RESEARCH ARTICLE | APRIL 06 2023

Path planning of indoor mobile robot based on improved A* algorithm incorporating RRT and JPS

Zengzhen Mi ; Hongjian Xiao; Chonghua Huang

AIP Advances 13, 045313 (2023)
https://doi.org/10.1063/5.0144960

Path planning of indoor mobile robot based on improved A* algorithm incorporating RRT and JPS

Cite as: AIP Advances 13, 045313 (2023); doi: 10.1063/5.0144960
Submitted: 2 February 2023 • Accepted: 24 March 2023 •
Published Online: 6 April 2023

Zengzhen Mi,[a] Hongjian Xiao, and Chonghua Huang

AFFILIATIONS

College of Mechanical Engineering, Chongqing University of Technology, No. 69, Hongguang Avenue, Chongqing 400054, China

[a]Author to whom correspondence should be addressed: mizengzhen@163.com

ABSTRACT

Indoor mobile robots are widely used in modern industry. Traditional motion control methods for robots suffer from discontinuous path curvature, low planning efficiency, and insufficient verification of theoretical algorithms. Therefore, a motion control system for an intelligent indoor robot was designed. By optimizing the radar map detecting and positioning, path planning, and chassis motion control, the performance of the system has been improved. First, a map of the warehouse environment is established, and the number of resampling particles interval is set for the Gmapping building process to improve the efficiency of map construction. Second, an improved A* algorithm is proposed, which converts the path solution with obstacles between two points into the path solution without obstacles between multiple points based on the Rapidly expanding Random Trees and Jump Point Search algorithms and further improves the pathfinding speed and efficiency of the A* algorithm by screening the necessary expansion nodes. The Dynamic Window Approach (DWA) algorithm based on the dynamic window is used to smooth the path, and the target velocity is reasonably assigned according to the kinematic model of the robot to ensure the smooth motion of the chassis. By establishing raster map models of different sizes, the traditional and improved A* pathfinding algorithms are compared and validated. The results illustrate that the improved pathfinding algorithm reduces the computing time by 67% and increases the pathfinding speed by 47% compared with the A* algorithm. Compared with the traditional method, the speed and effect are greatly improved, and the motion control system can meet the requirements of autonomous operation of mobile robots in indoor storage.

I. INTRODUCTION

In indoor robot path planning, it is necessary to sense the surroundings, fuse the environmental data, and build the environmental map through various sensors. The raster environment modeling method is extensively used in path planning because of its simplicity and effectiveness and easy implementation. The path-planning algorithms based on the grid method include the Dijkstra algorithm, the A*(A-star) algorithm, the D* algorithm, and so on. For static or quasi-static scenarios such as warehousing environment, Dijkstra and A* algorithms can find the optimal path effectively.

The traditional Dijkstra and A* algorithms have some shortcomings. Both algorithms need to maintain the two lists of "OpenList and CloseList" in the pathfinding process. In the process of node expansion, the nodes are detected at all times to query the nodes with the lowest cost. Dijkstra's breadth-first divergent search method has high space complexity and time complexity. A* algorithm combines the ideas of Dijkstra's algorithm by adding the cost function, making the Dijkstra-based search strategy directional.[1] Although it can reduce the computing time of path solving problem, this method also requires a high cost for path solving problem under non-depth optimal conditions, and its computing complexity and computing time are also huge when the paths are long, and the pathfinding efficiency is not high.

To solve the above problems, Zhao Hong *et al.* proposed a modified Dijkstra algorithm by using a bi-directional search technique, which improved the divergent search by searching clockwise or counterclockwise direction, significantly reducing the computing

time of the node cost function, but the pathfinding method is still a region-wide calculation.[2,3] Korf *et al.* improved the A* algorithm, which is an iterative deepening (depth-first) of the A* algorithm. Pruning optimization by the evaluation function eliminates the need for valuation sorting and reduces the consumption of search memory, thus optimizing the pathfinding strategy of the A* algorithm. Harabor *et al.* proposed a Jump Point Search (JPS) algorithm, which avoids unnecessary calculations by screening representative jump points for computation, thus effectively improving the pathfinding efficiency. LaValle *et al.* proposed Rapidly expanding Random Trees (RRT), a search method with incremental sampling, which can quickly and efficiently search the high-dimensional space and direct the search to the barrier-free region to find a safe and efficient path, but the random expansion is highly correlated with the size of the regional space and has a large impact on the pathfinding time and path planning with large uncertainty.[4–7]

To solve the problems of Dijkstra's and A* algorithms, such as large memory occupation, long computation time, and low pathfinding efficiency, this paper combines RRT and JPS algorithms to improve the A* algorithm.

(1) Find multiple nodes on the path by the RRT algorithm.
(2) Filter these nodes, keep the necessary nodes, and arrange them in the order of path expansion.
(3) Calculate the path by the JPS algorithm to reduce the expansion of unnecessary nodes. The improved algorithm not only reduces the system memory usage but also effectively improves the pathfinding strategy by converting the solution problem with obstacles between two points into a solution problem without obstacles between multiple points.
(4) The dynamic window approach (DWA) local path planning algorithm is used to smooth the path, and the pathfinding efficiency is improved by setting a reasonable speed resolution based on a smooth path.[8–11]
(5) The integrated development environment is used to build grid maps of different sizes to verify the feasibility and optimization effect of the improved A* algorithm.

In addition, the map-building algorithm and the underlying driver are also optimized. In the Mapping and Localization of the Gmapping algorithm, the memory is substantially relieved by reducing the number of particles, reducing the computation of iterative updates, improving the response in real time, increasing the frequency of map updates, and improving the efficiency of map building while ensuring the diversity of particles.[12] When the target velocity is transmitted to the driver board, the target velocity is reasonably allocated for the motion characteristics of the McNamee wheel, and the velocity is controlled to reach the target value by using micro-increments to make the cart move smoothly. Using the McNamee wheel indoor mobile robot as an experimental platform, the effect of the application of the kinematic control system is analyzed qualitatively and quantitatively.[13]

II. Gmapping map-building algorithm

Gmapping is a 2D SLAM algorithm based on the Rao-Blackwellized Particle Filters (RBPF) algorithm proposed by Grisetti *et al.* It is based on 2D LiDAR to complete the building of 2D grid maps. The steps of the specific algorithm are as follows:

(1) After Gmapping initializes the particle swarm, iterates over the particles of the previous moment, takes the poses, weights, and map data carried by them, and updates the poses through odometer coordinate conversion. And the proposed distribution area is confirmed according to the particles of the previous moment.[14]
(2) Find the local extrema by the maximum likelihood estimation, and if there are no local extrema, perform the particle positional update and calculate the particle weights using the observation model; otherwise, take a positional in the vicinity of the local extrema (considering that the particle population positional obeys Gaussian distribution), and find the mean, weight, and variance of this positional, first normalize the mean and variance and then use the multivariate normal distribution to approximate the new positional, and calculate the particle weights.[15,16]
(3) Update the particle population and map.
(4) Calculate the dispersion of all particle weights and determine whether to resample.
(5) Proceed to the next cycle.

The workflow diagram of the Gmapping algorithm is shown in Fig. 1.

The method has the following problems, although Gmapping uses the proposed distribution area to limit the number of sampled particles to avoid the memory explosion problem, the number

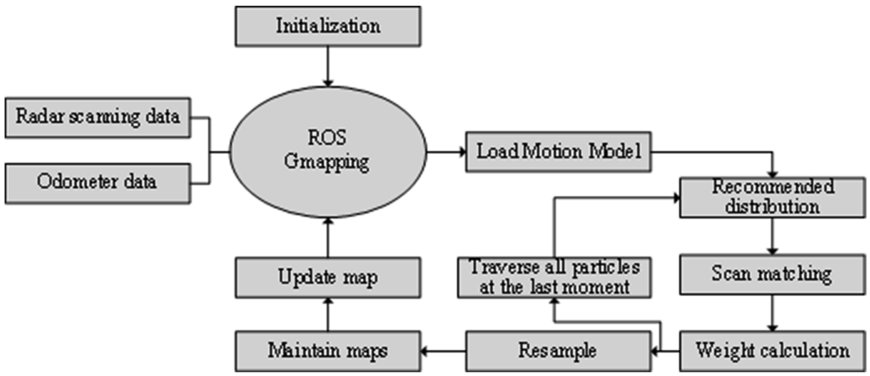

FIG. 1. Workflow of Gmapping algorithm.

of sampled particles is still large, and multiple iterations will also cause a huge amount of computation, affecting the real-time positioning; second, the degree of dispersion of the particle population as the basis for resampling and cannot completely solve the particle dissipation problem, but only slow down the particle dissipation rate.[17]

Due to the application scenarios such as indoor storage, the environment is more stable and the obstacle shape is more regular. Therefore, this paper gives the following solution, so the number of resampled particles can be set by setting the resampling particle number interval, limiting the upper limit of the number of resampled particles, improving the real-time map building and positioning at the same time, ensuring the minimum number of sampled particles, avoiding particle dissipation, and ensuring the accuracy of positioning, this experiment sets the number of particles interval for 20–30.

III. IMPROVEMENT OF PATHFINDING ALGORITHM

A. Dijkstra and A* pathfinding algorithms

1. Dijkstra's pathfinding algorithm

Dijkstra algorithm is a shortest path routing method that is used to calculate the minimum moving cost from one node to all other nodes. The main characteristic is to expand from the initial point outward in layers until it expands to the target node.[18–20] The basic principle is to create two lists, Openlist to hold the points of the path to be determined and Closelist to hold the points of the determined path, all points record their parent node and the distance to the initial point. The first step is to put the initial point into the Openlist, take the node n in the Openlist with the smallest distance from the initial point as the center point, remove it from the Openlist and put it into the Closelist table, then put the neighboring nodes of node n that have not been accessed into the OpenList, and repeat the above steps. Until the expansion to the target node, according to the recorded information of the parent node backpropagation, extract the path.

The algorithm has the following problems. Since the algorithm uses a greedy strategy to expand outward from the initial point to the objective point layer by layer, it will cause the system memory to be heavily occupied, the pathfinding computation is too large, the real-time performance is poor, and thus, the pathfinding efficiency is low.

2. A* pathfinding algorithm

The A* algorithm searches in the direction of the convergence target, and each node in the search direction is examined during the search. When a node is reached, the nodes around the node are added to the Openlist, then select the node with the smallest estimate as the next extension node, put the node into Closelist, and repeat the process until the target node is added to CloseList to complete the pathfinding. The A* algorithm pathfinding process is shown in Fig. 2.

Table I shows the comparison of expanding nodes and pathfinding time under different obstacles (measured under 0.5 s delay of single node expansion).

The A* algorithm generates the shortest path; however, as the complexity of the obstacles increases, the number of expanding nodes and the total pathfinding time of the A* algorithm also increase, i.e., the obstacles have a greater impact on the A* pathfinding, which will lead to serious memory consumption, large computation, and poor real-time when path planning is performed in more complex scenarios; the non-black colored areas in the figure are the nodes visited during the pathfinding process, while the purple raster part is not associated with the final path, so these rasters cause a large number of unnecessary operations, and the above are caused by the search strategy of the algorithm.

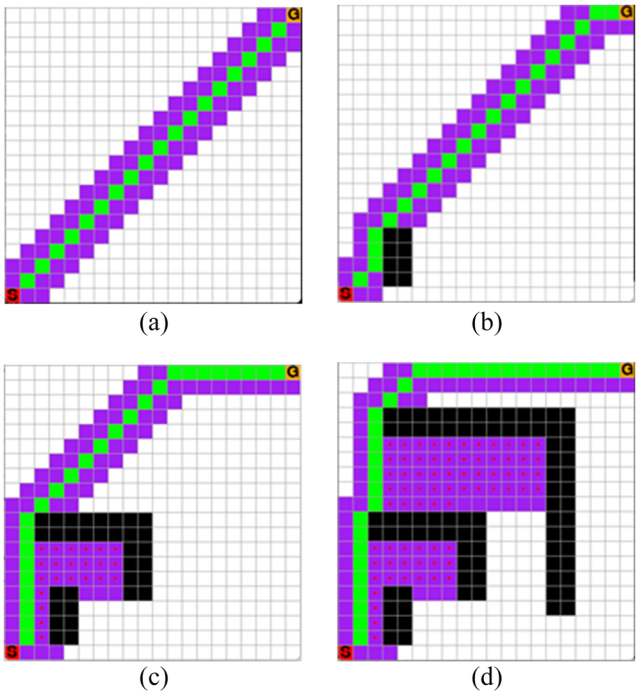

FIG. 2. A* algorithm pathfinding process under different obstacle conditions. (a) Barrier-free environment. (b) Simple barrier environment. (c) Intermediate barrier environment. (d) Complex barrier environment.

TABLE I. Extended nodes and time of A* algorithm pathfinding process in different obstacle environments.

Obstacle level	a	b	c	d
Number of expansion nodes/pc	90	88	110	156
Total pathfinding time/s	15.5	17	34.5	67.5

B. Improvement of A* algorithm

To solve the problems of large computation, serious memory consumption, and low pathfinding efficiency in the pathfinding process of the A* algorithm, this paper combines the RRT and JPS algorithms to improve and optimize the traditional A* algorithm.

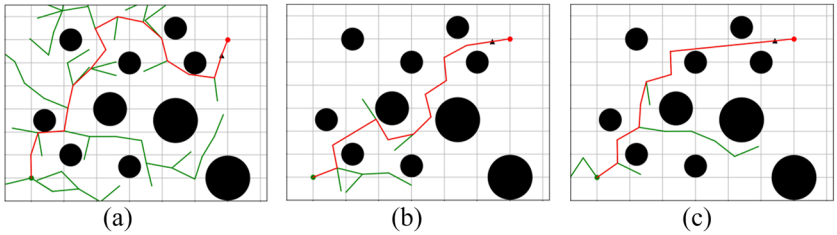

FIG. 3. RRT algorithm pathfinding effect. (a) Random sampling expansion process. (b) Assignment probability expansion process 1. (c) Assignment probability expansion process 2.

1. Pathfinding used RRT

The fast extended random tree algorithm is an incremental building method originally designed to efficiently search a high-dimensional nonconvex space, and this paper combines its application to a two-dimensional space to implement path planning for two-dimensional maps.[1] The effect of RRT pathfinding is shown in the following.

The RTT algorithm shown in Fig. 3(a) has two problems: (i) The random expansion property makes too many branches of the random tree, which leads to the expansion of a large number of unnecessary nodes and long pathfinding time and low efficiency. (ii) The red dashed segment is the final generated path, which is too tortuous and does not meet the requirements of robot path planning and motion.

To address problem (i), the objective point is assigned a random sampling probability so that it expands randomly with a high probability in the direction of the objective point. Figures 3(b) and 3(c) show two expansion processes with the same starting point and target point and a random sampling probability of 0.7 for the target point, which lead to inconsistent planning paths due to the random expansion. The way of assigning the sampling probability to the target point can significantly reduce the branches of the extended tree, i.e., significantly reduce the expansion of unnecessary nodes.

To address problem (ii), this paper proposes the method of retrieving the necessary nodes in order to smooth the expansion path, placing the expansion nodes in list 1 in the order from the initial point to the objective point, retrieving the necessary nodes in the order in list 2, retrieving over as shown in Fig. 4.

The effect of this method is shown in Fig. 5. It is clear that the zigzag path is made smoother by retrieving the necessary nodes on the path.

2. Pathfinding used JPS

The JPS algorithm is a new search strategy, and the so-called hopping point is the node that can be directly jumped during the search process. As shown in Fig. 6(a), there are three paths from node S to node G, which are S → A → G, S → B → G, and S → C → G. A* algorithm visits and operates each adjacent node (A, B, C) in the pathfinding process; obviously, it is meaningless to operate on two nodes A and C. To further improve the search speed, the search can be done directly in the direction of S → B → G, which is the search strategy of the JPS algorithm.

Figures 6(b) and 6(c) show the pathfinding process of the A* algorithm and the A* algorithm with fused JPS under the same environment map, respectively. Noticeably, the A* algorithm with fused JPS reduces the operation of numerous unnecessary nodes in the expansion process compared to the A* algorithm, and the quantity of expansion nodes is reduced by an order of magnitude, which effectively reduces the computation and memory consumption of the algorithm in the pathfinding process.

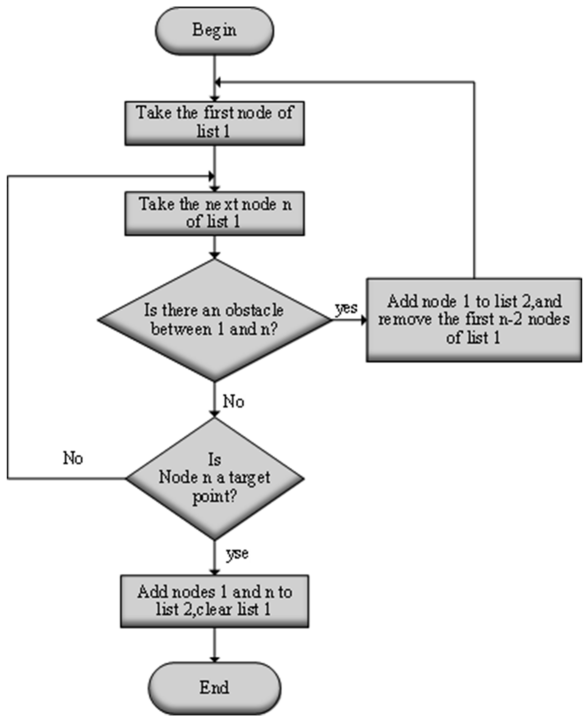

FIG. 4. Process of retrieving necessary nodes by improved RRT algorithm.

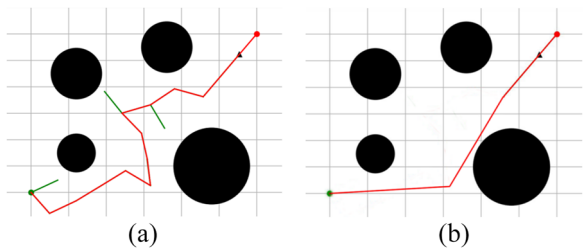

FIG. 5. Effect of improved RRT retrieval nodes. (a) Path before retrieving nodes. (b) Path after retrieving nodes.

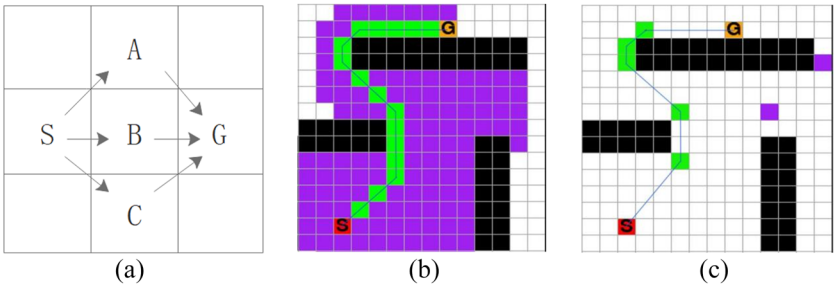

FIG. 6. A* pathfinding with fused JPS. (a) Jump point search process. (b) A* pathfinding process. (c) A* pathfinding process with fused jump points.

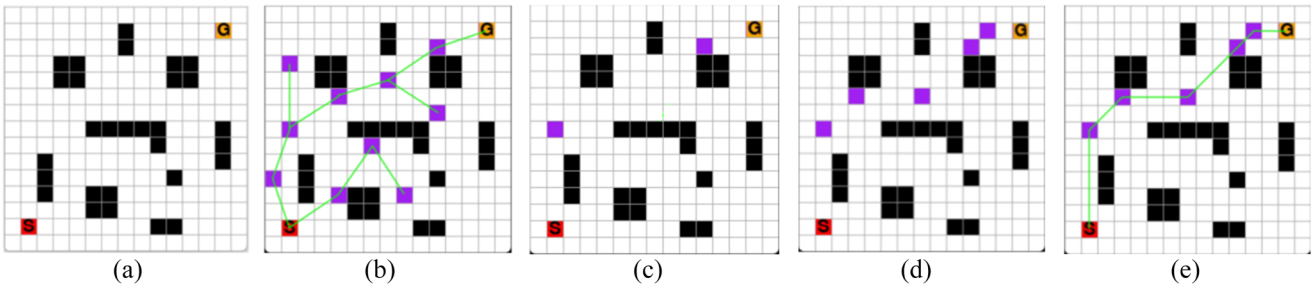

FIG. 7. Steps to improve the A* algorithm. (a)–(e) Improved A* algorithm pathfinding: step 1 to step 5.

3. Improved A algorithm incorporating RRT and JPS*

Therefore, this paper proposes to combine RRT and JPS algorithms to improve the A* algorithm. The specific steps are to first use the RRT algorithm to initially determine the path and screen the necessary extension nodes, and then use the A* algorithm that incorporates jump point search to solve the path between adjacent nodes so that the path solution with obstacles is transformed into the path solution without obstacles. Figure 7 shows the flow of the improved A* algorithm.

C. Chassis motion smoothing treatment

The control command of SLAM is based on the speed information of the vehicle motion model, while the motion control of the chassis needs to combine the wheel motion characteristics and the cart motion model to assign the target speed.

1. McNamee wheel chassis kinematic model

The McNamee wheel, which can move arbitrarily in all directions, consists of wheel axles and roller shafts and is spatially at a 45°angle, as shown in Fig. 8.

Therefore, the velocity direction of the chassis is 45° to the wheel axis direction, each wheel has three degrees of freedom, and each McNamee wheel can be regarded as a prime mover, so at least three McNamee wheels are needed to realize the control of three degrees of freedom in the plane. In practical applications, four McNamee wheels are usually used as a group, two for each of the left and right spinning wheels, and the left and right wheels are symmetrical in the left- and right-hand structures and different installation methods determine the difference of their motion control effects.

The overall motion of the chassis is considered as rigid body motion, then the motion of the rigid body in the plane can be decomposed into three independent components, and the motion speed of the chassis as Vt can be decomposed into forward and backward moving speed V_x, left and right moving speed V_y, and the rotation speed of the chassis around the center point V_z. V_A, V_B, V_C, and V_D are the linear speeds of McNamee wheels A, B, C, and D, respectively. The execution of the Robot Operating System (ROS) control command is to assign the target speed to each McNamee wheel according to the relationship between V_x, V_y, V_z and V_A, V_B, V_C, V_D. Their relationship can be expressed by the following equation.

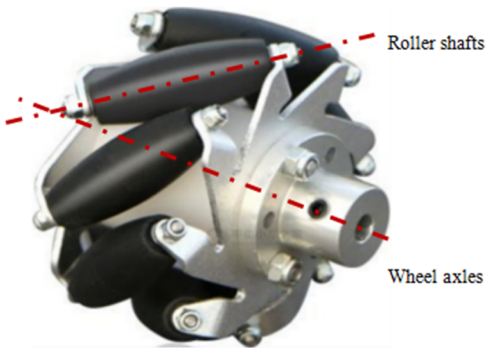

FIG. 8. McNamee wheel structure.

(a) (b) (c)

FIG. 9. Indoor mobile with the construction of its human hardware platform. (a) Design of PCB board. (b) Production of control board. (c) Experimental platform.

$$\begin{aligned} V_A &= V_x + V_y - V_z\left(\frac{H}{2} + \frac{W}{2}\right), \\ V_B &= V_x - V_y - V_z\left(\frac{H}{2} + \frac{W}{2}\right), \\ V_C &= V_x + V_y + V_z\left(\frac{H}{2} + \frac{W}{2}\right), \\ V_D &= V_x - V_y + V_z\left(\frac{H}{2} + \frac{W}{2}\right), \end{aligned} \tag{1}$$

where W is the center distance between the left and right wheat wheels of the trolley, and H is the center distance between the front and rear wheat wheels. Also, the real-time speed of the trolley needs to be fed back to the ROS as follows:

$$\begin{aligned} V_x &= \frac{V_A + V_B + V_C + V_D}{4}, \\ V_y &= \frac{V_A - V_B + V_C - V_D}{4}, \\ V_z &= \frac{-V_A - V_B + V_C + V_D}{4}. \end{aligned} \tag{2}$$

2. Smoothing process

The third step is the smoothing of robot motion. The control command given by the SLAM scheme is based on the velocity information of the motion model. The robot motion control, on the other hand, needs to combine the wheel motion characteristics and the vehicle motion model to assign the target speed, and it needs to control the wheel speed to change to the target speed. The wheel speed cannot jump directly to the target speed, which requires a smooth increment to control the speed to reach the target value. The selection of this increment will determine the smoothness of the vehicle operation: If the selected value is too small, below the speed resolution, an increment almost does not change the speed but will make the vehicle travel slowly, and if the selected value is too large, it will cause the robot movement deformation or stalling. In this experiment, the wheel rotates one week, the encoder count is 400, the motor control task execution frequency is 100 Hz, the wheat wheel diameter is 7 cm, the motor reduction ratio is 1:30, and, thus, the single speed increment is selected as 0.01 m/s.

IV. SIMULATION AND EXPERIMENT

A. Hardware platform for indoor mobile robots

The STM32 is used as the control kernel of the underlying driver board. The model of LIDAR is Silan A2M8, which is capable of 360° laser scanning in the 2D plane; the model of IMU (Inertial Measurement Unit) is ICM-20948, which contains a 9-axis gyroscope and can reduce the data processing burden of STM32; the DC geared motor with photoelectric encoder MG513P30 is used to achieve high precision mileage calculation. The power supply circuit, gyroscope circuit, encoder data acquisition circuit, motor driver circuit, and Universal Serial Bus (USB) hub circuit are designed. The circuit schematic and Printed Circuit Board (PCB) are carried out by using an Electronic Differential Analyzer (EDA). The PCB design is shown in Fig. 9(a), the soldering object is shown in Fig. 9(b), and the hardware connection object is shown in Fig. 9(c).

B. Framework of the software

STM32 control board is responsible for multi-sensor configuration and data acquisition, STM32 has many management tasks, if executed sequentially, the communication efficiency with the ROS will be greatly reduced, which will lead to lagging data received by the ROS at the current moment, resulting in wrong operation of the ROS on the motion. Therefore, the embedded real-time system FreeRTOS is used to manage various tasks, which provides the functions of resource management, synchronization, and task communication required by the real-time operating system, effectively solving the problem of low ROS communication efficiency, and the basic framework of the software is shown in Fig. 10.

C. Environment representation of raster maps

A series of raster maps of the same size are used to simulate the robot's working environment, with blank areas indicating that the robot is passable and shaded areas indicating that it is impassable

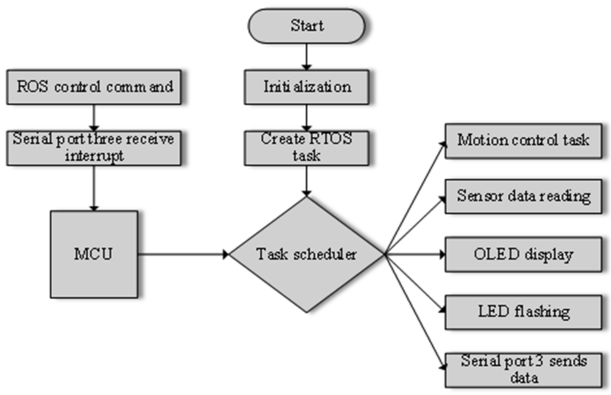

FIG. 10. Framework for STM32 control board software.

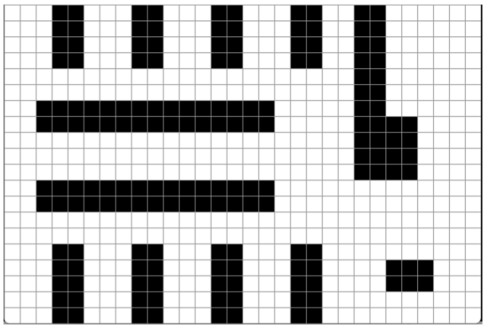

FIG. 11. Environment representation of the raster maps.

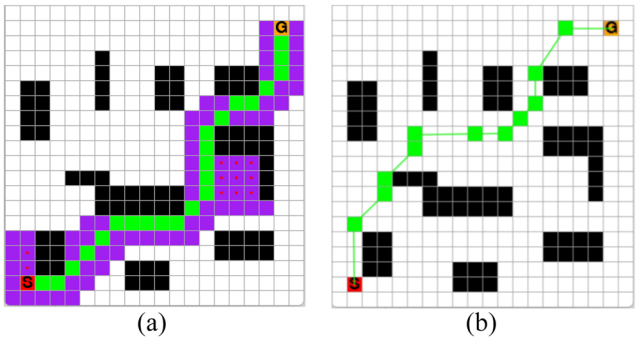

FIG. 12. Simulation results under obstacle density 1/5. (a) Results of A* algorithm. (b) Results of improved A* algorithm.

when occupied by obstacles. As shown in Fig. 11, the raster map has the following features: (1) The raster map model has considered the risk of collision between the obstacles and the trolley, and the blank areas are all safe areas where the trolley can pass. (2) The obstacle area detected by the robot is the shelf boundary, and no spatial collision will occur, so the simulation environment also does not consider the robot's height. (3) When there is no boundary/obstacle, the robot can move in the direction of the surrounding eight grids.

D. Simulation analysis

To verify the effectiveness of the improved A* algorithm, the pathfinding effects of the A* algorithm and the improved A* algorithm incorporating the RRT and JPS algorithms are simulated in the same environment, respectively. The computer configuration is Windows 11 operating system, AMD 4600U processor with 2.1 GHz, and 16 G running memory.

Simulations were performed in five raster maps of different sizes and similar obstacle complexity with 1/5 of the obstacle density of the raster map, and Fig. 12 shows the simulation results under a 20 × 20 raster map. The red grid indicates the initial node; the yellow grid indicates the objective node; the green and purple grids indicate the nodes that have been visited during the pathfinding process (where the green grid is the final actual path via node); and the green dash indicates the final path. It can be seen that the quantity of expansion nodes of the improved A* algorithm is significantly less than the traditional A* algorithm.

Table II gives the comparison between the search time (under the condition of 0.5 s delay for one node expansion), the number of expanded nodes, and the length of paths in different raster maps for the A* algorithm and improved A* algorithm.

For Table I, under the same obstacle density, as the map increases, the pathfinding time ratio of the improved A* algorithm gradually decreases, the expanding node count ratio gradually decreases, and the advantages become more and more obvious.

The simulation was performed in five raster maps of the same size and different levels of obstacle complexity. Figure 13 shows the simulation analysis results for a 30 × 20 raster map with obstacle density 1/4. It is easy to see that in the case of more complex obstacles, the A* algorithm needs to traverse almost all the points, while the improved A* algorithm completes the pathfinding with only a few nodes, and the two paths overlap. Table IV statistically compares the search time, the expanding node count, and the path length of the A* algorithm with the improved A* algorithm under different obstacle densities.

As can be seen from Tables II and III, the traditional A* algorithm is a heuristic search, based on the valuation function and the current cost as the ranking criteria of the priority queue, while the improved A* algorithm operates on a much smaller number of hopping points, thus greatly reducing the complexity of the computation and reducing the memory consumption during the pathfinding process. The results show that the improved A* algorithm not only effectively improves the search speed of the A* algorithm but also

TABLE II. Performance comparison of A* algorithm with improved A* algorithm for different size maps.[a]

Map size	Search time/s		Number of expansion nodes		Path length		Pathfinding time ratio	Expanding node count ratio
	I	II	I	II	I	II		
10 × 10	3.5	13	6	34	27	25	0.27	0.17
20 × 20	6.5	25.5	12	94	59	56	0.25	0.13
30 × 20	9.5	41	14	104	91	79	0.23	0.13
30 × 30	13.5	61.5	11	96	106	90	0.21	0.11
40 × 30	15.5	74.5	14	142	128	116	0.20	0.10

[a]Remarks: **I** denotes the improved A* algorithm; **II** denotes the A* algorithm.

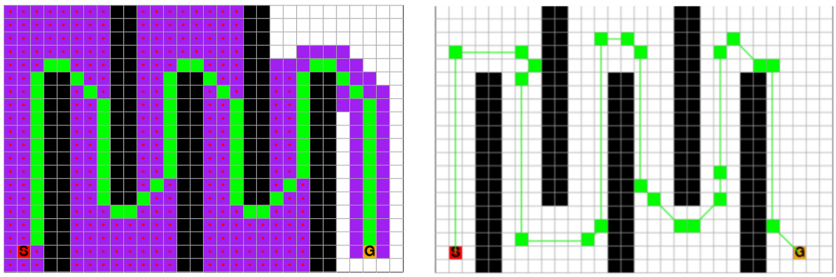

FIG. 13. 30 × 20 raster map simulation result. (a) Simulation result of A* algorithm. (b) Simulation result of improved A* algorithm.

TABLE III. Performance comparison of the A* algorithm and the improved A* algorithm under different obstacle densities.[a]

Map size	Search time/s		Number of expansion nodes		Path length		Pathfinding time ratio	Expanding node count ratio
	I	II	I	II	I	II		
1/6	8	42	16	89	69	61	0.03	0.06
1/4	7.5	197.5	22	360	178	171	0.04	0.06
1/3	6.5	131	18	168	84	71	0.05	0.11
2/1	6	72	11	194	121	96	0.08	0.06
2/3	5	57.5	12	127	128	116	0.09	0.09

[a]Remarks: **I** denotes the improved A* algorithm; **II** denotes the A* algorithm.

greatly reduces the number of expansion nodes in the search process. In terms of both the pathfinding efficiency and the optimization of the algorithm, the improved A* algorithm is significantly better than the A* algorithm.

E. Experimental analysis

The mobile robot experimental platform adopts McNam wheel chassis, equipped with Jetson nano system, the LiDAR sensor is Silan A2M8, and the inertial measurement unit (IMU) is ICM-20948. LiDAR's laser ranging sampling is up to 8000 times/second, and within 12 m, it can realize two-dimensional. The LiDAR can scan 360° in all directions in a 12 m range and generate the plane point cloud of its space.

First, the environment information is obtained by LiDAR, and the 2D point cloud data are generated by calculation, and the positioning and 2D map building is completed by using adaptive Monte Carlo Localization (AMCL) and Gmapping, then the map expansion processing, global and local path planning are realized, and finally, the velocity information is transmitted to the chassis, and the chassis does motion smoothing processing by combining the vehicle motion characteristics to complete the map building and navigation. To verify the performance of the indoor robot motion control system, this experiment is conducted in two steps.

(1) First step, the availability of the improved A* algorithm in the mobile robot experimental platform is verified. The experimental scene uses a laboratory building to simulate the storage environment, as shown in Fig. 14(a). First, the point cloud information of the environment is obtained, then the positioning and mapping tasks are completed, and finally, global and local path planning experiments are performed separately. The 2D raster map of the environment is presented by Rviz, a visualization tool of ROS: First, the vehicle is controlled to move to build a map of the environment, and then the path planning experiments are conducted on the built map to prove the efficiency of the improved A* algorithm by comparing the pathfinding efficiency of different

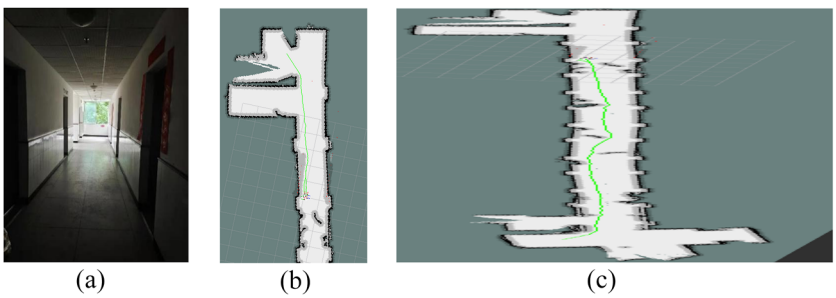

FIG. 14. Feasibility experiment of improving A* algorithm. (a) Experimental scenes 1. (b) Path 1. (c) Path 2.

TABLE IV. Time and path length for two path planning with different algorithms.

	Path 1		Path 2	
	A* algorithm	Improved A* algorithm	A* algorithm	Improved A* algorithm
Path length/m	11.8	11.4	41.6	40.2
Pathfinding time/ms	210.8	82.4	682.7	184.8

pathfinding algorithms under the same path. Figures 14(b) and 14(c) indicate the 2D raster map and path planning results of the improved A* algorithm. Table IV indicates the pathfinding time and path length of the two algorithms under different paths. The difference between the two algorithms in terms of path length is not significant, but the pathfinding time of the improved A* algorithm is significantly reduced, and its response speed is significantly better than that of the A* algorithm.

(2) Based on the first step, the Gmapping algorithm is introduced according to the designed SLAM optimization scheme, and the interval of the number of resampled particles is set to 20–30, limiting the upper limit of the number of resampled particles to improve the real-time performance of map building and positioning, while ensuring the minimum number of sampled particles to avoid particle dissipation and ensure the accuracy of localization. Figure 15(a) shows the warehouse simulation environment, Fig. 15(b) shows the 2D raster map of the corresponding condition, and Fig. 15(c) shows the result of pathfinding. The experimental results show that the optimized motion control system can perform the localization and navigation tasks well.

The experimental results show that the robot motion control system can complete its autonomous motion requirements, and the improved A* algorithm, which incorporates the RRT and JPS algorithms, can effectively complete the path planning of the mobile robot. The algorithm of this paper optimizes the pathfinding strategy, which is faster in computation and more efficient in pathfinding compared with the traditional path-planning method. The mobile robot runs smoothly without jitter and motion deformation during pathfinding.

FIG. 15. Mapping and positioning based on Gmapping grid map. (a) Experimental scenes 2. (b) Environmental raster map. (c) Planned path.

V. DISCUSSIONS

The number of Gmapping resampling particles affects the map-building effect when it is large. This paper combines the characteristics of the actual scene and sets the interval for the number of resampling particles, which has a more obvious effect on map-building efficiency and obstacle recognition. The disadvantage of the A* algorithm is that the memory overhead is large and the computation time is long in the pathfinding process, which cannot meet the real-time requirements of the mobile robot in planning paths in larger scenes. To improve the speed and efficiency of path planning, the RRT and JPS algorithms are combined to improve the A* algorithm and optimize the search strategy in the search process: First, the expansion nodes are initially retrieved by the RRT algorithm. Second, the necessary expansion nodes without obstacle occlusion are screened. Then, the path is solved between the screened nodes using the A* algorithm combined with the JPS and the representative jump points for expansion, which converts the path planning into multi-segment obstacle-free A* pathfinding and replaces the original A* algorithm for each adjacent node, thus improving the pathfinding efficiency. Simulations and experiments demonstrate that algorithms in this paper can significantly improve the pathfinding speed and reduce the memory overhead while generating near-optimal paths, especially in larger scenarios. In addition, by smoothing the speed, the designed motion control system can meet the application requirements of the quasi-static working environment of an indoor storage robot with an obvious optimization effect.

ACKNOWLEDGMENTS

This research was funded by the National Natural Science Foundation of China (Grant No. 61901068), the Science and Technology Foundation of CMEC (Grant Nos. KJQN201901150 and KJQN202001131), the Chongqing Natural Science Foundation (Grant No. cstc2021jcyj-msxmX0525), and the Chongqing Graduate Student Innovation Program (Grant Nos. gzlcx20222041 and CYS21467).

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Zengzhen Mi: Supervision (lead). **Hongjian Xiao**: Software (equal). **Chonghua Huang**: Data curation (equal).

DATA AVAILABILITY

All data, models, and code generated or used during the study appear in the submitted article.

REFERENCES

[1]S. Khalesidoost, J. Faiz, and E. Mazaheri-Tehrani, "An overview of thermal modelling techniques for permanent magnet machines," IET Sci., Meas. Technol. **16**(4), 219–241 (2022).

[2]S. Yu, C. Fu, A. K. Gostar *et al.*, "A review on map-merging methods for typical map types in multiple-ground-robot SLAM solutions," Sensors **20**(23), 69–88 (2020).

[3]X. Zhang, D. Rey, and N. Chen *et al.*, in *Shortest Path Navigation Algorithm for Driverless Plug-In Electric Vehicle* (Australasian Transport Research Forum, 2016).

[4]W. Ge, B. Wang, and H. N. University, "Global path planning method for mobile logistics robot based on raster graph method," Bull. Sci. Technol. **35**(11), 72–80 (2019).

[5]C. Liu, B. Lin, J. Lai, and D. Miao, "An improved decision tree algorithm based on variable precision neighborhood similarity," Inf. Sci. **615**, 152–166 (2022).

[6]J. Qi and L. Liu, "The stereo matching algorithm based on an improved adaptive support window," IET Image Process. **16**(10), 2803–2816 (2022).

[7]R. E. Korf, "Depth-first iterative-deepening: An optimal admissible tree search," Artif. Intell. **27**(1), 97–109 (1995).

[8]D. Harabor and A. Grastien, "The JPS path finding system," in *5th Annual Symposium on Combinatorial Search* (AAAI, Menlo Park, 2012), pp. 207–208.

[9]B. Kim, T. T. Um, C. Suh *et al.*, "Tangent bundle RRT: A randomized algorithm for constrained motion planning," Robotica **34**(01), 202–225 (2016).

[10]C. Lau and K. Byl, "Smooth RRT-connect: An extension of RRT-connect for practical use in robots," in *IEEE International Conference on Technologies for Practical Robot Applications* (IEEE, 2015), pp. 1–7.

[11]G. Im, M. Kim, and J. Park, "Parking line based SLAM approach using AVM/LiDAR sensor fusion for rapid and accurate loop closing and parking space detection," Sensors **19**(21), 4811 (2019).

[12]L. Hengjie, B. Hong and X. Cheng. "Fast closed-loop SLAM based on the fusion of IMU and Lidar," J. Phys.: Conf. Ser. **1914**(1), 012019 (2021).

[13]L. Hengjie, B. Hong, and X. Cheng, "Optimal parameter analysis of two 2D Lidar SLAM," World Sci. Res. J. **6**(1), 56–62 (2021).

[14]S. Wen, Y. Zhao, X. Yuan, Z. Wang, D. Zhang, and L. Manfredi, "Path planning for active SLAM based on deep reinforcement learning under unknown environments," Intelligent Serv. Rob. **13**, 263–272 (2020).

[15]H. Xing, Y. Liu, S. Guo *et al.*, "A multi-sensor fusion self-localization system of a miniature underwater robot in structured and GPS-denied environments," IEEE Sens. J. **21**(23), 27136–27146 (2021).

[16]M. Dong, D. Chen, N. Ye, and X. Chen, "Design of ackerman mobile robot system based on ROS and Lidar," J. Phys.: Conf. Ser. **1838**(1), 012073 (2021).

[17]I. Afanasyev, A. Sagitov, and E. Magid, "ROS-based SLAM for a gazebo-simulated mobile robot in image-based 3D model of indoor environment," in *International Conference on Advanced Concepts for Intelligent Vision Systems* (Springer, Cham, (2015).

[18]J. Xu,G. Liu, J. Liu *et al.*, "EKF-based positioning study of a mobile robot with McNamee wheels," J. Phys.: Conf. Ser. **2281**, 012008 (2022).

[19]C. Chen, H. Zhu, L. Wang, and Y. Liu, "A stereo visual-inertial SLAM approach for indoor mobile robots in unknown environments without occlusions," IEEE Access **7**, 185408–185421 (2019)

[20]H. Williams, W. N. Browne, and D. A. Carnegie, "Learned Action SLAM: Sharing SLAM through learned path planning information between heterogeneous robotic platforms," Appl. Soft Comput. **50**, 313–326 (2017).