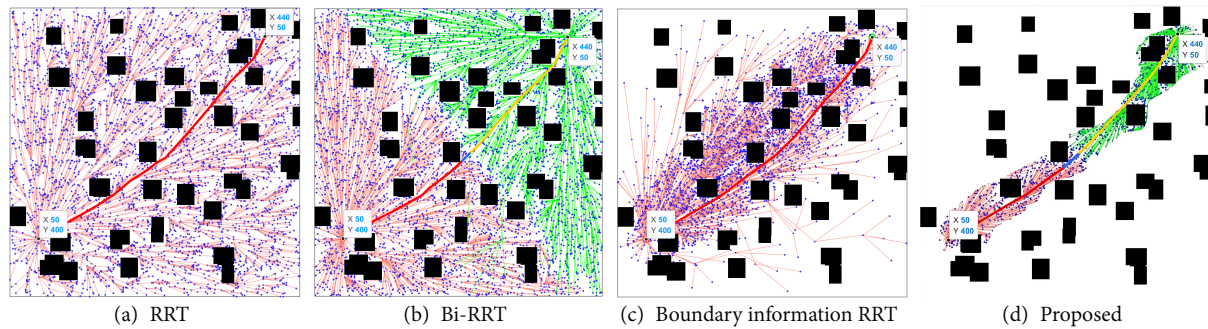


Figure 6. Simulation in the crowded environment. RRT: Rapidly-exploring random tree; Bi-RRT: bidirectional-rapidly-exploring random tree.

The hardware environment used for simulation is a computer with Intel Core(TM) i5-1135G7 CPU @3.5GHz, 16GB RAM with Windows 10 system, and the software environment is Matlab R2018b. The simulation experimental environment adopts the 600×600 -pixel ratio region of complex block-type obstacles as the two-dimensional simulation environment, where the black pixels denote the obstacle region and the white pixels represent the free region without barriers. Pixels denote the obstacle region, while white pixels denote the free region without barriers. The initial parameters of the algorithm are as follows: initial position $x_s = (50, 400)$, target position $x_t = (440, 50)$, step size $step = 40$, distance threshold $disTh = 40$, i.e., nodes within the distance threshold are considered as the same point.

Figure 6 shows the simulation comparison of the algorithm in the complex obstacle environment. Figure 6A presents the simulation of the RRT algorithm, which causes the algorithm to explore blindly due to the random selection of sampling points, resulting in the generation of more redundant nodes. Figure 6B shows a feasible path planned by Bi-RRT, which makes the search rate much higher and generates a better path by growing the tree in both directions. However, the algorithm still generates many redundant nodes. Figure 6C demonstrates the RRT algorithm based on boundary information from the figure. The number of nodes of the algorithm can be significantly reduced, and the planning efficiency is further improved. Figure 6D depicts the simulation of the proposed algorithm at the SFRT = 0.2. Thanks to the path-pruning strategy, the node utilization of the algorithm is further improved, and the search performance is significantly enhanced.

Due to the three-stage search strategy, the proposed algorithm can significantly improve the convergence speed of the algorithm by evaluating the real-time sampling failure rate p associated with the threshold SFRT as a measure of the complexity of the current environment. From the data analysis in Figure 7, it can be seen that in terms of path length, it is improved by about 37.32% compared to the RRT algorithm, and about 0.31% and 0.29% compared to the Bi-RRT and boundary information-based RRT algorithms, respectively; in terms of number of nodes, it is improved by about 50.06% and 50%. In terms of number of nodes, it is improved by about 50.06% and 50.20% over the RRT and Bi-RRT algorithms, respectively, and by about 50.06% and 50.20% over the boundary information-based RRT algorithm. In terms of running time, it is improved by about 19.23%, 10.50% and 23.32% over the RRT, Bi-RRT and boundary information-based RRT algorithms, respectively. In summary, when compared with RRT algorithm, Bi-RRT algorithm and boundary information-based RRT algorithm, the proposed algorithm shows significant superiority in terms of path length, number of nodes and running time, which proves its robustness and practicality in complex environments.

4.2. Real environment experiment

To deeply verify the effect of the algorithms proposed in this paper in practice, the authors decided to use a highly realistic and operable software experimental platform, ROS. ROS is an open-source software platform and a general platform widely accepted by most robot developers^[23]. The experiments in this paper use a

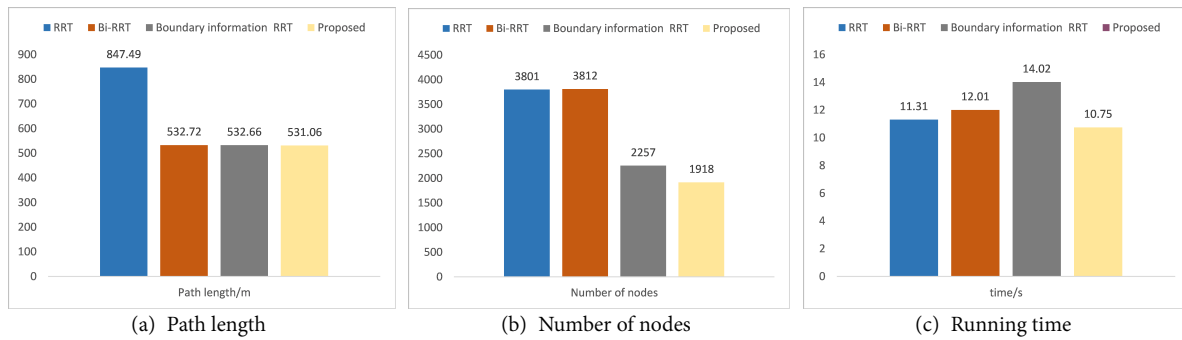


Figure 7. Performance comparison.

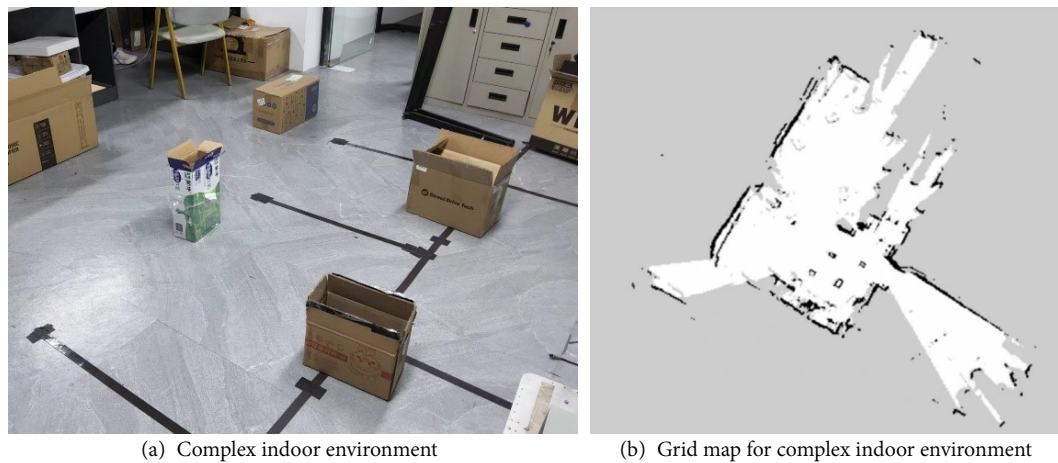


Figure 8. Experimental environment.

customized mobile robot with a differentially driven motor on each side to control its forward motion. The front of the robot is also equipped with an auxiliary wheel whose main function is to provide balancing support to ensure that the robot remains stable and non-slip during its movement.

The homemade mobile robot based on the above was experimented in a complex indoor environment to verify the effectiveness and robustness of the algorithm. The complex indoor environment is shown in Figure 8A. The dimensions of the environment in the laboratory are 10.8×5.8 meters. The floor smoothness of the experimental site is consistent, while on the raster map, the black circle represents the robot's position, and the block graphic indicates static obstacles. First, a 2D map of the indoor environment is created based on the Hector SLAM algorithm, and the resulting raster map is imported into RViz; as shown in Figure 8B, the resolution of the raster map is set to $20 \text{ cm} \times 20 \text{ cm}$. Next, an adaptive Monte Carlo localization algorithm is applied to accurately determine the position and direction of the mobile robot, which tracks the robot's movement by particle filtering method. With iterations, all the particles will converge gradually so that the robot's current position can be accurately determined.

In this paper, a comparison experiment between the proposed algorithm and the RRT algorithm based on boundary information is carried out in a complex indoor environment as shown in Figure 9. The proposed algorithm is used in the experimental scenario. The proposed algorithm introduces a balanced exploration strategy, which effectively improves search efficiency by generating the search tree from two directions simultaneously. In addition, the introduced three-stage search strategy also enhances the quality of sampling points. As for the improved RRT algorithm based on boundary information, since the selection of sampling points near the obstacles is randomized, in real experimental scenarios, the robot will turn in place and move

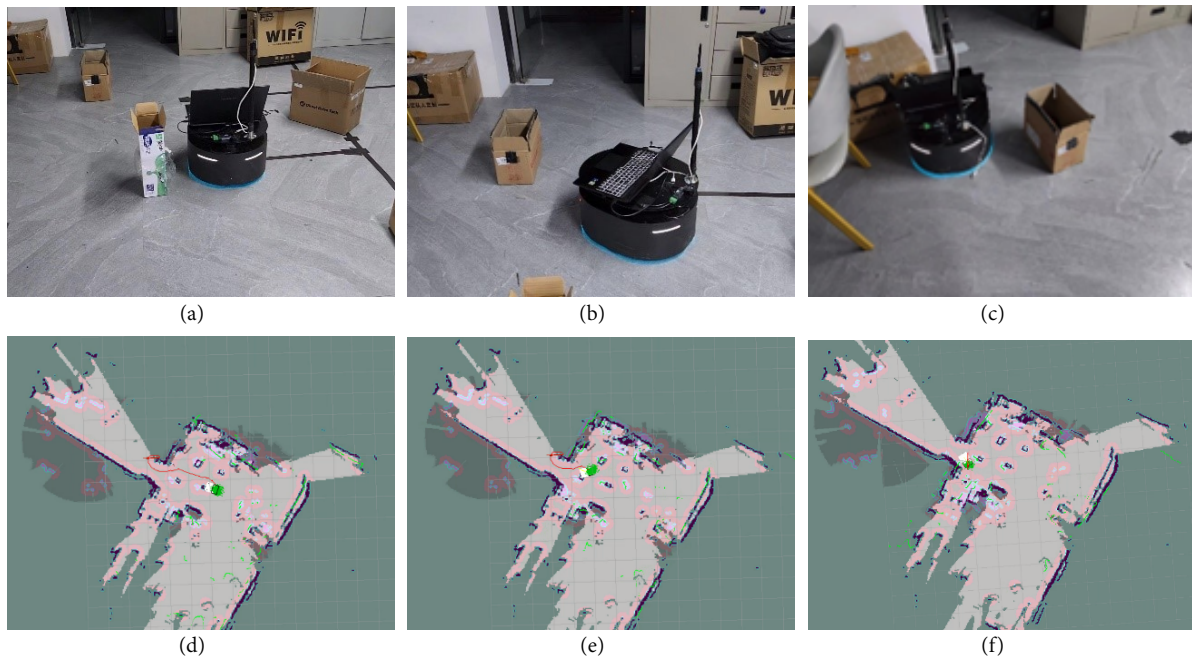


Figure 9. Results of the RRT algorithm using boundary data. (A) Schematic showing the robot spinning around an obstacle; (B) Schematic of the robot spinning during operation; (C) Schematic of the robot spinning near the target point; (D) Rviz interface showing the schematic of the robot spinning near an obstacle; (E) Rviz interface showing schematic of robot spinning near a target point; (F) Rviz interface showing schematic of robot spinning near a target point. RRT: Rapidly-exploring random tree.

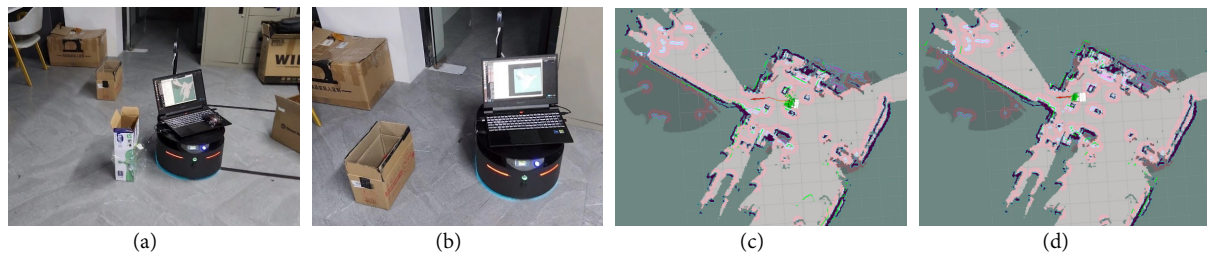


Figure 10. Experimental results of the improved RRT algorithm. (A) Diagram of the robot's movement process; (B) Diagram of the robot reaching the target point; (C) Rviz interface during the movement process; (D) Rviz interface when the robot reaches the target point. RRT: Rapidly-exploring random tree.

backward around the obstacles (as shown in [Figure 9](#)).

In contrast, the algorithm proposed in this paper introduces a balanced exploration strategy, which effectively improves the search efficiency by simultaneously generating the search tree from two directions. In addition, the introduced three-stage search strategy also improves the quality of the sampling points, which significantly reduces the in-situ steering and backward phenomenon of the boundary RRT algorithm. According to the actual measurement, the deviation of the final position of the robot from the expected target point is less than 0.1 meter, which basically meets the requirements of actual robot navigation [[Figure 10](#)].

Combined with the experimental results in [Figure 11](#), the proposed algorithm in this paper improves the running time by 13.4% and the planning path length by 9.51% compared with the RRT algorithm based on boundary information. The proposed algorithm shows stronger robustness in complex indoor environments.

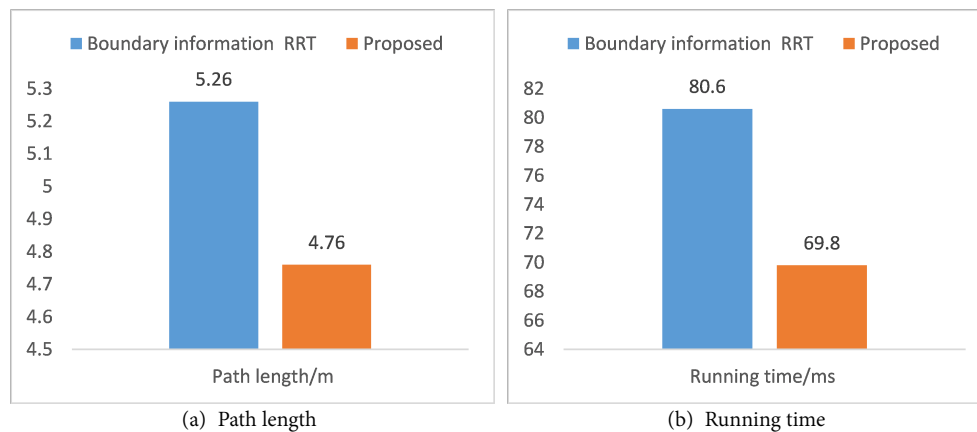


Figure 11. Experimental results comparison.

5. FUTURE WORK

The algorithm proposed in this study demonstrated good performance in static environments, but there are still some limitations and areas for improvement. Firstly, we will further explore the applicability of the algorithm in dynamic environments, such as handling the situation of moving obstacles, to verify its robustness and real-time performance. Additionally, in response to the issue of computational overhead, we will strive to optimize the efficiency of the algorithm, reduce resource consumption, and study its applicability on different robot platforms (such as unmanned aerial vehicles and wheeled robots). For the application scenarios of unmanned aerial vehicles, we need to extend the algorithm from 2D space to 3D space to support a wider range of practical needs. Finally, we plan to combine reinforcement learning with existing algorithms to achieve adaptive adjustment and dynamic optimization, thereby further enhancing the performance and practicality of the algorithm.

6. CONCLUSIONS

This paper presents an enhanced Bi-RRT algorithm based on boundary information. Initially, a balanced exploration strategy is employed to search for new nodes by constructing two alternating random trees. Concurrently, a three-stage search strategy is implemented to enhance the quality of sampling points. Additionally, to address the issue of winding generated paths in the Bi-RRT algorithm, a path pruning strategy is introduced, where paths are pruned by comparing their costs, thereby optimizing the generated paths. Furthermore, the convergence speed of the algorithm is further improved by integrating the enhanced Bi-RRT with the boundary information-based RRT algorithm. Finally, simulations and experiments demonstrate that the algorithm not only outperforms mainstream path planning algorithms such as RRT and Bi-RRT but also that the fusion algorithm surpasses the boundary information-based RRT algorithm in terms of algorithmic time and path length. Future work will focus on enhancing the algorithm's adaptability across various scenarios, real-time performance, and path safety.

DECLARATIONS

Authors' contributions

Made substantial contributions to the research, idea generation, algorithm design, simulation, wrote and edited the original draft: Sun, Y.; Zhu, H.; Liang, Z.

Performed data acquisition and provided administrative, technical, and material support: Ni, H.; Liu, A.; Wang, Y.

Availability of data and materials

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Financial support and sponsorship

This work has been funded by the Key Research and Development Program of Zhejiang Province (Project No. 2023C01224).

Conflicts of interest

Liu, A. is the Junior Editorial Board Member of the journal *Intelligence & Robotics*. Liu, A. was not involved in any steps of editorial processing, notably including reviewer selection, manuscript handling, or decision-making. The other authors declare that there are no conflicts of interest.

Ethical approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Copyright

© The Author(s) 2025.

REFERENCES

1. Liu, S.; Wang, X.; Wu, Y.; Li, Q.; Yan, J.; Levin, E. Path planning method for USVs based on improved DWA and COLREGs. *Intell. Robot.* **2024**, *4*, 385-405. [DOI](#)
2. Jin, Z.; Liu, A.; Zhang, W. A.; Yu, L.; Yang, C. Learning an autonomous dynamic system to transfer periodic human motion skills. *IEEE Tran. Neural Netw. Learn. Syst.* **2025**, *36*, 7757-63. [DOI](#)
3. Qin, D.; Jin, Z.; Liu, A.; Zhang, W. A.; Yu, L. Asynchronous event-triggered distributed predictive control for multi-agent systems with parameterized synchronization constraints. *IEEE Trans. Autom. Control*, **2024**, *69*, 403-9. [DOI](#)
4. Jin, Z.; Si, W.; Liu, A.; Zhang, W. A.; Yu, L.; Yang, C. Learning a flexible neural energy function with a unique minimum for globally stable and accurate demonstration learning. *IEEE Tran. Robot.* **2023**, *39*, 4520-38. [DOI](#)
5. Hassan, M. U.; Ullah, M.; Iqbal, J. Towards autonomy in agriculture: design and prototyping of a robotic vehicle with seed selector. In *2016 2nd International Conference on Robotics and Artificial Intelligence (ICRAI)*, Rawalpindi, Pakistan. Nov 01-02, 2016. IEEE; 2016. pp. 37-44. [DOI](#)
6. Balding, S.; Gning, A.; Cheng, Y.; Iqbal, J. Information rich voxel grid for use in heterogeneous multi-agent robotics. *Appl. Sci.* **2023**, *13*, 5065. [DOI](#)
7. Saleem, O.; Hamza, A.; Iqbal, J. A fuzzy-immune-regulated single-neuron proportional–integral–derivative control system for robust trajectory tracking in a lawn-mowing robot. *Computers.* **2024**, *13*, 301. [DOI](#)
8. Huang, H.; Li, Y.; Bai, Q. An improved A star algorithm for wheeled robots path planning with jump points search and pruning method. *Complex Eng. Syst.* **2022**, *2*, 11. [DOI](#)
9. Tawhid, M. A.; Ibrahim, A. M. An efficient hybrid swarm intelligence optimization algorithm for solving nonlinear systems and clustering problems. *Soft Comput.* **2023**, *27*, 8867-95. [DOI](#)
10. Zohaib, M.; Pasha, S. M.; Javaid, N.; Iqbal, J. IBA: intelligent bug algorithm – A novel strategy to navigate mobile robots autonomously. In: Shaikh, F.; Chowdhry, B.; Zeadally, S.; Hussain, D.; Memon, A.; Uqaili, M. editors. *Communication technologies, information security and sustainable development. IMTIC. 2013. Communications in Computer and Information Science*. Springer, Cham; 2014. pp. 291-9. [DOI](#)
11. LaValle, S. M.; Kuffner, J. J. Randomized kinodynamic planning. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, Detroit, USA. May 10-15, 1999. IEEE; 1999. pp. 473-9. [DOI](#)
12. Hua, M.; Zhou, W.; Cheng, H.; Chen, Z. Improved DDGP algorithm-based path planning for unmanned surface vehicles. *Intell. Robot.* **2024**, *4*, 363-84. [DOI](#)
13. Ha, I. K. Improved A-star search algorithm for probabilistic air pollution detection using UAVs. *Sensors* **2024**, *24*, 1141. [DOI](#)
14. Guruj, A. K.; Agarwal, H.; Parsediya, D. K. Time-efficient A* algorithm for robot path planning. *Proc. Technol.* **2016**, *23*, 144-9. [DOI](#)
15. Zhang, R.; Guo, H.; Andriukaitis, D.; Li, Y.; Królczuk, G.; Li, Z. Intelligent path planning by an improved RRT algorithm with dual grid map. *Alex. Eng. J.* **2024**, *88*, 91-104. [DOI](#)
16. Xia, Z.; Chen, G.; Xiong, J.; Zhao, Q.; Chen, K. A random sampling-based approach to goal-directed footstep planning for humanoid

- robots. In *2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Singapore. Jul 14-17, 2009. IEEE; 2009. pp. 168-73. [DOI](#)
17. Urmson, C.; Simmons, R. Approaches for heuristically biasing RRT growth. In *Proceedings 2003 IEEE International Conference on Intelligent Robots and Systems*, Las Vegas, USA. Oct 27-31, 2003. IEEE; 2003. pp. 1178-83. [DOI](#)
18. Li, D.; Li, Q.; Cheng, N.; Song, J. Extended RRT-based path planning for flying robots in complex 3D environments with narrow passages. In *2012 IEEE International Conference on Automation Science and Engineering (CASE)*, Seoul, Korea. Aug 20-24, 2012. IEEE; 2012. pp. 1173-8. [DOI](#)
19. Palmieri, L.; Koenig, S.; Arras, K. O. RRT-based nonholonomic motion planning using any-angle path biasing. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden. May 16-21, 2016. IEEE; 2016. pp. 2775-81. [DOI](#)
20. Ma, G.; Duan, Y.; Li, M.; Xie, Z.; Zhu, J. A probability smoothing Bi-RRT path planning algorithm for indoor robot. *Future Gener. Comput. Syst.* **2023**, *143*, 349-60. [DOI](#)
21. Fan, H.; Huang, J.; Huang, X.; Zhu, H.; Su, H. BI-RRT*: an improved path planning algorithm for secure and trustworthy mobile robots systems. *Heliyon* **2024**, *10*, e26403. [DOI](#)
22. Wang, J.; Li, X.; Meng, M. Q. H. An improved RRT algorithm incorporating obstacle boundary information. In *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Qingdao, China. Dec 03-07, 2016. IEEE; 2016. pp. 625-30. [DOI](#)
23. Kohút, M.; Čornák, M.; Dobiš, M.; Babinec, A. Teaching robotics with the usage of robot operating system ROS. In: Balogh, R.; Obdržálek, D.; Christoforou, E. editors. *Robotics in Education. RiE 2023. Lecture Notes in Networks and Systems*. Springer, Cham; 2023. pp. 299-313. [DOI](#)