ITC 2/53
Information Technology and Control
Vol. 53 / No. 2 / 2024
pp. 442-452
DOI 10.5755/j01.itc.53.2.36240

Research on Autonomous Mobile Robot Path Planning Based on M-RRT Algorithm

Received 2024/02/01 | Accepted after revision 2024/03/27

HOW TO CITE: Tang, Z., Ma, H., Xue, B. (2024). Research on Autonomous Mobile Robot Path Planning Based on M-RRT Algorithm. *Information Technology and Control*, 53(2), 442-452. https://doi.org/10.5755/j01.itc.53.2.36240

Research on Autonomous Mobile Robot Path Planning Based on M-RRT Algorithm

Zhuozhen Tang
Hohai University, Electrical and Power Engineering College, Nanjing, China
Jiangsu Maritime Institute, Nanjing,China

Hongzhong Ma
Hohai University, Electrical and Power Engineering College, Nanjing, China

Bin Xue
State Grid Jiangsu Electric Power Co., Ltd, Nanjing, China

Corresponding author: Zhuozhen Tang, e-mail: tina160@hhu.edu.cn

The Rapidly-Exploring Random Tree (RRT) algorithm has demonstrated proficiency in adapting to path search challenges within high-dimensional dynamic environments. However, a notable limitation of the RRT algorithm lies in its inability to fulfill the criteria for achieving the shortest and smoothest path for mobile sensing nodes. To address the limitations of the conventional RRT algorithm and enhance the path planning for mobile robots, this paper proposed an innovative approach named M-RRT, designed to overcome the aforementioned shortcomings and optimize the path planning process for mobile sensing nodes. First, the search area is constructed according to the defined coverage density. After searching the path in the search area, the RRT algorithm uses the greedy method to delete the intermediate nodes in the path, and obtains the uniquly optimal path. Finally, the Bezier curve is used to optimize the path, which makes the path shortest and meets the dynamic requirements of the mobile node. Simulation results show that M-RRT has better path and faster convergence speed than traditional RRT, which can better meet the planning requirements of mobile nodes.

KEYWORDS: MRRT, search area, coverage density, mobile node.

1. Introduction

In the rapidly evolving field of wireless sensor networks, the path planning problem of mobile nodes stands out as a fundamental challenge. The path planning problem of mobile node is to search an optimal, safe (collision-free) and complete path from the starting point to the end point in the robot configuration space according to a certain optimization criterion. Researchers have proposed that deploying a small number of redundant mobile nodes in wireless sensor networks can solve the coverage and node failure problems [12-13]. This scheme not only improves the network performance but also controls the network overhead. However, the process of moving requires nodes to navigate obstacles efficiently, follow the shortest possible route, and maintain a continuous smooth path, aligning with the dynamics and energy efficiency needs of mobile nodes [6].

The strategic deployment of redundant mobile nodes in wireless sensor networks represents a pivotal enhancement, addressing coverage and node failure issues effectively. This approach not only boosts network performance but also manages network overhead, fostering a resilient and efficient system. Nevertheless, the principal challenge lies in the motion of these nodes. They must skillfully negotiate obstacles, minimize travel distance, and ensure a smooth, uninterrupted path, a trio of requirements crucial for their operational utility.

Researchers have extensively explored various algorithms to tackle these complex requirements. Classical pathfinding algorithms like Dijkstra and Floyd have been foundational, offering reliable solutions that necessitate a thorough understanding of the entire environment and necessitate subsequent path smoothing [7]. On the other hand, the A* and D* algorithms represent a leap forward, capable of rapidly deriving optimal paths [10]. Their efficiency, however, is contingent on factors like grid size and the choice of heuristic methods, which can be limiting in more dynamic or constrained environments.

The advent of evolutionary intelligence algorithms – including genetic algorithms, ant colony optimization, simulated annealing, and particle swarm optimization – marked a significant milestone in global path optimization[9]. These methods excel in accuracy and have found widespread application. Nevertheless, these methods require prior global environmental information, and their time and space complexity are high, limiting their adaptability to edge devices like mobile robots. Moreover, in most instances, wireless sensor networks can only provide limited environmental data, making the application of such algorithms challenging due to the extensive preprocessing needed, especially in dynamic unknown environments [3].

It is found that planning algorithms based on random sampling can adapt to more complex environments with lower time and space complexity and can effectively adapt to the path planning problem of mobile sensor nodes [16]. such as the Rapidly-exploring Random Tree (RRT) and its derivatives, including RRT-CONNECT, RRT*, RRT*-FN, Informed RRT* and other algorithms [1-2,7], which can well adapt to the motion dynamics requirements of mobile nodes However, their reliance on random node generation introduces a significant drawback: the paths generated are often longer than optimal, lacking in efficiency and practicality for real-world applications. In order to obtain smooth, shortest and obstacle-avoiding paths, researchers combined Bezier, Benstine and other curves with heuristic and artificial intelligence methods, and proposed many excellent results [15, 18]. For example, the artificial potential field method based on Bezier curve adopted the quadratic Bezier method and artificial potential field method to avoid obstacles. Better optimization results can be obtained in sparse environment. However, for dense wireless sensor networks, performance degrades significantly. Therefore, in dense infinite sensor networks, a reasonable path planning algorithm is needed to obtain the path.

In recent developments, methods like the Dynamic Window Approach combined with deep reinforcement learning have shown improvements in dynamic obstacle avoidance, yet they face limitations in movement options and environmental adaptability [8]. Bi-layer hybrid algorithms involving ACO, PSO, and A* have been explored for multi-task path planning, but their complexity and interaction requirements limit their efficiency in dynamic environments [14].

The Modified Adaptive Ant Colony Optimization (MAACO) algorithm offers faster convergence and reduced path lengths but struggles in dynamic scenarios requiring real-time adaptability [17]. The Expanding Path RRT (EP-RRT) has improved efficiency but faces challenges in diverse environments and dynamic changes [4]. The Probability Smoothing Bi-RRT (PSBi-RRT) algorithm enhances convergence speed and collision reduction but does not guarantee initial solution quality [11]. A continuous RRT-based method using B-spline curves for non-holonomic mobile robots showed promise but needed additional collision avoidance mechanisms [5].

In response to these challenges, this paper introduces the Modified Rapidly-exploring Random Tree (M-RRT) search strategy, an innovative approach designed for efficient path planning in dense, dynamic wireless sensor networks. M-RRT represents a novel solution that adeptly combines the strengths of various existing algorithms while addressing their primary limitations. This strategy is especially suited for real-time applications where rapid adaptation to changing environments is paramount. The main contributions are:

1 This paper introduces the M-RRT search strategy to achieve coverage and connectivity in wireless sensor networks. The strategy accounts for the distribution of initial access points, obstacles, and static nodes, employing a density set to define the search area.

2 Enhanced path optimization is proposed, utilizing a greedy algorithm to eliminate intermediate nodes for a more efficient path. The optimization aims to achieve a path that meets mobile nodes' needs, including shortest distance, obstacle avoidance, and smooth trajectory.

Path smoothing with quadratic Bezier curves is utilized to refine the solution path obtained through the M-RRT strategy. This step aims to minimize angles between adjacent segments, ensuring the final path is not only optimal but also smooth for mobile nodes.

2. RRT Algorithm

The RRT algorithm initiates with the root node set at the initial point and proceeds to construct a random expansion tree by iteratively adding leaf nodes through random sampling. The expansion process encompasses four essential steps: (1) the random generation of a node; (2) identification of the nearest point in the tree to the randomly generated node and connecting them; (3) generation of a node along the line based on a predetermined value; and (4) insertion of the node into the tree if it satisfies certain criteria. Illustrated in Figure 1, the process commences with the starting point, represented as q_{start}, and a randomly selected point q_{rand}. Subsequently, the nearest node q_{near} is determined, and a point on the line between q_{near} and q_{rand} is chosen, with the condition that the distance, D(q_{near} , q_{new}), satisfies the prescribed step size d_{step}. If this condition is met, the newly generated node q_{new} is inserted into the tree; otherwise, the process is repeated until a suitable node is found. This iterative procedure continues until the destination point q_{end} is reached. Ultimately, the constructed random tree provides a path from the initial point to the destination point, as depicted in Figure 1 and Figure 2.

Figure 1
Schematic diagram of RRT algorithm random tree growth

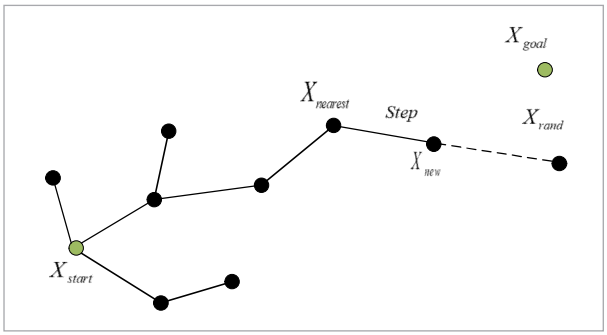
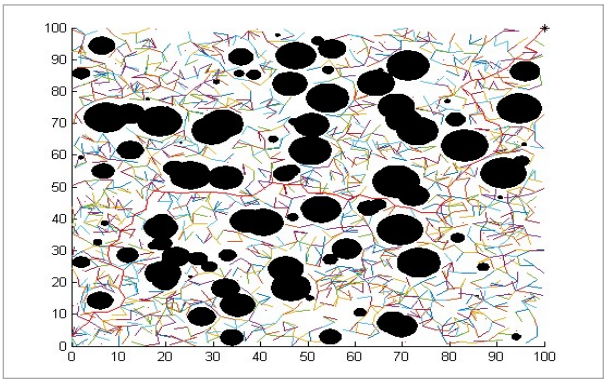


Figure 2
RRT planning path



3. M-RRT Algorithm Modeling

After considering the assumptions, M-RRT is elaborated as follows: (1) All mobile nodes, except for themselves, are considered as obstacles; (2) The static sensing node has a radius of R_{static} and the mobile node has a radius of R_{move}; (3) Let R_{obsi} represent the projected radius of the i^{th} obstacle in the forward direction; (4) By increasing the size of all obstacles by $R_{move,}$ the algorithm simplifies the search process by not needing to account for its own size.

3.1. Initial Region Setting Based On Coverage Density

The performance of the algorithm is directly influenced by the configuration of the initial region, which is conducted in two steps.

1. Establishing the baseline for the initial region: In a wireless sensor network, the baseline line refers to the line connecting the starting point and ending point. The search area extends on both sides of this line, with a length denoted as q_{start}, q_{end} and L_d.
2. Determining the size of the initial area, as depicted in Figures 3 and Figure 4, referred to as d_{free}.

Based on the connectivity within a wireless sensor network, the number and distribution of nodes within a specific area can be deduced. Despite potential local inconsistencies, a generally uniform distribution is observable. To derive a more accurate initial value, this paper introduces the concept of obstacle coverage density.

Figure 3
Schematic diagram of the initial expanded area

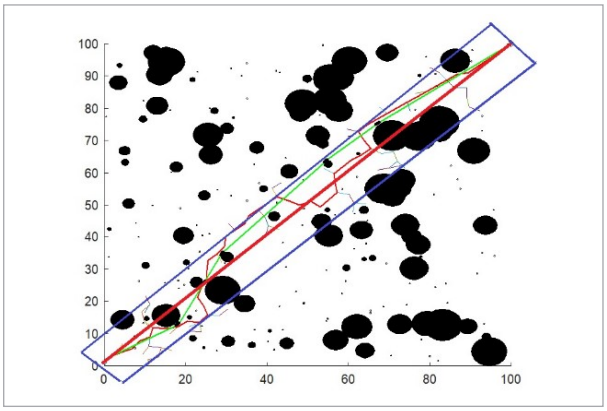
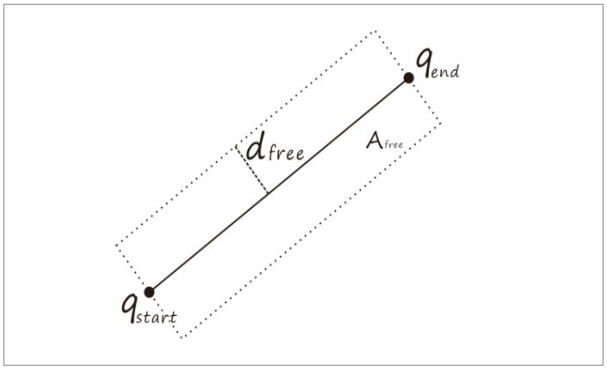


Figure 4
Initial area



Definition 1: Obstacle Coverage Density. This metric, defined within wireless sensor networks, quantifies the ratio of the combined projected area of all obstacles and static nodes to the total area covered by the wireless sensor network, as expressed in Formula (1). In this formula, A_{Obs} represents the transmission area occupied by obstacles, A_{Static} denotes the individual area of a static node, A_s signifies the overall area covered by the wireless sensor network, and N_{static} indicates the quantity of static nodes.

$$\rho = \frac{A_{Obs} + N_{static} * A_{Static}}{A_s} \tag{1}$$

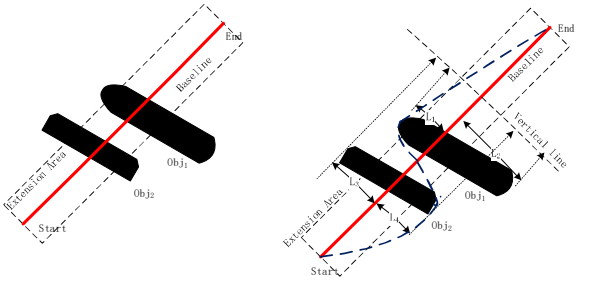
The calculation of the initial coverage area width, considering the coverage and connectivity characteristics of nodes, allows for the determination of the width of the initial coverage area (d_{free_init}) for a given number of static nodes (M), as shown in Equation (2).The symbol [ ] denotes the operation of taking the integer part.

$$d_{free\_init} = \left[\frac{M * \rho * A_{Static}}{L_d}\right] \tag{2}$$

The initial area calculated by Equation (2) might include significant obstacles within the initial area, rendering the RRT algorithm unable to find a path from the initial region, as depicted in Figure 5.

However, at this time, it is found that the size of the initial area can be modified according to the obstacle projection, and the size to be adjusted can be calculated according to the obstacle projection, as shown in

Figure 5
The Obstacle initial region



(1) Obstacle traversing the initial region (2) Obstacle forward direction projection

Equations (3)-(4). Formula (3) is to calculate the minimum distance from each obstacle to the baseline projection, where is the projection length of the obstacle on the right side of the baseline, $D(Obs_{ri}, Line_{vert})$ $D(Obs_{ri}, Line_{vert})$is the projection length of the obstacle on the left side of the baseline, and Obs_iMin Obs_iMin is to find their minimum value. Formula (4) means to find the maximum value after the obstacle projection is solved, so that the value Obs_{cross} to be adjusted can be obtained.

$$Obs_i = \text{Min}\{D(Obs_{ri}, Line_{vert}), D(Obs_{li}, Line_{vert})\} \quad (3)$$

$$Obs_{cross} = \max_i\{Obs_i\} \quad (4)$$

In accordance with the formulations presented in Equations (4) and (2), the derivation of Equation (5) is achievable through the process of unification, wherein the equation is consolidated to yield the essential initial region, denoted as $d_{free\_init}d_{free\_init}$.

$$d_{free\_init} = \begin{cases} \left\lceil \frac{M * \rho * Static_{area}}{L_d} \right\rceil & \text{others} \\ Obs_{cross}, & \text{obstacle cross extension area} \end{cases} \quad (5)$$

3.2. Extended Search Area Setting Based on Coverage Density

Due to the influence of the shape, size and position of obstacles, the RRT algorithm cannot find the path in the set area. In this case, the search area needs to be expanded, as shown in Equation (6).

$$d_{free} = d_{free} + d_{exp} \quad (6)$$

In Formula (6), the step size is expanded after each search failure. Repeat the preceding steps until the search path is found, as shown in Figures 6-7.

Figure 6
Region expansion

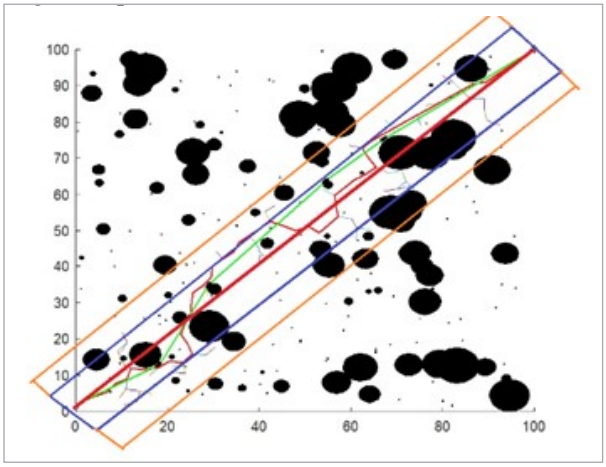
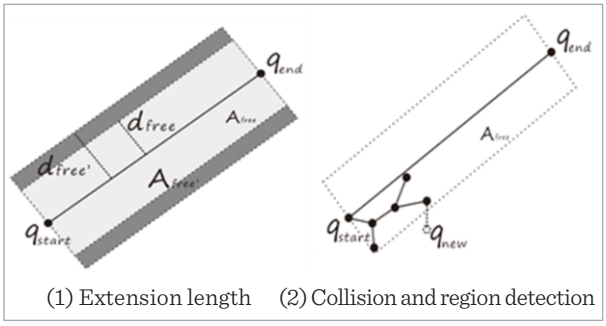


Figure 7
Extension length and collision detection



(1) Extension length (2) Collision and region detection

Normally, it can be set according to the initial value, but during the expansion process, the value depends on: (1) the coverage density, (2) whether the obstacle crosses the extended area; (3) d_{free_init}, the initial value of d_{free}, as shown in Formula (7), k represents the number of extensions, indicating that the more the number of extensions, the smaller the area to be expanded; d_{exp} represents the coefficient, indicating the size of the initial area in the default state.

$$d_{exp} = \frac{1}{k} * R_{exp} * \rho * d_{free\_init} \quad (7)$$

At the same time, if the obstacle is too large after taking into account the extended area, the Formula (6) is used to calculate the adjustment area, so that the width of the expanded search area can be obtained according to the Formula (7), and $d_{free}d_{free}$ can be obtained, as shown in the Formula (8).

$$d_{free} = \begin{cases} d_{free} & \text{others} \\ Obs_{\text{cross}}, & \text{obsatcle cross extension area} \end{cases} \quad (8)$$

It can be seen from the Formulas (7)-(8) that after gradual expansion, d_{free} is the sum of the initial value of the region multiplied by the natural reciprocal; from the limit summation formula. After infinite expansion, d_{free} will become infinite to cover the entire region, thus satisfying the probability completeness.

3.3. Random Expansion Tree

RRT adds the boundary detection function in the search process after increasing the search area limit, as shown in Figure 7(2). When q_{new} is not in the search area, nodes are generated again. Figure 8 shows the flow of M-RRT's random search tree algorithm (Extend_Tree).

Figure 8
Extended search tree of M-RRT

```
Extend_Tree(tree)
1: flag1←0
2: while flag1==0
3:    q_rand ←Random_Configration()
4:    q_near ←Nearest_Neighbor(tree,q_rand)
5:    q_new ←New_Configration(q_rand, q_near,d_step)
6:    if Collision(q_new, tree)==0&&Collision_Border(q_new,d_free)==0
7:          tree.add_node(q_new)
8:          flag1←1
9: if | q_new-q_end | <d_step
10:    return 1
```

3.4. Greedily Delete Node Optimization Path

The paths obtained through tree expansion consist of a series of nodes, which are not necessarily the shortest. Further optimization is required, and this paper employs a greedy algorithm to prune nodes from the expanded tree, resulting in a shorter path. The key steps of the proposed greedy algorithm are as follows:

1 Initialization of the original path P_K, where P_K represents the path between q_{start} and q_{end} obtained from a random tree T_K.

2 Initialization of the sequence of optimized path nodes S_k, with an initial empty value. The starting point from P_K is inserted into S_k as the first anchor point.

3 Iterative selection of anchor points and insertion into S_k. Nodes are continuously selected from P_K, and it is determined whether the line connecting the selected point and the anchor point intersects with obstacles. If no intersection occurs, the selection process continues until an intersection is detected. At this point, the node selected just before the intersection becomes the next anchor point, and it is inserted into Sk. The search continues until the endpoint becomes the last anchor point and is inserted into Sk. This process effectively removes many intermediate nodes. In Figure 9, the sequence of points S_k represents the anchor points selected from P_K. After this step, in wireless sensor networks, situations similar to Figure 10 often arise, where obstacles are contained between three anchor points. In such cases, the intermediate anchor points need to be removed, constituting step (4).

Figure 9
Anchor sequence after iterative selection

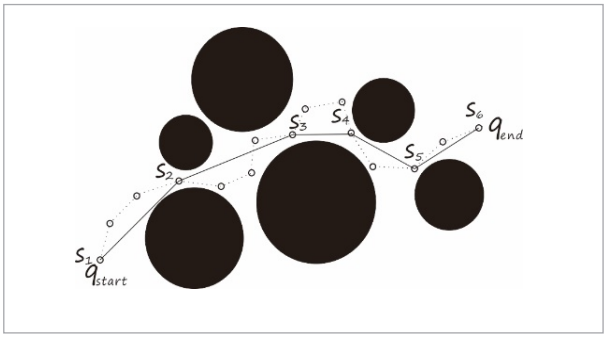
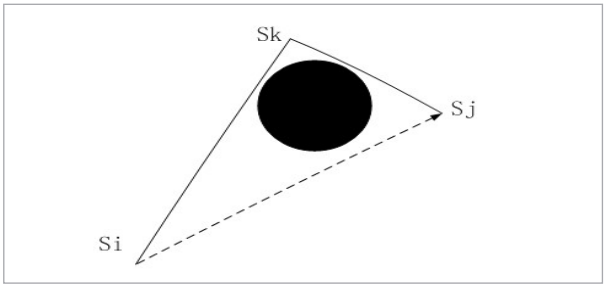


Figure 10
The obstacle is in the middle of the three anchors



4 Verification of the equality of lengths between P_K and S_k sequences. If they are not equal, P_K is cleared, S_k is copied to P_K, S_k is cleared, and step (3) is repeated. If they are equal, the optimization process is concluded.

3.5. Shortest Smooth Path

The path formed by the optimized node sequence S_k S_k cannot meet the demand of stable movement of nodes, mainly because the node sequence cannot meet the kinematic mechanical demand of mobile nodes at the connection point. For mobile nodes, their dynamic equation can be expressed by Formula (9).

$$\begin{cases} \dot{x}(t) = v\cos(t) \\ \dot{y}(t) = v\sin(t) \\ \dot{\theta}(t) = u(t), u(t) \in [-U_{max}, U_{max}] \end{cases} \quad (9)$$

where, v represents the moving speed of the node, $u(t)$ represents the angular speed, and U_{max} represents the maximum angular speed acceptable to the mobile node. In this way, the minimum radius of the mobile node when turning can be obtained, as shown in Formula (10).

$$R_{min} = \frac{v}{U_{max}} \quad (10)$$

The necessary condition for a mobile node to move smoothly is that the curvature at each point of the path is less than the value that the mobile node can bear. Therefore, the average probability condition [8] is adopted here, which satisfies Equation (11), where a and b represent the slope of any two points on the curve and the change point of P(a).

$$\|P'(a) - P'(b)\| \le \frac{|a-b|}{R_{min}} \quad (11)$$

According to the greedy optimization path, there must be obstacles between the two-line segments. In special cases involving a static sensor node, the radius length is: $R = Rmove + Rstatic$. If an arc is arbitrarily taken from a circle with radius and length R, the appropriate moving time can be found to enable smooth movement of the mobile sensing node. This arc can be replaced by a quadratic Bezier curve. Therefore, the process of searching for a smooth path involves search control points in S_k, ensuring that the path is both smooth and smooth, and the path is shortest.

Given n+1 points, the smooth path is solved by determining the junction points $a_1, a_2, \ldots, a_n, \ldots, a_{2n}$ on the two adjacent line segments, satisfying Formula (11) and collision avoidance. This makes the path shortest, that is, satisfying the solution of the Equation (12).

$$\begin{cases} \text{Min}(L); \\ L=\sum_{i=0}^{n} B_i+\sum_{i=0}^{n} l_i \\ B_i=\int_{t=0}^{t=1} a_{2i}(1-t)^2+2S_{i+1}(1-u)+a_{2i+1}t^2 \\ l_i=\text{length}(a_i,a_{i+1}) \\ B_{i\cap}\text{Obs}_j=\emptyset \\ \|P'(a_{2i})-P'(a_{2i+1})\|\le\frac{|a_{2i}-a_{2i+1}|}{R_{min}} \\ a_i(x)\le a_{i+1}(x) \end{cases} \quad (12)$$

In the Equation (12), B_i represents the length of the second Bezier curve. The integral method is employed for solving, where the a_{2i} subscript in B_i is an even number to start counting. The control points of the second quadratic curve, a_{2i} and a_{2i+1}, indicate that the horizontal coordinate of the previous control point is less than the horizontal coordinate of the next curve li represents the remaining line segment on S_iS_{i+1}, implying that, after smoothing, some line segments need to be traversed on S_iS_{i+1}. L represents the sum of the length of the conic and the remaining line segment. To solve for L is to find an optimal conic so that the path is the shortest.

During the solution process, if only the quadratic curve between each two adjacent line segments meets the conditions for the shortest path. (that is, the quadratic curve does not intersect with obstacles, the curvature meets the Formula (11)), the path between the two-line segments is the shortest. The sequence of control points may cross, that is, the horizontal coordinate of the i control point is greater than the horizontal coordinate of the i+1 control point, resulting in a conflict, and the required path is not obtained, as shown in Figure 11, $a_{i+2} > a_{i+3}$.

This situation, depicted in Figure 11 ($a_{i+2} > a_{i+3}$), implies that when the sensor node moves to a_{i+2}, it needs to fall back to a_{i+3}, which doesn't align with the requirements.

It is evident from Formula (12) that the process of solving smooth curves is an NP-difficult problem. However, it has been observed that minimizing the angle between two-line segments fulfills the requirements of smooth processing without control point conflicts. Figure 12 illustrates this scenario with $a_{i+2} = a_{i+3}$, and

the NP problem is transformed into a P problem. The proposed solution path is shorter and meets the collision avoidance and dynamics requirements of mobile sensor nodes. Figure 13 is the optimal path obtained by smooth optimization on the basis of Figure 8.

Figure 11
Cross conflict of control points

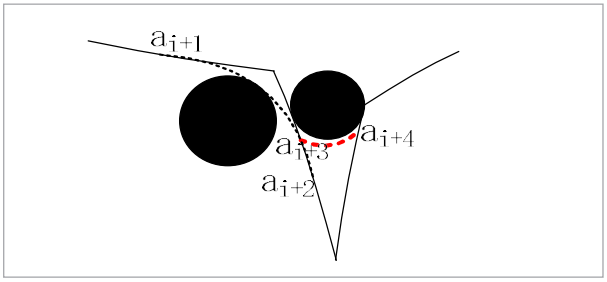


Figure 12
Minimum angle priority smoothing treatment

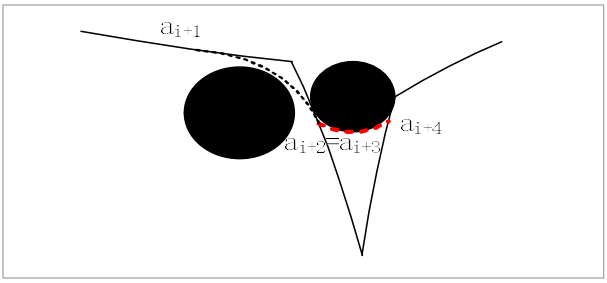
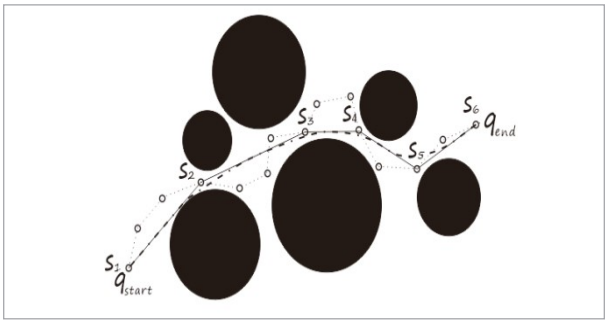


Figure 13
Smooth Optimization Path



3.6. Mobile Node Planning Strategy

RRT includes five key processing processes: region initialization, random search, region extension, path optimization, and smoothing, as shown in Figure 14.

The find extend Tree function calls Extend Tree(tree) and iterates step by step to determine whether the path can be searched for a finite number of times. If the path is found, Greedy Optimal(Tree) is called for optimization. Then call the Bezier Smooth(Path) function to smooth and get the required path.

Figure 14
M-RRT path planning strategy

```
1. Init d_free ;                          // region initialization (1)
2. While fpath = 0;
3.   fpath=find_Extend_Tree(d_free,N);     // Search in restricted area (2)
4    if(fpath==1 | d_free>S)   break;
5.       dfree=Extend(dfree);             // expand area (3)
6. End
7.   Path=Greedy_Optimal(Tree);           // Path optimization (4)
8.   Path=Bezier_Smooth(Path) ;           // Smooth optimization (5)
9.   return Path;
```

4. Experimental Results and Analysis

This study conducts simulations in a rectangular coordinate system [0, 100] × [0, 100], deploying 120 static sensing nodes and 5 redundant mobile nodes, where Nodes=125, k=10, S=100, M=1, featuring low-density circular obstacles. The radius of static nodes is denoted as r = 0.1, while the radius of mobile nodes is represented by R=0.2. For the R-RRT algorithm, the step size d_{step} is set to 3, d_{free} is 10, d_{exp} is 2, and N is 200. The outcomes of the M-RRT algorithm, as depicted in Figure 15, and the RRT algorithm, as illustrated in

Figure 15
Path planning of M-RRT

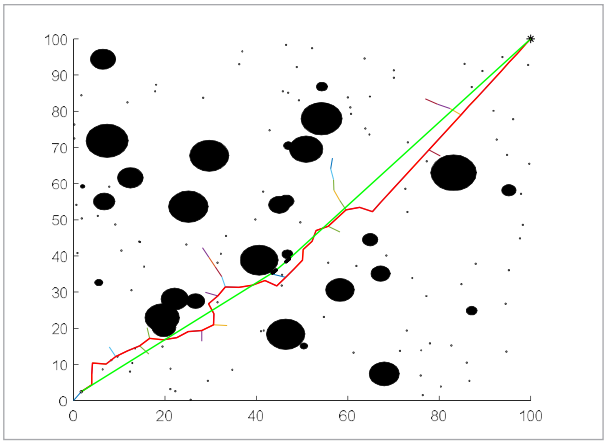
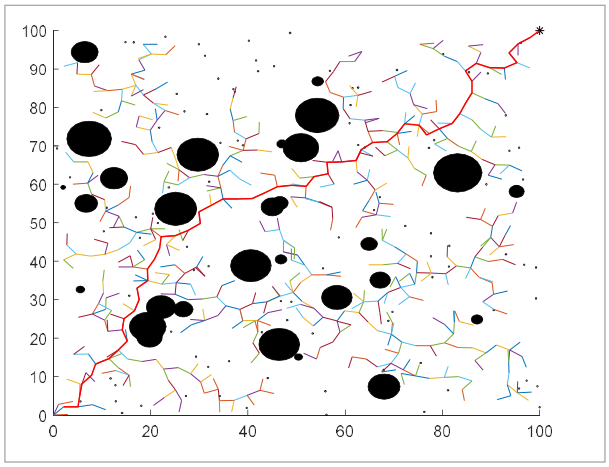


Figure 16, are compared. The results reveal that the enhanced M-RRT algorithm confines the search region to a smaller space, yielding paths significantly superior to those generated by the RRT algorithm.

Figure 16
Path planning of RRT



To further verify the improved strategy proposed in this paper, the number of obstacles was increased to 100, with other conditions unchanged. The results, shown in Figure 17 for M-RRT and Figure 18 for RRT, clearly demonstrate M-RRT's superior performance.

Figure 17
M-RRT planning after adding obstacles

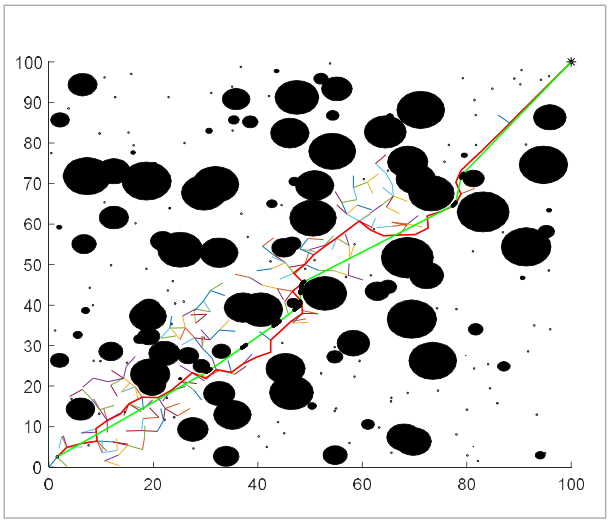
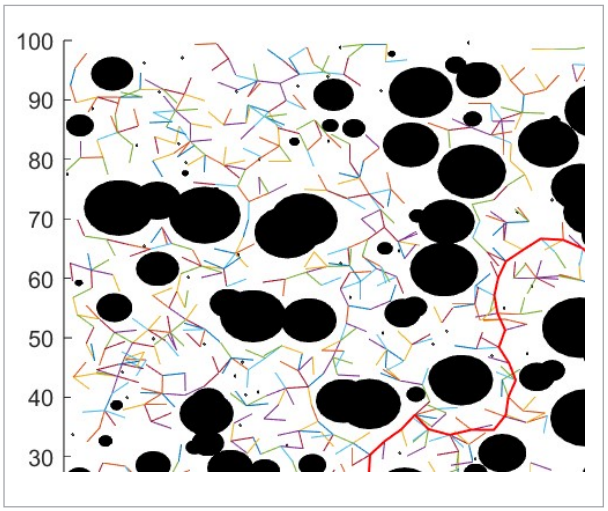


Figure 18
RRT algorithm after adding obstacles



In order to further verify the superiority of M-RRT algorithm, a case study involving autonomous vehicle navigation with dynamic obstacles (e.g., moving vehicles and pedestrians) was examined. five maps were randomly generated by changing the number of obstacles, and M-RRT and RRT algorithms were used to search the path. The experiments were repeated for 100 times, and their average convergence time and average path length were obtained, as shown in Figure 19 and 20, respectively. It can be seen from the results that the search time of M-RRT is much shorter than RRT, and the path length of M-RRT is also much shorter than RRT. The M-RRT algorithm facilitated efficient path planning, optimizing the route in real-time as the environmental conditions changed. This application highlighted the algorithm's potential in industries such as autonomous driving and drone navigation, where adaptability and real-time processing are critical.

Figure 19
Comparison of average search time

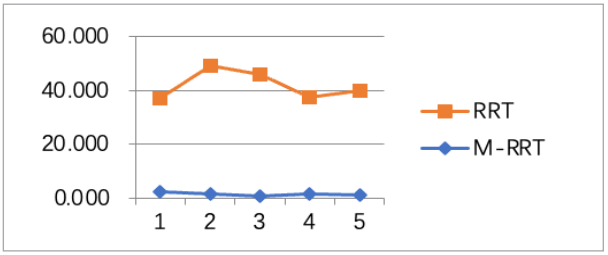
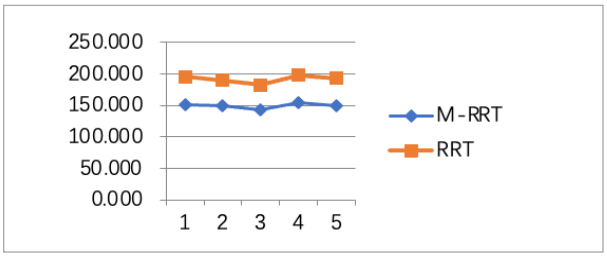


Figure 20
Comparison of average path length



5. Conclusion

By leveraging the coverage and connectivity of wireless sensor networks, this paper introduces the M-RRT strategy for efficient path planning of mobile nodes. From setting the initial search area based on coverage density to optimizing paths using a greedy algorithm and smoothing with quadratic Bezier curves, the M-RRT strategy not only reduces search time and path length but also provides a smooth and optimal path for mobile sensor nodes. Compared to the traditional RRT algorithm, the M-RRT algorithm demonstrates superior performance, especially in dense sensor networks, underscoring its advantages in terms of search efficiency and path quality.

Future work will focus on expanding the algorithm's applicability to a wider range of real-world scenarios, further refining its computational efficiency, and exploring its integration into larger-scale autonomous systems.

Acknowledgment

This work was sponsored in part by Science and Technology Project of State Grid Jiangsu Electric Power Co., Ltd (Ref. No.: 2023-1-32).

Competing Interests

The authors have no relevant financial or non-financial interests to disclose.

References

1. Aggarwal, S., Kumar, N. Path Planning Techniques for Unmanned Aerial Vehicles: A Review, Solutions, and Challenges. Computer Communications, 2020, 149, 270-299. https://doi.org/10.1016/j.comcom.2019.10.014
2. Cao, X., Zou, X., Jia, C., Chen, M., Zeng, Z. RRT-based Path Planning for an Intelligent Litchi-Picking Manipulator. Computers and Electronics in Agriculture, 2019, 156, 105-118. https://doi.org/10.1016/j.compag.2018.10.031
3. David Boon Moses, E., Anitha, G. Goal Directed Approach to Autonomous Motion Planning for Unmanned Vehicles. Defence Science Journal, 2016, 67(1), 45. https://doi.org/10.14429/dsj.67.10295
4. Ding, J., Zhou, Y., Huang, X., Song, K., Lu, S., Wang, L. An Improved RRT* Algorithm for Robot Path Planning based on Path Expansion Heuristic Sampling. Journal of Computational Science, 2023, 67, 101937-101937. https://doi.org/10.1016/j.jocs.2022.101937
5. Eshtehardian, S. A., Khodaygan, S. A continuous RRT*-based Path Planning Method for Non-Holonomic Mobile Robots Using B-Spline Curves. Journal of Ambient Intelligence and Humanized Computing, 2022, 14, 8693-8702. https://doi.org/10.1007/s12652-021-03625-8
6. Han, G., Jiang, J., Chao, J., Yang, X. Path Planning for a Group of Mobile Anchor Nodes Based on Regular Triangles in Wireless Sensor Networks. Neurocomputing, 2017, 270, 198-208. https://doi.org/10.1016/j.neucom.2016.10.097
7. Huang, H., Savkin, A. V. Viable Path Planning for Data Collection Robots in a Sensing Field with Obstacles. Computer Communications, 2017, 111, 84-96. https://doi.org/10.1016/j.comcom.2017.07.010
8. Kim, J., Yang, G.-H. Improvement of Dynamic Window Approach Using Reinforcement Learning in Dynamic Environments. International Journal of Control, Automation and Systems, 2022. https://doi.org/10.1007/s12555-021-0462-9
9. Lamini, C., Benhlima, S., Elbekri, A. Genetic Algorithm Based Approach for Autonomous Mobile Robot Path Planning. Procedia Computer Science, 2018, 127, 180-189. https://doi.org/10.1016/j.procs.2018.01.113
10. 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, 2023, 227, 120254-120254. https://doi.org/10.1016/j.eswa.2023.120254
11. Ma, G., Duan, Y., Li, M., Xie, Z., Zhu, J. A Probability Smoothing Bi-RRT Path Planning Algorithm for Indoor Robot. Future Generation Computer Systems, 2023, 143, 349-360. https://doi.org/10.1016/j.future.2023.02.004

12. Sabiha, A. D., Kamel, M. A., Said, E., Hussein, W. M. Real-Time Path Planning for Autonomous Vehicle Based on Teaching-Learning-Based Optimization. Intelligent Service Robotics, 2022, 15(3), 381-398. https://doi.org/10.1007/s11370-022-00429-3

13. Senturk, I. F., Akkaya, K., Janansefat, S. Towards Realistic Connectivity Restoration in Partitioned Mobile Sensor Networks. International Journal of Communication Systems, 2014, 29(2), 230-250. https://doi.org/10.1002/dac.2819

14. Sui, F., Tang, X., Dong, Z., Gan, X., Luo, P., Sun, J. ACO+PSO+A*: A Bi-Layer Hybrid Algorithm for Multi-Task Path Planning of an AUV. Computers & Industrial Engineering, 2023, 175, 108905. https://doi.org/10.1016/j.cie.2022.108905

15. Wang, B., Ju, D., Xu, F., Feng, C. Bi-RRT*: An Improved Bidirectional RRT* Path Planner for Robot in Two-Dimensional Space. IEEJ Transactions on Electrical and Electronic Engineering, 2023, 18(10), 1639-1652. https://doi.org/10.1002/tee.23898

16. Wu, D., Sun, Y., Wang, X., Wang, X. AN IMPROVED RRT ALGORITHM FOR CRANE PATH PLANNING. International Journal of Robotics and Automation, 2016, 31(2). https://doi.org/10.2316/Journal.206.2016.2.206-4180

17. Wu, L., Huang, X., Cui, J., Liu, C., Xiao, W. Modified Adaptive Ant Colony Optimization Algorithm and Its Application for Solving Path Planning of Mobile Robot. Expert Systems with Applications, 2023, 215, 119410-119410. https://doi.org/10.1016/j.eswa.2022.119410

18. Xie, C., Wang, Y., Liu, Y., Li, Z., Zhu, J., Qin, J. An AUV Path Planning Method Based on Improved APF-RRT*. 2023 IEEE International Conference on Mechatronics and Automation (ICMA), 2023, 1190-1195. https://doi.org/10.1109/ICMA57826.2023.10216242