Paper:

Obstacle Avoidance and Path Planning of Intelligent Robots Based on Optimized A* and DWA algorithms

Zhenjiang Hu[1], Ying Jin[1*], Hongbin Ma[1], Tao Fu[1]

Beijing Institute of Technology, Beijing 100081, P. R. China
E-mail: mathmhb@bit.edu.cn

Abstract. Typically, mobile robots work in an environment that is not fully known, such as wilderness, where they need to move from a specified starting point to a desired location. The traditional A* algorithm is based on Dijkstra algorithm to increase the heuristic function (Euclidean distance between the end point), which greatly reduces the number of grids included while obtaining the optimal path. However, in the search process, the number of traversed grids is still large and the planned path is not smooth. In the view of the above problems, the Euclidean distance in the heuristic function is changed to the diagonal distance and the Manhattan distance, and different weights are given. The Bezier curve is used for smoothing. Finally, the optimized A* algorithm is combined with the DWA algorithm to quickly reach the destination while bypassing the obstacles.

Keywords: Mobile robot, Path planning, A* algorithm, Heuristic function, Optimized A* algorithm, DWA algorithm

1. Introduction

Usually, a mobile robot works in an environment that are not fully known. In this article, we will discuss in detail how mobile robots plan an appropriate path from the specified starting point to the destination. Path planning is one of the key issues in the field of mobile robot research. Path planning can be roughly divided into global path planning and local path planning. In an unknown environment, our information about obstacles is also unknown. Therefore, while avoiding obstacles, we can quickly reach the designated position, which is the purpose of this paper.

The traditional path planning obstacle avoidance algorithm is mainly divided into sampling-based path planning algorithm and search-based path planning algorithm. Search-based path planning algorithms mainly include the artificial potential field method[1], Dijkstra algorithm[2], A* algorithm[3], grid method[4], topology method[5], D * algorithm[6], and D * lite algorithm[7, 8]. Sampling-based path planning mainly includes RRT algorithm[9], RRT * algorithm[10] and Informed RRT algorithm[11].

In this paper, A* algorithm is selected as the main research object. A* algorithm adds a heuristic function h(n) based on Dijksta algorithm, whrere h(n) is the expected consumption from the specified grid to the end point, making its cost function f(n)=g(n)+h(n). Thus, the number of contained grids is reduced and the path planning results are obtained quickly.

2. Design of Autonomous Mobile Robot

This chapter describes the software and hardware system of mobile robot

2.1. Hardware System Design of Mobile Robot

The car shown in Fig. 1 selects STM32 as the main control board to design the bottom plate of the car. The Omni-directional wheel is used, L298n is used as the motor drive module, and the encoder and MPU6050 are used to measure the wheel information. At the same time, the ultrasonic module is used for obstacle avoidance.

The following are the design of each module:

1) STM32, Arduino, and 51 are the same main board controllers. The difference is that the programming and wiring are complex, and most library functions cannot be directly called, but the control is more accurate.

2) L298n motor drive module is widely used because of its strong driving ability, low calorific value, large output current, and high reliability.

3) The encoder can be achieved by installing receivers and transmitters on both sides of the encoder aperture and then measuring the rise and fall edges of phase A and phase B.

4) MPU6050 has three accelerometers and three gyroscopes, which are used to measure the pose information of the mobile robot. In this case, the angle of the mobile robot is applied to correct the steering of the vehicle by the angle difference with the planned path.

5) The ultrasonic module continuously emits and receives ultrasonic waves at a higher frequency, and measures the distance between the obstacle and the ultrasonic wave by recording the time of ultrasonic travel and return.

6) Bluetooth module is sensitive, easy to develop, and cost-effective, which is used to realize the data transmission between the computer end and the car to control the movement of the car.

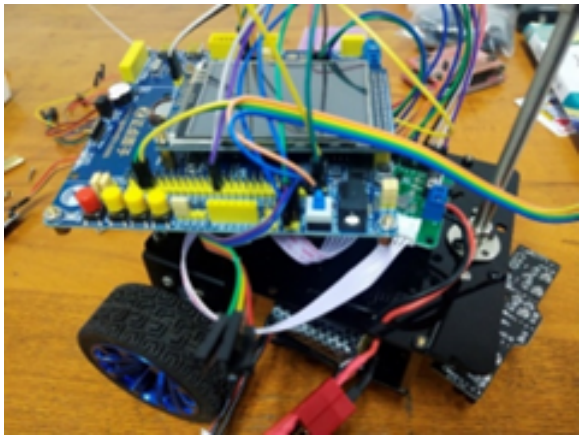


Fig. 1. Car building model

2.2. Software System Design of Mobile Robot

Fig. 2 is the robot software framework. The mobile robot uses the Ubuntu18.04 system to build the ROS robot operating system and completes the construction of the grid map, the construction of the mobile robot model, the corresponding path planning content, and the underlying drive on the ROS.

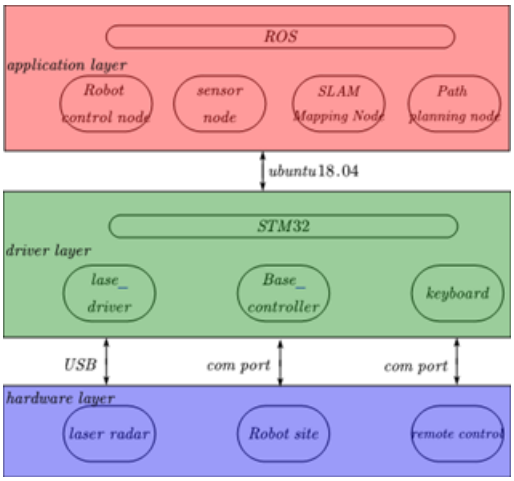


Fig. 2. Robot software framework

The whole software is divided into three layers: application layer, driver layer, and hardware layer. The bottom layer of the hardware is the chassis of the mobile robot and the encoder, laser radar, IMU, and other sensors. The encoder and IMU can output the pose of the intelligent robot in real-time. The laser radar sensor can obtain the detailed environmental grid map and the environmental information of the planning route for the intelligent robot. The driver layer is used to make the hardware layer sensor work normally and keep running, at the same time through different ports (USB, serial, CAN bus, etc.) to write different sensor driver layers and chassis transmission programs; the application layer is based on the driver layer and hardware layer for information processing and transmission instructions, while achieving data transmission between the three layers to improve the stability of various functions.

3. Path Planning

Path planning is the most important step in mobile robot navigation. With the development and progress of technology, path planning algorithms are becoming more and more diverse. Some have been relatively maturely applied to real life, and some remain to be further studied. Through the analysis and comparison of different path planning algorithms, global path planning based on the A* algorithm and local path planning based on the DWA algorithm are selected.

3.1. Local Path Planning Based on DWA

DWA algorithm is called the dynamic window approach. For the DWA algorithm, Fig. 3 shows the flow chart of the algorithm. This section introduces the DWA algorithm in detail according to the process.

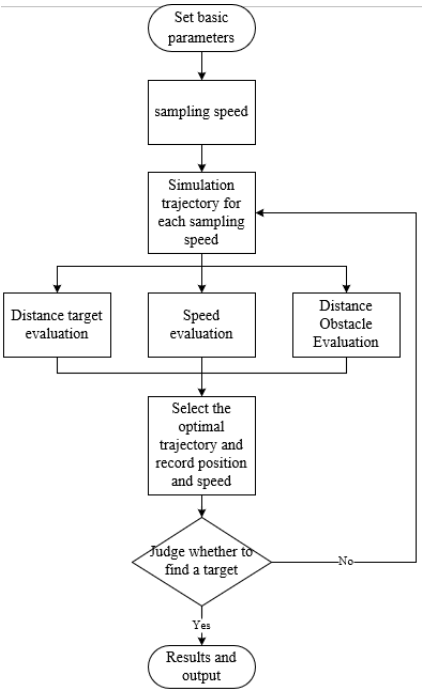


Fig. 3. Flow chart of DWA algorithm

3.1.1. Motion Model of Mobile Robot

The motion model is the simulation of the future di-

rection of the mobile robot, so it is particularly critical to understand the motion model of the mobile robot. Knowing the motion model, it is better to make the mobile robot move

in the direction of good planning, and at the same time, it is also conducive to real-time path planning of the robot.In the dynamic window algorithm, it is assumed that the path that the mobile robot can walk makes a straight line or arc, that is, it cannot travel horizontally, that is, it does not use Omni-directional wheels. When calculating the trajectory of the robot at adjacent moments, because the time interval between adjacent moments is short, the trajectory can be considered as a straight line. Fig. 4 assumes that the interval is Δt, then the path in the abscissa is v_x*Δt, and the path in the ordinate is v_y*Δt

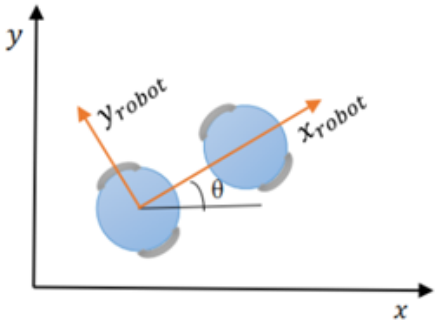


Fig. 4. Motion models of mobile robot

Then the relative movement Δx and Δy in the coordinate system are:

$$\Delta x = v\Delta t\cos(\theta_t) \quad . . . . . (1)$$

$$\Delta y = v\Delta t\sin(\theta_t) \quad . . . . . (2)$$

If the displacement is calculated in this period, according to (1) and (2), it can be known that

$$x^{n+1} = x^n + v\Delta t\cos(\theta_t) \quad . . . . . (3)$$

$$y^{n+1} = y^n + v\Delta t\sin(\theta_t) \quad . . . . . (4)$$

$$\theta^{n+1} = \theta^n + \omega\Delta t \quad . . . . . (5)$$

Through the above formula, we can know its motion model, but obviously, the speed is unknown, that is, the relative displacement is known. For speed sampling, it is mainly limited to the following three factors:

1) The limit of maximum speed and minimum speed

$$v_m = \{v \subset [v_{min}, v_{max}], \omega \subset [\omega_{min}, \omega_{max}]\} \quad (6)$$

2) Influence of motor performance
For each motor, its power is always limited, resulting in a large acceleration and deceleration time limit, and thus the required speed can be obtained in this limited time window.

$$v_d = \{(v,\omega)|v \subset [v_c - c_b\Delta t, v_c + v_a\Delta t] \cap \omega \subset [\omega_c - \omega_b\Delta t, omega_c + \omega_a\Delta t]\} \quad (7)$$

3) Safety Consideration Based on Mobile Robot
Similar to the safe distance of the driving vehicle, it is necessary to stop before the collision between the two vehicles to prevent accidents. The mobile robot also needs a distance or speed to prevent the robot from hitting the obstacle. Therefore, under the maximum deceleration condition, the speed needs to have a range of constraints.

$$v_a = \left\{(v,\omega)|v \le \sqrt{2dist(v,\omega)v_b} \cap \omega \le \sqrt{2dist(v,\omega)\omega_b}\right\} \quad . . . (8)$$

So far, we can know the motion model in the adjacent time, so if we want to know the trajectory in a period, we can sum it up.

3.1.2. Evaluation Function of Trajectory

After sampling the velocity, many trajectories can be obtained by substituting them into the motion model. Therefore, it is necessary to discuss and select the cost and feasibility of these trajectories. Similar to a scoring system, it is necessary to evaluate or score each trajectory, and ultimately find an optimal trajectory for local navigation. The scoring criteria have the following three factors:

1) Evaluation function of azimuth angle
Heading (v,ω) represents the azimuth angle of the mobile robot, as shown in Fig. 5. The mobile robot reaches the predicted position, and then the connection between it and the target and the angle between the motion direction of the mobile robot at this moment is θ. It can be seen from the figure that the smaller the value of θ is, the better it is. When θ is 0, that is, the trajectory is a straight line, that is, the trajectory with the minimum cost. However, if there are obstacles or other objects that hinder the movement between the two, the path will not work. Therefore, two other standard gaps and velocities are needed as references.

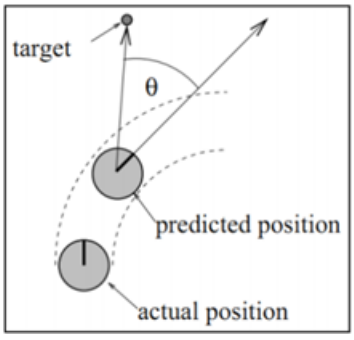


Fig. 5. Azimuth diagram

2) Void
Dist (v, ω) represents the gap between the mobile

robot. The gap refers to the distance between the robot and the nearest obstacle on the designed road. Assuming that there is no obstacle on this road, the coefficient is designed constantly.

3) Velocity
Velocity (v, ω) represents the speed of the mobile robot, that is, the speed of the mobile robot on this planned path.

Finally, these three factors, namely, azimuth evaluation function, gap, and velocity, are normalized, namely, smoothing.

$$normal_head(i) = \frac{head(i)}{\sum_{i=1}^{n} head(i)} \quad (9)$$

$$normal_dist(i) = \frac{dist(i)}{\sum_{i=1}^{n} dist(i)} \quad (10)$$

$$normal_velocity(i) = \frac{velocity(i)}{\sum_{i=1}^{n} velocity(i)} \quad (11)$$

After the weight processing, the three factors are combined to get the final evaluation function:

$$G(v,\omega) = \sigma[\alpha * head(v,\omega) + \beta * dist(v,\omega) + \gamma * velocity(v,\omega)] \quad (12)$$

The significance of the evaluation function is to find an optimal path so that the mobile robot can avoid obstacles in the constructed map and move towards the target at a faster speed in the local path planning navigation.

Only when the arrival position, the arrival direction, and the speed satisfy the three conditions of stopping, can the robot reach the target pose, that is, to complete the specified task. The effect diagram is shown in Fig. 6.

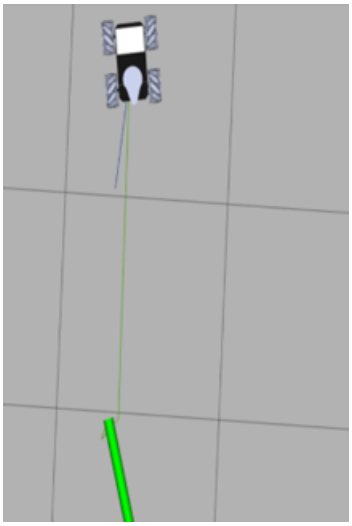


Fig. 6. Local path planning

The green line is the global planning route, and the blue short line is the local path planning route.

3.2. Global Path Planning Based on Optimized A*

This section describes the steps of A* algorithm and the improvement of A* algorithm.

3.2.1. The Detailed Steps of the A* Algorithm for Path Finding

The steps of the A* algorithm can be roughly divided into six steps, in which S (start) is the starting point and E (end) is the endpoint.

1) Starting with the starting point S, traverse the F value (up to 8) of the grids occupied by obstacles near S and include them in the open list.

2) Detecting whether there is a grid in the open list indicates that there is no path from the starting point S to the end point E if not.

3) Get the grid with the smallest F value from the open list and include it in the closed list.

4) The latest grid included in the closed list is selected as a new starting point to update the F value in the open list.

5) If the updated F value at this time has a smaller value than it is not updated, instead, select the grid with the F value in (3) to include in the closed list and repeat the operation in (4).

6) Check whether the latest grid included is ended E, if not repeated from step i, otherwise the shortest path from starting point S to end E has been found.

The comparison between the Dijkstra algorithm and the A* algorithm is shown in Fig. 7 and Fig. 8.

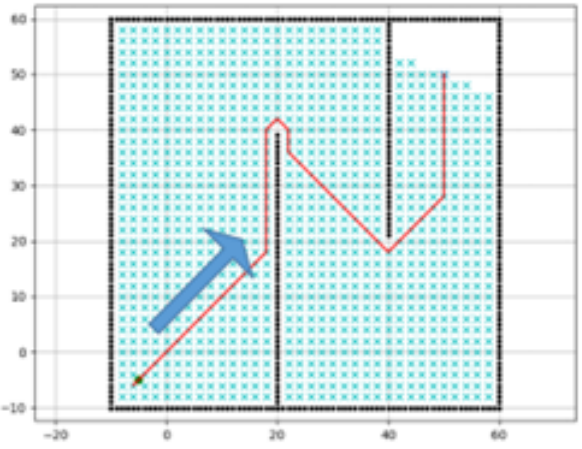


Fig. 7. Dijkstra algorithm

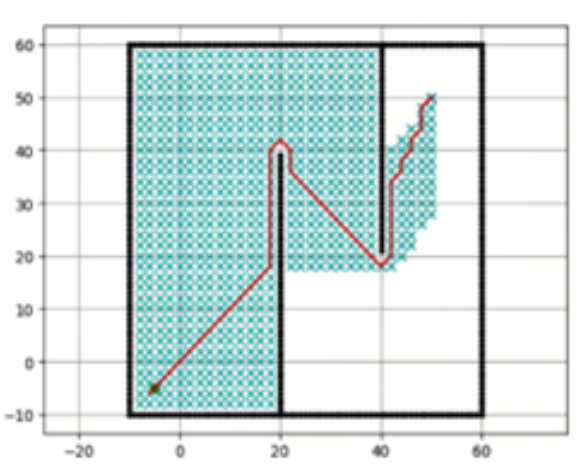


Fig. 8. A* algorithm

3.2.2. Improved A* Algorithm

1) Improvement of heuristic function

The evaluation function of A* algorithm is f(n)=g(n)+h(n), where h(n) is the heuristic function. The heuristic function can guide the correct extension direction of the path, and its improvement will have a great impact on the entire algorithm. Normally, in the grid map, the car only allows left and right up and down or diagonal movement, but cannot move at any angle. Based on this, the diagonal distance and Manhattan distance are selected as heuristic functions. Equ. 13 and Equ. 14 are the Manhattan distance and Chebyshev distance respectively.

$$h = |x_1 - x_2| + |y_1 - y_2| \quad \ldots\ldots\ldots (13)$$

$$h = max(|x_1 - x_2|, |y_1 - y_2|) \quad \ldots\ldots (14)$$

Fig. 9(Change only heuristic functions) shows the effect diagram for changing heuristic functions.

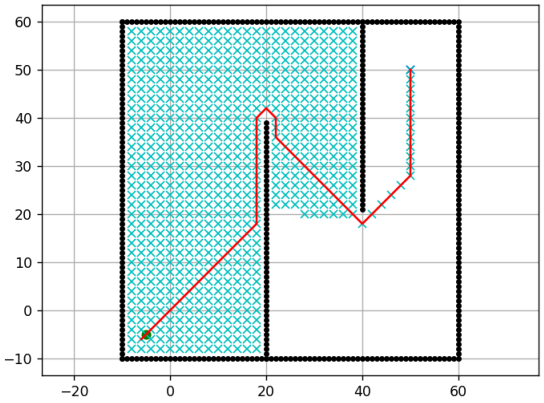


Fig. 9. A* algorithm after changing heuristic function

It can be seen from Fig. 9 that changing the heuristic function only reduces the grids included in the path planning at the end point, and basically includes all grids at the beginning, which greatly affects the efficiency of path planning. To change this situation, change the cost function f(n)=g(n)+h(n) to f(n)=g(n)+w * h(n). By giving the weight of the heuristic term to weaken the influence of g(n), the grid direction included tends to the end.

$$f(n) = g(n) + w * h(n) \quad \ldots\ldots\ldots (15)$$

When the weight coefficient w is large, the A algorithm will expand to the end point as soon as possible, and the search speed is fast but the optimal path will be missed. When w is small, A algorithm tends to search the optimal path and slow down the search speed. We can 't just consider the search speed without considering the planned path, that is, we can 't always let w = 3, and consider using dynamic weighting. In the case of abandoning the search optimal path, dynamic weighting is used to shorten the search time of A star. The principle is that it is more important to reach the destination area quickly at the beginning of the search. At the end of the search, getting the best path to the target is more important. That is, when h is large, the weight coefficient w should also be large. At this time, A algorithm will expand to the end point as soon as possible, and the search speed is fast but the optimal path will be missed. When h is small, w should also be small, where A algorithm tends to search the optimal path and slows down the search speed.

$$\begin{cases} h(n) = \begin{cases} 3 * d(n), & d(n) < 18 \\ 0.8 * d(n), & d(n) > 18 \end{cases} \\ f(n) = g(n) + h(n) \end{cases} \quad (16)$$

Fig. 10 shows the planned path given weight 3. Fig. 11 shows the A* algorithm after adding dynamic weight.

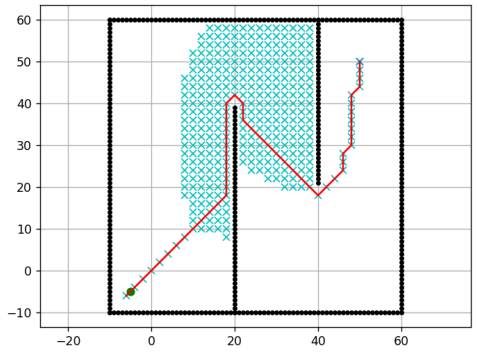


Fig. 10. A* algorithm after adding weight

2) Increasing different weights

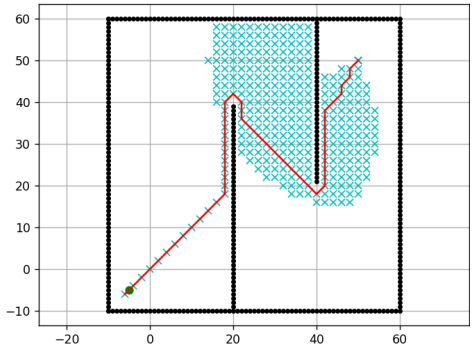


Fig. 11. A* algorithm after adding dynamic weight

3) Combining improved heuristics with dynamic weights

Combining the advantages of the improved heuristic term and the dynamic weight, the optimal path is planned while the number of grids is less in a short time. Fig. 12 shows the results.

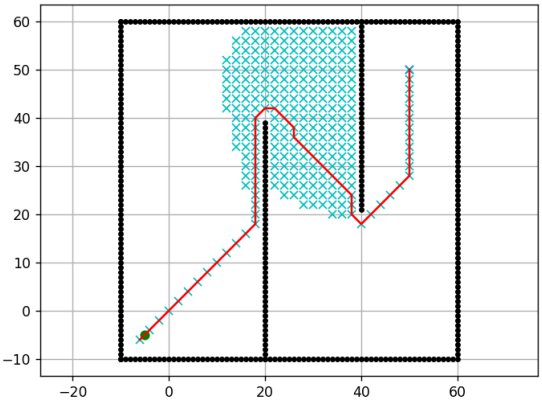


Fig. 12. A* algorithm after combining with the improved heuristic and dynamic weight

4) smoothing processing

In the path planned by the improved or unimproved A* algorithm, it can be seen that there are many bends or corners in the path. In order to make the path smoother, the third-order Bezier curve is introduced to smooth the path

$$P_i^k = (1-t) * P_i^{(k-1)} + t * P_{(i+1)}^{(k-1)} \quad . \quad . \quad . \quad . \quad (17)$$

Fig. 13 shows the smoothed path

4. Results

4.1. Mobile Robot Function Experiment Shows

In this design, C++ programming is used to realize the basic functions of car automatic driving, such as straight driving, reversing, ranging, speed measurement, turning, and obstacle avoidance. Fig. 14 shows part of the car information displayed for TFT.

Fig. 13. Smoothed A* algorithm

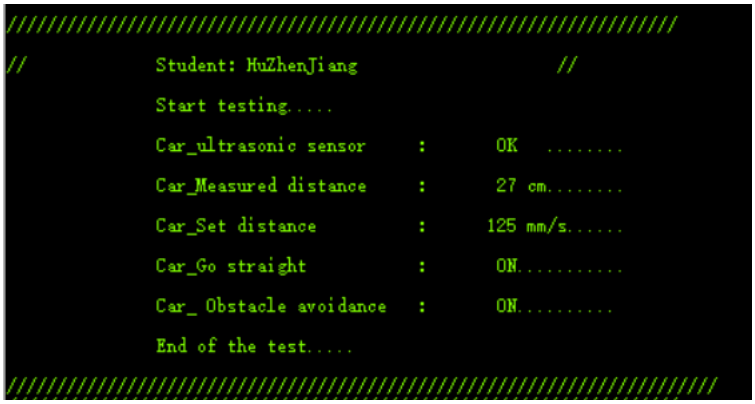


Fig. 14. Some information displayed in the process of car movement

4.2. Obstacle Avoidance and Path Planning for Mobile Robots in ROS

4.2.1. Robot Modeling in ROS System

The modeling steps of the mobile robot model are as follows:

1) The urdf model is established, and the basic structure of the car model is realized by a robot, link, and joint.

2) Call the function package provided by the Arbotix control board to control the motor and steering gear, accept instructions to update the action state of the joint at this time, and realize the movement of the mobile robot in the Rviz.

3) Based on the mobile robot model built in the first two steps, configure the radar model and start Gazebo to display radar information.

Fig. 15 shows the modeling of mobile robots in ROS environments



Fig. 15. Mobile robot model

4.2.2. Simulation Environment Construction Based on Gazebo

Due to the convenience of the ROS environment, most of the environments can be simulated, such as the most classical simulation of the indoor environment in sweeping robots, furniture, and wall used as obstacles. There are two ways to create a simulation environment in Gazebo:

1) Hand-painted simulation environment;

2) Manually add components.

Fig. 16 show the designed simulation environment.



Fig. 16. virtual map 1

4.2.3. Gmapping Mapping Results Display

Fig. 17 show that through the study of laser SLAM, the laser radar is used as the main sensor to realize the perception of the indoor environment, and the Gmapping algorithm is used to construct the grid map and save it into a yarm file.

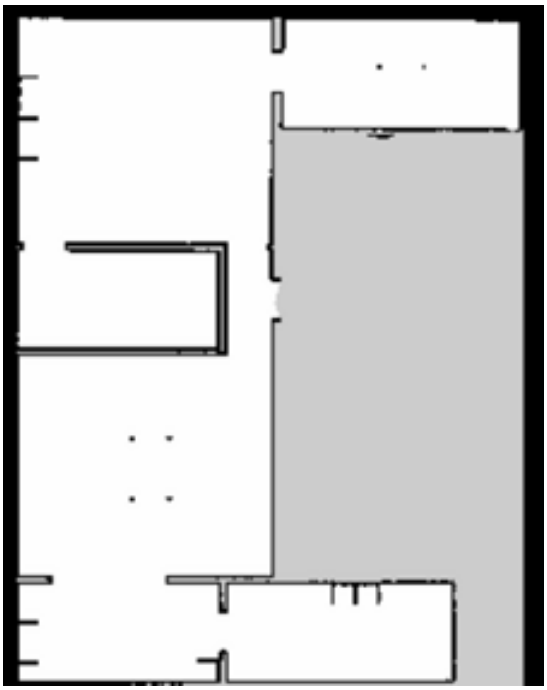


Fig. 17. Grid map built by virtual map 1

4.2.4. Path Planning Results Show Based on Optimized A* and DWA

In the constructed grid map, firstly, the position of the mobile robot at this moment is known in the gazebo map. Then, in the Rviz map, the position of the mobile robot needs to be initialized, and the '2D Pose Estimate' is selected. According to the initial position in the Gazebo map, the corresponding position in the Rviz is clicked to adjust its position and direction. After setting its initial approximate position, the '2D Nav Goal' is selected, that is, the endpoint you want the mobile robot to reach and the direction after reaching the endpoint. After running the launch file of the path planning algorithm, it can be seen that the mobile robot reaches the destination along the planned path, and the effect is shown in Fig. 18.

5. Conclusion

This paper combines global path planning based on the A* algorithm and local path planning based on the DWA algorithm. The mobile robot uses STM32 as the main control board, laser radar, and ultrasonic as the main sensors. The chassis is driven by two omnidirectional wheels, and a bull eye wheel is used as the support and steering

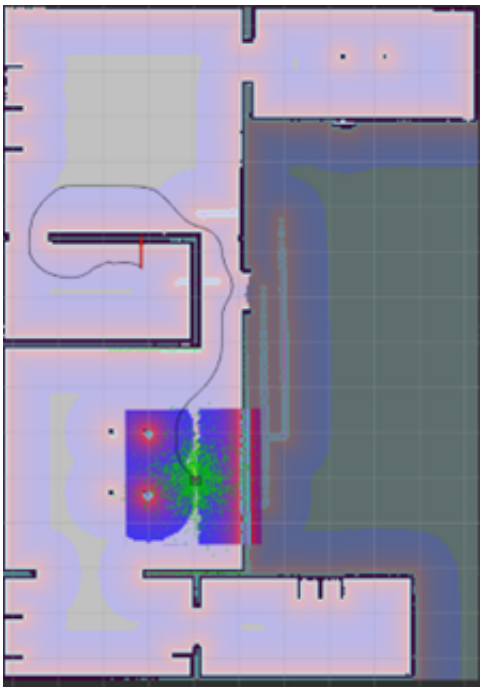


Fig. 18. Navigation effect map of mobile robot

control of the chassis. The deceleration encoder motor cooperates with the encoder and MPU6050 to control the motion of the car, and the position-based PID algorithm is used to stabilize the speed. In the ROS environment, the map construction and path planning are realized, and transmitted to STM32 through Bluetooth. STM32 controls the movement of the car accordingly. In addition, due to a lack of personal ability, and lack of time, there is still some design that is not completed or to be optimized:

1) In the indoor environment, there are often dynamic obstacles, and due to the sensor sensitivity and other issues, the planned path is not optimal.

2) In a completely unknown environment, because the sensor on the car has a great limitation on the sensing ability of the surrounding environment, UAV mapping can be used. However, due to the lack of physical objects and the lack of understanding of communication protocols, the hardware and software debugging and display of the system are not completed, resulting in the eventual failure to achieve.

Acknowledgements

This work was partially funded by the National Key Research and Development Plan of China (No. 2018AAA0101000) and the National Natural Science Foundation of China under grant 62076028.

References:

[1] Liu S, Zhang Q, Zhou D, Obstacle Avoidance Path Planning of Space Manipulator Based on Improved Artificial Potential Field Method[J], Journal of The Institution of Engineers (India): Series C,2014, 95(1):31-39.

[2] Kumar S., Mansuri S. B, Comparative analysis of path finding algorithms[J], IOSR Journal of Computer Engineering, 2018, 20(5): 38-45.

[3] Borenstein, J., Koren, Real-time obstacle avoidance for fast mobile robots[J], Systems, Man & Cybernetics, IEEE Transactions on.1989,19(5).1179-1187.

[4] Khatib O, Real-time obstacle avoidance for manipulators and mobile robots [J], The International Journal of Robotics Research,1986,5(1):90 -98.

[5] Bustince H, Campión M J, De Miguel L, Strong negations and restricted equivalence functions revisited: An analytical and topological approach[J], Fuzzy Sets and Systems, 2022, 441: 110-129.

[6] Liu J, Feng S, Ren J H, Directed D* algorithm for dynamic path planning of mobile robots[J], Journal of Zhe Jiang University (Engineering Science), 2020, 54(2): 291-300.

[7] Koenig S, Likhachev M, Fast replanning for navigation in unknown terrain[J], IEEE Transactions on Robotics and Automation, 2005, 21(3): 354-363.

[8] Oral T., Polat F, MOD* Lite: an incremental path planning algorithm taking care of multiple objectives[J], IEEE Transactions on Cybernetics, 2016, 46(1): 245-257.

[9] Jia H, Ma B, Research and improvement of RRT algorithm for mobile robot navigation[C], 2021 China Automation Congress (CAC). IEEE, 2021: 8354-8358.

[10] Noreen I, Khan A, Habib Z, Optimal path planning using RRT* based approaches: a survey and future directions[J], International Journal of Advanced Computer Science and Applications, 2016, 7(11).

[11] Gammell J D, Srinivasa S S, Barfoot T D, Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic[C], 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2014: 2997-3004.

[12] Ejm A, Nl B, Mo C, Dynamic window based approaches for avoiding obstacles in moving[J], Robotics and Autonomous Systems, 2019, 118:112-130.