Computing, Performance and Communication Systems (2023)
Clausius Scientific Press, Canada

DOI: 10.23977/cpcs.2023.070103
ISSN 2371-8889 Vol. 7 Num. 1

Research on the Navigation System of Indoor Supplies Delivery Robot Based on ROS

Hongkai Gu[a,*], Zhiwei Wang[b], Chenxi Ouyang[c]

School of Mechanical & Power Engineering, Harbin University of Science and Technology, Number 52 Xuefu Drive, Harbin, China
[a]1051184851@qq.com, [b]123806716@qq.com, [c]1298030863@qq.com
[*]Corresponding author

Keywords: ROS, Gmapping-SLAM, A*star algorithm, Dynamic window approach algorithm

Abstract: As the COVID-19 pandemic continues, people are realizing that efficient distribution of supplies can greatly improve work efficiency and reduce unnecessary conflicts, thus solving the problem of delivering goods to isolated locations. This paper studies the problem of robot autonomous map construction and path planning in the indoor corridor environment based on ROS. Gmapping-SLAM algorithm is selected as the robot mapping algorithm, A* algorithm is selected as the global path planning algorithm, and DWA algorithm is selected as the local path planning algorithm. Simulation experiments of autonomous map building and path planning are carried out. Therefore, the rationality of the above algorithm applied to indoor corridor environment is verified.

1. Introduction

As the COVID-19 pandemic continues to spread, problems related to the shortage of medical resources and substandard isolation conditions have emerged in various parts of the world. Some quarantine hotels could not timely and reasonably distribute epidemic prevention materials to quarantined people, and conflicts between doctors and patients occurred frequently. Unmanned distribution of materials is an effective way to solve the above problems. In this paper, by studying the navigation system of the unmanned car, the car can carry out the automatic distribution of materials in the complex building environment, and simulation, to verify the quasi swarm of the navigation system.

This paper mainly uses ROS system to simulate and control the delivery car. ROS is a secondary system that runs on Linux, connecting the Linux operating system to ROS applications. ROS provides a platform for the development of driverless cars, which effectively integrates various technical modules and provides a means of communication. Ros-based applications such as awareness module, navigation module, and underlying control module can easily establish communication. In addition, we can use the physical simulation platform Gazebo to build a two-dimensional map, and use Rviz to visualize the process of robot mapping and road stiffness planning.

2. The Study of SLAM Algorithm

SLAM is the abbreviation of Simultaneous Localization and Mapping, which includes two parts: simultaneous localization and map construction. Localization is the robot's determination of its position in the map. Map construction is the process of a robot using sensor and odometer information in ROS to build a raster map. The path planning module must be implemented via ROS on top of this. ROS topic communication mechanism is shown in the Figure 1:

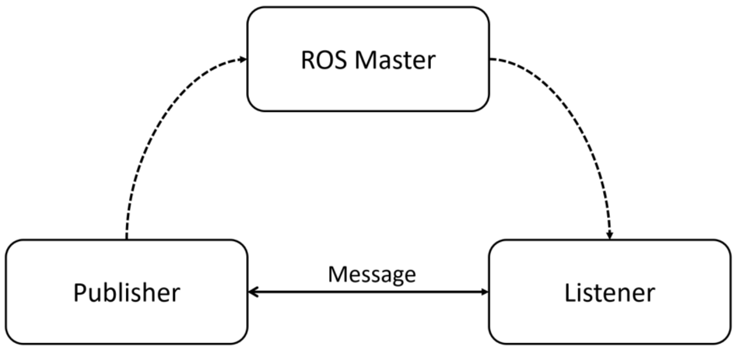

Figure 1: Topic Communication Process Diagram

Multiple open source SLAM algorithms are integrated in ROS, such as Karto SLAM, Cartographer, Gmapping, Hector SLAM, etc. In this paper, because Gmapping algorithm has the advantages of small computation, small computer burden, strong system robustness, etc., it is of high quality for drawing construction in narrow environment. Gmapping algorithm is the best choice for simulation experiment and research in the long and narrow hotel corridor environment studied by us.

2.1. Gmapping-SLAM Introduction

As one of the more commonly used and mature SLAM algorithms, Gmapping is mainly applicable to single beam lidar. Using RBPF algorithm, it is suitable for real-time construction of indoor environment map with high map accuracy. Gmapping creates two-dimensional raster maps based on mobile robot odometer data and lidar data.

2.2. Research on Gmapping-SLAM Algorithm

The Gmapping algorithm adopted in this paper is mainly based on RBPF algorithm. SLAM is divided into two parts: localization and mapping, but there is a contradiction between the two. In order to locate, a map must be established, and in order to determine the map, the robot also needs to locate itself. In order to solve this problem, RBPF gives a solution strategy, that is, location first and then see the map, separate location and map construction, and each particle carries a map, the specific decomposition process is as follows:

$$p(x_{1:t}|u_{1:t},z_{1:t}) = p(x_{1:t}|u_{1:t},z_{1:t}) \cdot p(m|x_{1:t},z_{1:t}) \tag{1}$$

$p(x_{1:t},m|u_{1:t},z_{1:t})$ is the joint probability density function of pose and map, $p(x_{1:t}|u_{1:t},z_{1:t})$ is the trajectory estimation of robot and $p(m|x_{1:t},z_{1:t})$ is the related calculation of map construction. Among them, it is easier to construct the image under known pose condition, and the robot trajectory estimation is the key problem.

In order to estimate the trajectory of the robot, RBPF adopts the idea of particle filtering and uses the importance resampling algorithm to repeatedly carry out the steps of sample-calculate weights-resampling, constantly iterating particle weights. Particles with high weights are gradually retained, while particles with low weights are constantly eliminated. Gmapping reduced particle number and particle degradation rate by improving proposed distribution and optimizing selective resampling strategy. In selective resampling, if the number of effective particles is less than the preset threshold, the resampling will continue[1]; otherwise, the resampling will stop and the estimated map will be obtained. The formula to determine the effective particle number is as follows:

$$N_{eff} = 1 / \sum_{i=1}^{N} (w^{(i)})^2 \tag{2}$$

$w^{(i)}$ represents the normalized weight. Through repeated sampling and particle iteration, the map is updated by calculating $p(m|x_{1:t}, z_{1:t})$ according to particle trajectory $x_{1:t}$ and observation information $z_{1:t}$. The specific algorithm flow chart is shown in Figure 2:

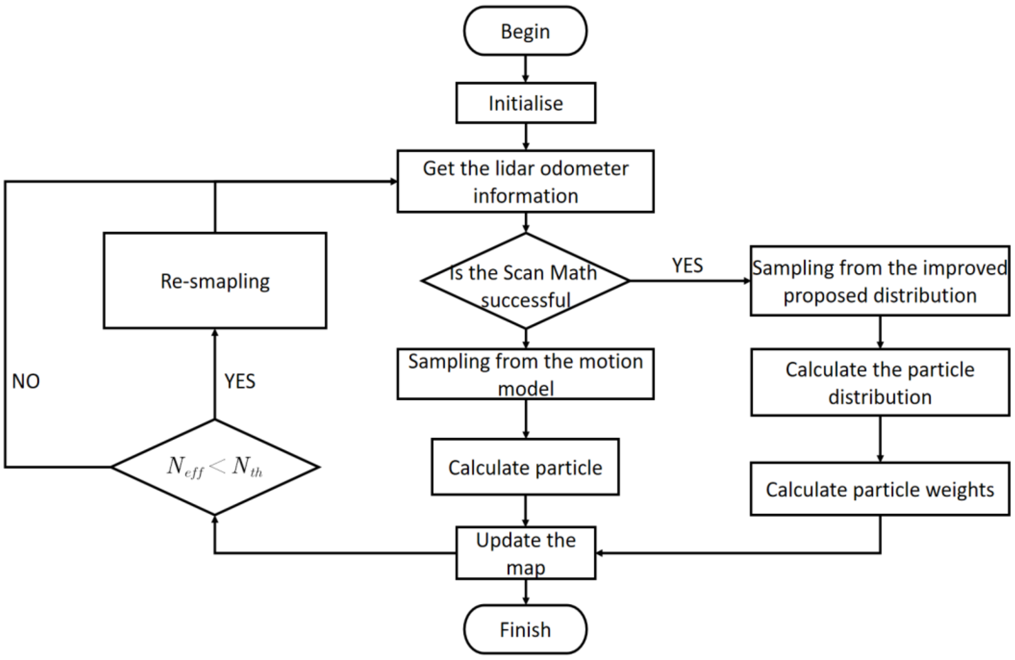
Begin
Initialise
Get the lidar odometer information
Is the Scan Math successful
YES
Sampling from the improved proposed distribution
Calculate the particle distribution
Calculate particle weights
Sampling from the motion model
Calculate particle
Update the map
Finish
Re-smapling
NO
YES
$N_{eff} < N_{th}$

Figure 2: Gmapping-SLAM Algorithm Flow Chart

The main disadvantage of RBPF algorithm is the high complexity caused by the large number of particles and the need for repeated sampling operations, which puts forward high requirements on computer computing power and memory. In addition, frequent resampling will also lead to particle degradation, which mainly includes two aspects: First, the interference of environmental repetition, observation noise and other factors will make the weight function of particles close to the correct state smaller, resulting in the possibility of correct but small weight particles being discarded. Second, the excessive frequency of resampling will cause the rate of decrease of particle diversity to increase.

Target distribution is to determine the maximum degree of bit confidence of the robot through the data collected by various sensors equipped with the robot. The robot cannot measure its own state directly, so it needs to collect various sensor data to estimate and control its own state. As shown in Figure 3, the dotted line represents the probability distribution of $p(x_t|x_{t-1}, u_t)$, which is also the Gaussian distribution of odometer sampling. The solid line represents the probability distribution of $p(z_t, x_t)$, which is also the Gaussian distribution of states obtained after laser observation. In terms of distribution, the variance of Lidar matching is much smaller than that of odometer model.[2] If the sampling range of particles is changed from the flat and wide region to the peak region represented by the lidar observation model, the new particle distribution can be more closely related to the target distribution.

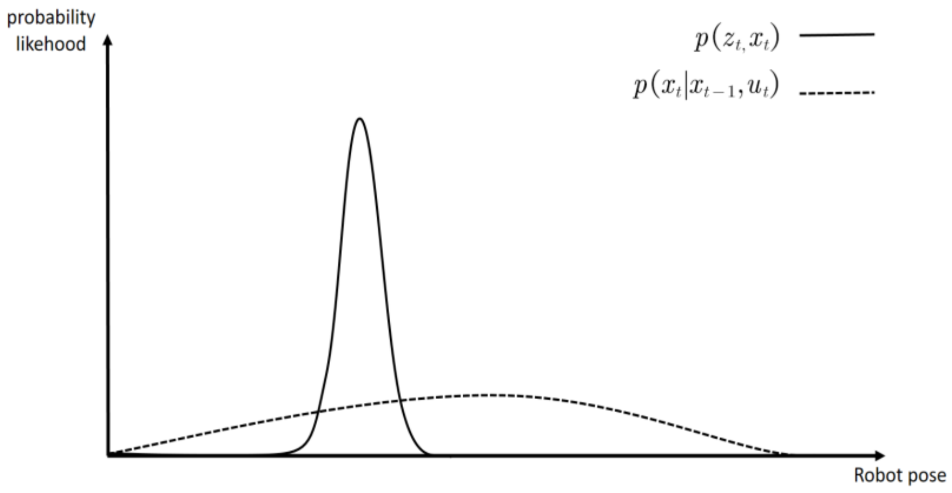

Figure 3: Probability Distribution in Fig

In order to improve the shortcomings of the above RBPF algorithm, Gmapping proposed a selective resampling scheme so as to reduce the resampling times and reduce the complexity of the algorithm. Gmapping gave up using odometer distribution as the proposed distribution, but integrated lidar motion information and observation information into the proposed distribution, which made the proposed distribution closer to the target distribution, reduced the number of samples, improved the utilization rate of computing resources, and was conducive to slowing down the degradation of particles. The improved probability distribution function is as follows:

$$\begin{aligned} q(x_t|x_{1:t}, z_{1:t}, u_{1:t}) &= p(x_t|m_{t-1}, x_{t-1}, z_t, u_t) \\ &= \frac{p(z_t|m_{t-1}, x_t) \cdot p(x_t|x_{t-1}, u_t)}{p(z_t|m_{t-1}, x_{t-1}, u_t)} \end{aligned} \quad (3)$$

In order to solve the problem of decreasing particle diversity, the selective resampling mechanism of Gmapping_slam ensures that resampling will not be carried out when the number of effective particles is greater than the threshold value, so as to improve particle diversity.

3. Research on Path Planning Algorithm

3.1. Introduction to path planning algorithms

After completing related operations of self-positioning and map construction in SLAM, we need to specify the starting point and end point for the robot, and plan a feasible and efficient traveling path for the robot. Path planning consists of global path planning and local path planning. Global path planning is mainly for the robot to plan a path with the shortest full distance. Local path planning is to plan a local path to realize the autonomous obstacle avoidance function of the robot in the case of unknown or partially known environmental information. In this paper, A* algorithm and DWA algorithm are the main research objects of global path planning algorithm and local path planning algorithm respectively.

3.2. Research on A* Algorithm

The A* algorithm is a common path planning and image traversal algorithm. Its search direction is determined by the estimation function. It has good accuracy and efficiency. Its working principle can be summarized as: firstly, it searches from the starting point in the way of detecting nearby nodes, then searches and detects the nodes near the nodes that have been traversed, and gradually diffuses outward until the search reaches the end point and stops traversing. The A* Algorithm is a heuristic search algorithm of global planning. In the search process, the distance between its own position and the end point is measured in real time, and heuristic rules are established, which can assist the intelligent car designed in this paper to carry out global path planning of unfamiliar environment, so as to realize the prediction and avoidance of obstacles.

The algorithm can usually give an approximate solution, but it is not guaranteed to be the best solution. Algorithm A firstly calculates the priority of each node through the estimation function. In the process of repeated operations, the node with the highest priority is selected from the priority queue of nodes each time, and then it is the next node to be traversed. In this way, the end point is found and the optimal trajectory is selected. The estimation function formula is as follows:

$$f(n) = g(n) + h(n) \tag{4}$$

In the formula: f(n) means the priority judged by node n at last. g(n) means the cost of moving from the starting point to node n. The meaning of h(n) is the expected cost function of the node moving to the end point, and it also serves as the heuristic function of the A* algorithm. The key of this algorithm is to select different heuristic functions. There are three heuristic algorithms: Manhattan distance, Chebyshev distance and Euclidean distance. It is expressed by two points (x1,y1) and (x2,y2) in the plane. The three formulas are as follows: [3]

Manhattan distance:

$$D = |x_1 - x_2| + |y_1 - y_2| \tag{5}$$

Chebyshev distance:

$$D = \max(|x_1 - x_2|,|y_1 - y_2|) \tag{6}$$

Euclidean distance:

$$D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{7}$$

In this paper, the Euclidean distance algorithm is used as the heuristic function to study the global path planning algorithm of robot. The specific steps of the algorithm are as follows:

(1) Initialize and create two lists, OPEN list and CLOSE list, and ensure that both lists are empty.

(2) Take the S node as the starting point, that is, put the S node into the OPEN list, then add the squares adjacent to S into the OPEN list, and then set the S node as their parent node. Finally, add the S node to the CLOSE list.

(3) Select another square reached by S node and repeat the search operation. The squares chosen are determined by the minimum value obtained by the estimation function $f(n) = g(n) + h(n)$.At the same time, it can be seen from the above that in h(n), Euclidean distance is chosen in this paper for calculation.

(4) Select the square with the smallest value and continue the search operation. In addition, if an obstacle is found, ignore the node where the obstacle is located.

(5) Keep repeating the above step (4). If the search reaches the end point, add the end point to the open list and carry out path planning. If the open list is empty at the end, meaning that no endpoint has been found, then the best path has not been found.

The flow chart of A* algorithm is shown in Figure 4:

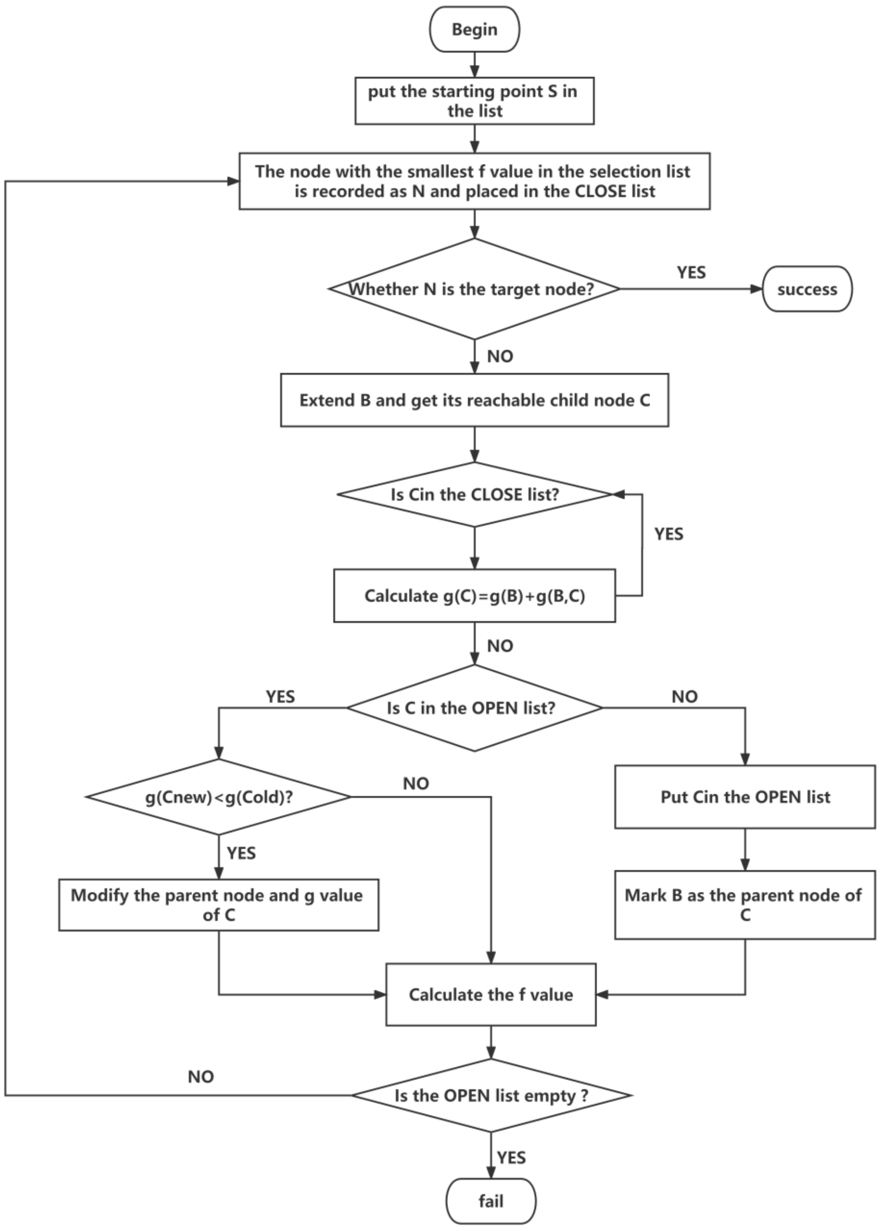
Begin
put the starting point S in the list
The node with the smallest f value in the selection list is recorded as N and placed in the CLOSE list
Whether N is the target node?
YES
success
NO
Extend B and get its reachable child node C
Is Cin the CLOSE list?
YES
Calculate g(C)=g(B)+g(B,C)
NO
Is C in the OPEN list?
YES
NO
g(Cnew)<g(Cold)?
NO
YES
Put Cin the OPEN list
Modify the parent node and g value of C
Mark B as the parent node of C
Calculate the f value
NO
Is the OPEN list empty ?
YES
fail

Figure 4: A* algorithm flow chart

3.3. Research on DWA algorithm

When the car travels along the path of global planning, it will also face some new obstacles or dynamic obstacles. At this time, it is necessary to use local path planning to further avoid obstacles, so as to complete the delivery task. The DWA algorithm is required for local path planning.

DWA algorithm is mainly based on the current position and motion status of the mobile robot, sampling multiple groups of velocities in the velocity space $V_s(v,w)$ composed of steering angular velocity and linear velocity, and then calculating and simulating the mobile robot's motion trajectories in the following period of time. After evaluating these trajectories, the optimal trajectories are selected and sent to the mobile robot for execution. DWA algorithm is mainly composed of three parts: velocity sampling, trajectory prediction and trajectory evaluation[4].

Due to the limitation conditions of the mobile robot, such as its own conditions and environmental factors, there are three constraints on the speed of the mobile robot, namely, the speed constraint, acceleration constraint and safety distance constraint, which jointly restrict the speed sampling space of the mobile robot. The first is speed constraint: due to the hardware conditions of the robot itself, there are upper and lower limits of linear velocity and angular velocity, which limits the velocity in the velocity space $V_m(v,w)$. Secondly, acceleration constraint: because the robot is affected by the motor, there is boundary angular acceleration and linear acceleration, so there is a dynamic window in the simulation period, and the velocity in the window is the actual velocity $V_d(v,w)$ that the robot can reach. Finally, the safe distance constraint: the local planning should carry out dynamic and real-time obstacle avoidance function, so for the surrounding obstacles, the robot needs to calculate a safe distance to ensure that it stops before encountering the obstacles, as shown in Figure 5. At this time, the speed constraint is in the velocity space $V_a(v,w)$. It should be noted that this restriction condition is not available at the beginning. The trajectory needs to be simulated. After the mobile robot travels for a period of time and finds the position of the obstacle, the distance between the trajectory and the obstacle can be calculated to determine whether the mobile robot can stop before hitting the obstacle at this speed.

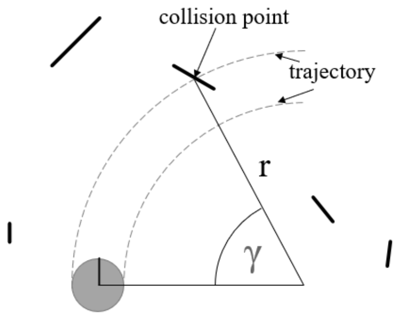

Figure 5: Predictive trajectory evaluation

The velocity space satisfying these three constraints simultaneously is the velocity sampling space Vs, which satisfies the formula:

$$V_s = V_m \cap V_d \cap V_a \tag{8}$$

After obtaining the velocity sampling space, the DWA algorithm needs to evenly sample the space with a certain resolution, discretized to obtain multiple groups of velocities (v,w), and then calculate the motion trajectory within the predicted time according to the kinematic model of the mobile robot to get the calculated trajectory. In the discrete process, the resolution E_v and E_w are set respectively for linear velocity and angular velocity, then the number of sampling velocity groups n is:

$$n = [(v_h - v_l)/E_v] \cdot [(w_h - w_l)/E_w] \tag{9}$$

In the formula: v_h, v_l, w_h and w_l are respectively the upper and lower limits of linear velocity and angular velocity in the velocity sampling space.

After obtaining multiple groups of predicted trajectories, some trajectories are still substandard, so it is necessary to select the trajectories obtained from sampling. The optimization is scored by the evaluation function, and the optimal trajectory is selected by comparing the scores, and then sent to the robot for execution. The evaluation function formula for trajectory evaluation is as follows:

$$G(v,w) = \sigma(\alpha \cdot heading(v,w) + \beta \cdot dist(v,w) + \gamma \cdot velocity(v,w)) \tag{10}$$

Where, $heading(v,w)$ is the azimuth evaluation function; $dist(v,w)$ is the distance evaluation function; $velocity(v,w)$ is the velocity evaluation function.

Since multiple sensors collect information and the evaluation criteria are different, the evaluation function will be discontinuous, which needs to be normalized. In the evaluation function, σ means normalization. The normalization formula is as follows:

$$\sigma \cdot heading(v,w) = normal\_heading = \frac{heading(i)}{\sum_{i=1}^{n} heading(i)} \tag{11}$$

$$\sigma \cdot dist(v,w) = normal\_dist = \frac{dist(i)}{\sum_{i=1}^{n} dist(i)} \tag{12}$$

$$\sigma \cdot velocity(v,w) = normal\_velocity = \frac{velocity(i)}{\sum_{i=1}^{n} velocity(i)} \tag{13}$$

Based on the above theory, a path can be obtained which can avoid dynamic obstacles and advance rapidly to the target point, so that the mobile robot can complete the local optimal planning. The flow chart of DWA algorithm is shown in Figure 6:

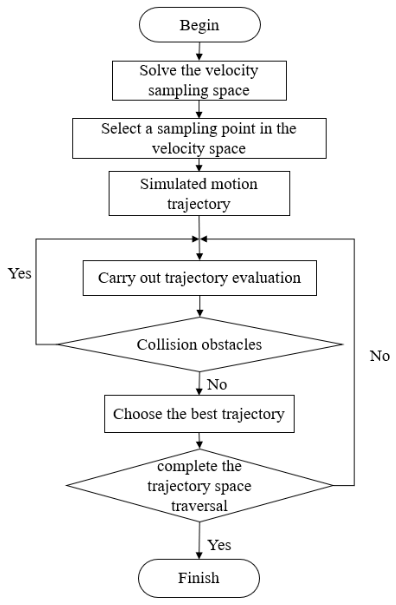
Begin
Solve the velocity sampling space
Select a sampling point in the velocity space
Simulated motion trajectory
Carry out trajectory evaluation
Yes
Collision obstacles
No
No
Choose the best trajectory
complete the trajectory space traversal
Yes
Finish

Figure 6: DWA Algorithm Flow Chart

4. Simulation Experiment

4.1. Build the Experimental Platform

In this experiment,we installed ROS Robot operating system on Ubuntu operating system. We choose a virtual machine for hardware simulation of Ubuntu, which is relatively simple to install and run, and has no impact on the host system windows. In this experiment, ROS has been installed on the PC and the version is noetic. Install Gazebo and Rviz in the Ubuntu operating system. Gazebo is a 3D dynamic simulator that allows operators to build 3D map simulation environments. Rviz is a simulation visualization platform that can visualize data. Gazebo and Rviz are often used in combination for autonomous mapping and routing.

4.2. Gmapping SLAM Experiment

In this experiment, Gmapping is used as the SLAM algorithm of the robot. Gmapping can draw two-dimensional raster map and determine its position and pose according to robot displacement data and laser radar data. The core node of the Gmapping software package is SLAM_Gmapping. The communication mechanism is shown in Figure 7:

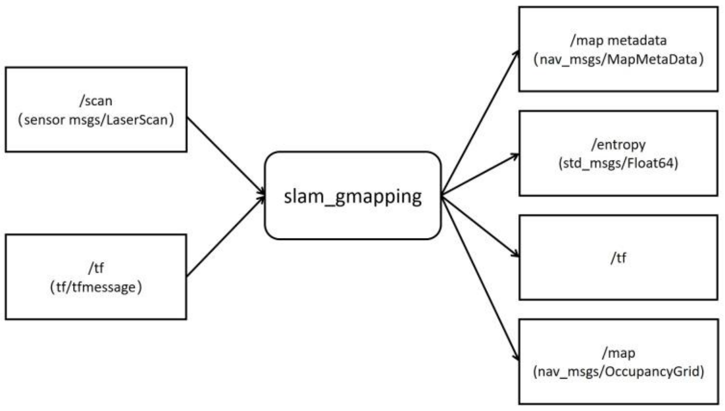

Figure 7: Gmapping SLAM Topic Chart

/slam_Gmapping node receives radar information of Lidar by subscribing topic /scan, and realizes transformation between radar coordinate system and chassis coordinate system by subscribing topic /tf. /slam_Gmapping also releases four topics: /map_metadata is periodically updated map metadata, including a series of basic map information; /entropy is the dispersion estimated by robot pose; /map is map raster data that is visually presented in Rviz; /tf realizes the transformation between the chassis and the odometer coordinate system.

By running relevant startup files, start each node, import the robot model into the simulation environment, and use the keyboard to control the movement of the robot, and complete the construction of the map in the process of robot movement. Figure 8 below shows the physical simulation environment built in gazbeo. Figure 9 shows the raster map constructed by the robot. By comparing the simulation map with the robot's mapping effect, it can be seen that in the corridor environment, the robot can build a relatively high precision map without obvious deviation. The Gmapping-SLAM algorithm is in line with the indoor food delivery environment, and can meet the experimental environment requirements in this paper.

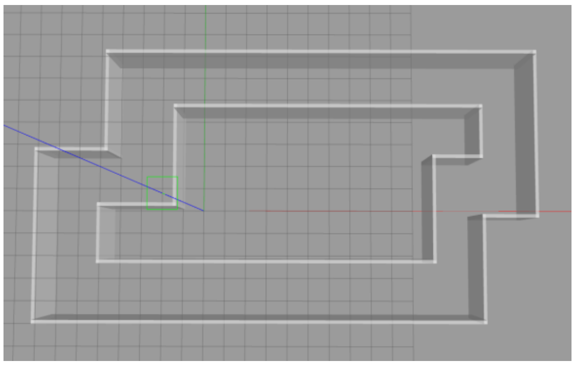
Figure 8: Simulation Environment

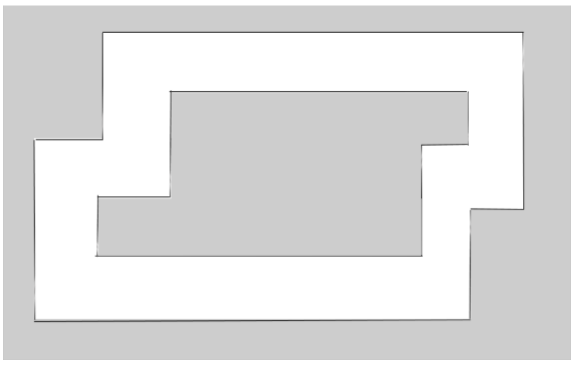
Figure 9: Gmapping Map

4.3. The Study of Path Planning

In this paper, the simulation experiment of path planning will be carried out through the move_base function package integrated by ROS. Based on the completion of the simulation map construction, the robot can complete its own positioning and path planning in the perceived map simulation environment. The move_base-based navigation package is shown in Figure 10:

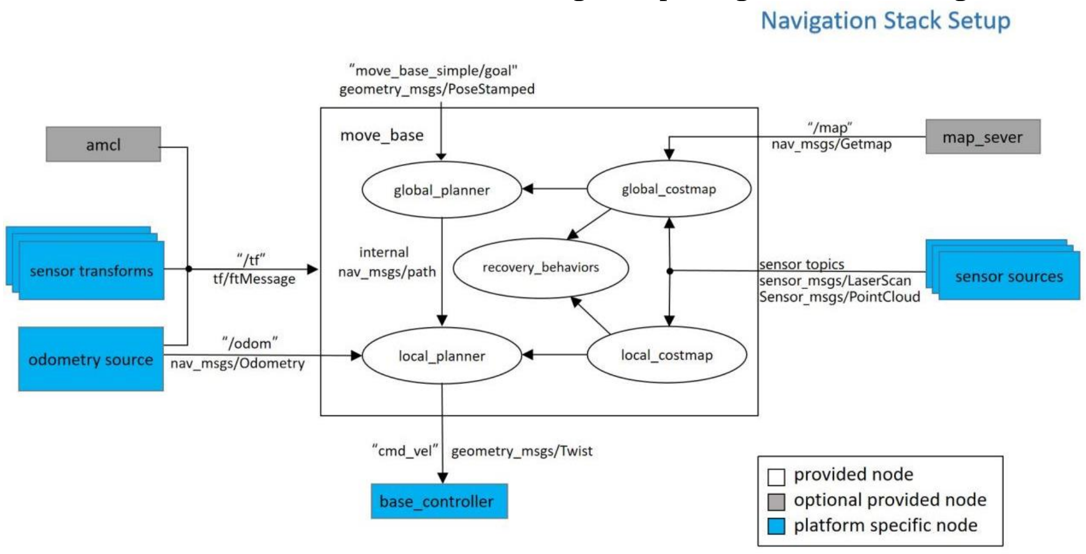

Figure 10: Navigation Frame Diagram

The move_base node is mainly composed of two planners, namely global path planner and local path planner. Meanwhile, there is also a built-in abnormal behavior handler in move_base, which is used to deal with the abnormal behavior of robots, as shown in Figure 11. This paper uses A* algorithm for global path planning and DWA algorithm for local path planning. In the path planning process of the robot, move_base obtains sensor information and senses the external environment by subscribing to the topic, thus completing the path planning. When the robot performs path planning in a known environment, map data is obtained through map_server; when the robot performs path planning in an unknown environment, real-time map construction is carried out through Gmapping_SLAM.

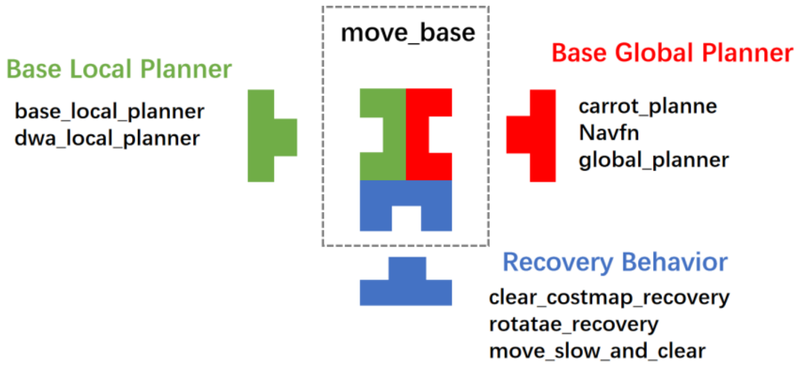

Figure 11: Move base Plugin

The robot model was imported into the physical simulation environment, and the global path and local path were viewed through the visualization plug-in Rviz. 2D pose Goal was used to set the robot target point pose. As shown in Figure 12, the green curve of robot operation represents the globally planned path. The red curve shown in Figure 13 is the local path of the robot.

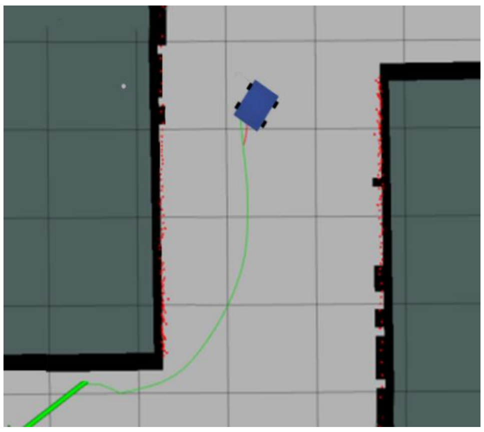
Figure 12: Global Path

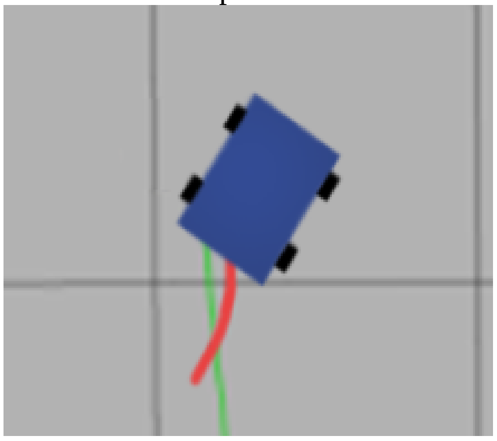
Figure 13: Local Path

When the robot moves in the perceived map, add temporary obstacles to the map, and the robot has the ability to identify unknown obstacles to achieve real-time obstacle avoidance, as shown in the Figure 14 and Figure 15:

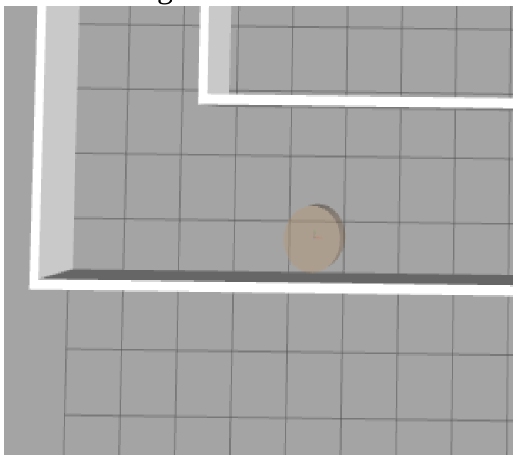
Figure 14: Temporary Obstacle

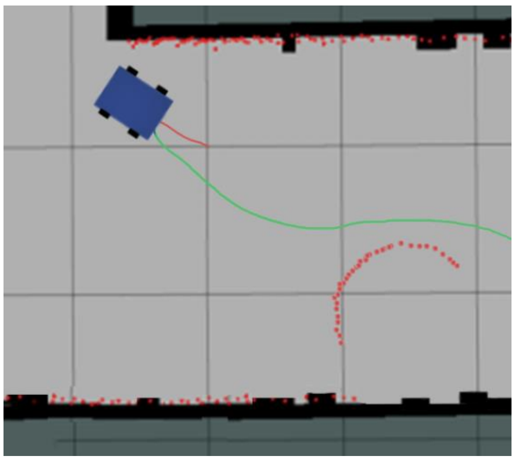
Figure 15: Autonomous obstacle avoidance

5. Conclusion

In this paper, the indoor corridor environment of the hotel is selected as the research environment. Firstly, the Gmapping_Slam algorithm based on the RBPF algorithm is described and the improvement of the RBPF algorithm is studied. In this paper, the global path planning algorithm A* algorithm and the local path planning algorithm DWA algorithm are studied through simulation experiments, and their principles and implementation process are analyzed and verified. In the simulation experiment stage, through the joint experiment of map building tool Gebezo and visualization tool Rviz, we can see that the above algorithm is suitable for map construction and robot path planning in the narrow corridor environment, which verifies the feasibility of the above algorithm.

References

[1] Di Wang, Guan Li, Teng Yu & Renshi Li.(2021).Research and simulation of mobile robot Navigation System based on ROS. Journal of North China Institute of Science and Technology (04), 54-60.
[2] Wenzhi Liu. (2018).Research and implementation of SLAM and path planning algorithm based on lidar [D].
[3] Dengxiang Chang. (2019).Research on autonomous navigation of driverless cars based on ROS (Master's Thesis, Hunan University)
[4] Haojie Wang, Xianghua Ma, Wanyu Dai & Wuxuan Jin. (2023).Research on Obstacle avoidance of mobile robot with improved DWA algorithm. Computer Engineering and Applications (06), 326-332.