Journal of Artificial Intelligence Practice (2022)
Clausius Scientific Press, Canada

DOI: 10.23977/jaip.2022.050106
ISSN 2371-8412 Vol. 5 Num. 1

Research on Autonomous Mapping and Path Planning of Indoor Food Delivery Robot Based on ROS

Juntao Li[a,*], Zhiwei Wang[b], Wenjie Yang[c], Hao Zhang[d]

School of Mechanical & Power Engineering, Harbin University of Science and Technology, number 52 Xuefu Drive, Harbin, China
a. LiJunTao913@163.com, b. wangzhiweihit@126.com, *c. 1264006805@qq.com,*
*d. 79598834@qq.com, *corresponding author*

Keywords: ROS, Gmapping-SlAM, A*star algorthim, Dynamic window approach algorthim.

Abstract: With the outbreak of the new crown epidemic, in order to better solve the meal delivery problem in isolation wards. In this paper, we investigate the autonomous map building and path planning of indoor meal delivery robot based on ROS. The Gmapping SLAM algorithm is selected as the map building algorithm of the meal delivery robot, while the A algorithm is selected as the global path planning algorithm of the robot, and the DWA algorithm is used to complete the autonomous obstacle avoidance of the robot, and the simulation experiments of autonomous map building and path planning are carried out separately, and the results of the experiment. show that the above algorithms can be applied to the autonomous map building and path planning of the meal deliver.

1. Introduction

With the rapid development of artificial intelligence and robots, there are more and more applications of autonomous mobile robots in logistics, transportation and other fields, but robot application environments are mainly indoors。At the same time, with the outbreak of the COVID – 19 pandemic, the number of close contacts has continued to rise, and the catering of the isolation ward has become a big problem, Not only does it require a lot of manpower, but there are also certain risks. At this time, the food delivery robot played a vital role. The realization of the function of the food delivery robot mainly lies in solving the problems of its autonomous positioning in the unknown floor, the establishment of the environment map and the path planning. This paper will focus on researching the method of mapping and path planning for food delivery robots, and conduct simulations.

ROS is an open source robot software framework. It adopts a distributed and low-coupling software architecture, so that each module can be designed and developed separately, and then different functional modules are combined together according to the corresponding requirements to complete the development of the entire robot project. In addition, ROS also provides a powerful physical simulation platform Gazebo and a visualization platform Rviz ,which is convenient for research on mapping and path planning

2. The Study of SLAM Algorithm

There are many kinds of open source SLAM algorithms integrated by ROS, such as Gmapping、Hector、Cartographer. In this paper, the Gmapping algorithm is selected for study and perform simulation experiments.

2.1. Gmapping-SLAM Introduction

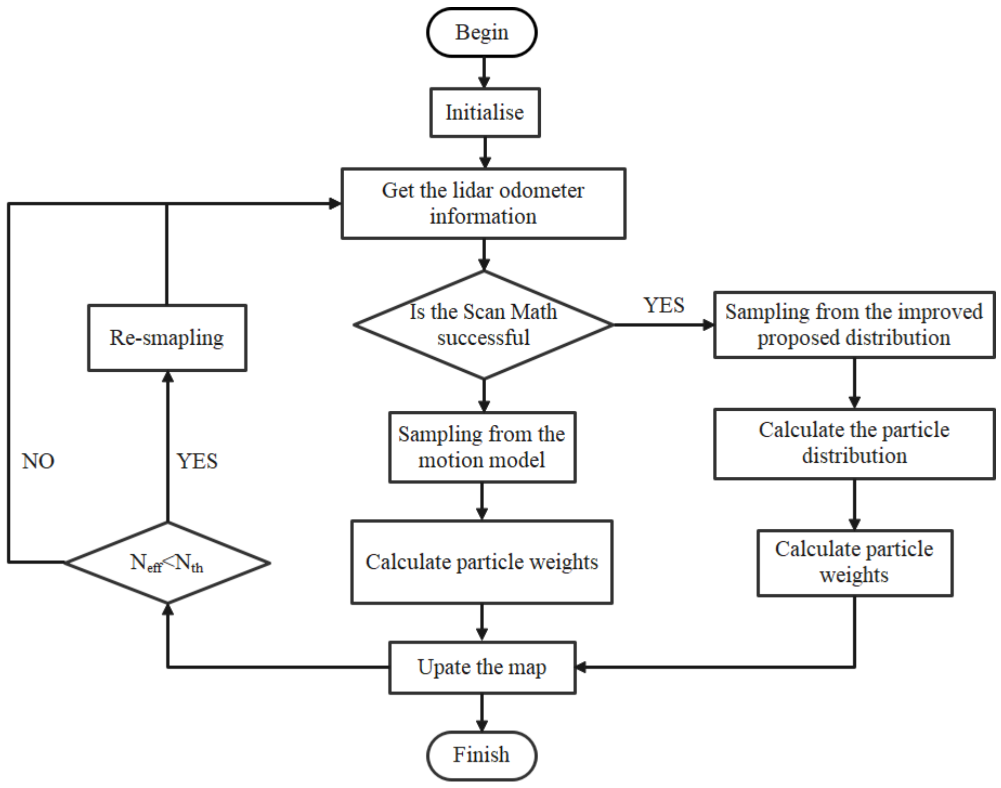
The Gmapping-SLAM is now the most widely used 2D laser SLAM algorithm, it is mainly based on the 2D lidar using the RBPF algorithm, which is suitable for the real-time construction of the indoor environment map, with high map accuracy and low requirements for lidar. It estimates the motion trajectory of the robot, then estimates the environment map based on the robot state, then the trajectory of the robot can also be updated through the environment map.

2.2. Research on Gmapping-SLAM Algorithm

Gmapping-SLAM is mainly based on the RBPF algorithm and has made two improvements. A complete SLAM problem is the problem of performing the simultaneous estimation of the map and the robot pose. However, in realistic situations, it is clearly contradictory that only matching the map can obtain exact pose, while only a good pose can get an accurate map. The SLAM algorithm based on RBPF can disassemble positioning and mapping into two processes, and each particle carries a map.The formula for the decomposition process is as follows:

$$\begin{aligned} p(x_{1:t}, m | u_{1:t}, z_{1:t}) &= p(x_{1:t} | u_{1:t}, z_{1:t}) \, p(m | x_{1:t}, u_{1:t}, z_{1:t}) \\ &= p(x_{1:t} | u_{1:t}, z_{1:t}) \, p(m | x_{1:t}, z_{1:t}) \end{aligned} \tag{1}$$

$p(x_{1:t}, m | u_{1:t}, z_{1:t})$ It is a joint probability density function, $p(x_{1:t} | u_{1:t}, z_{1:t})$ It represents the trajectory estimated by the robot. $p(m | x_{1:t}, z_{1:t})$ It represents the computations performed on the map construction. In order to obtain a more accurate distribution, the required number of samples is continuously increased, and each sample stores a map of the environment that matches its path. Due to the lack of loop closure detection in this method, a large number of samples are needed to complete the experiment ,which increases the consumption of memory and computing resources. Moreover, frequent resampling also causes particle degradation, which includes two main aspects: First, interference by factors such as environmental and observation noise may lower the particles weight function close to the correct state and the wrong state larger, and frequent resampling increases the likelihood that the correct particles are discarded. Second, the excessive frequency of resampling increases the rate of decreasing particle diversity. The RBPF-based Gmapping-SLAM algorithm reduces the number of particles and slows down the particle degradation speed by improving the proposed distribution and employing adaptive resampling methods[1][2]. The flow chart of the specific algorithm is as follows:



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

Figure 1: Gmapping-SLAM Algorithm Flow Chart.

The RBPF algorithm cannot sample directly from the target, so it samples from the importance probability density function according to the motion model to obtain samples conforming to the proposed distribution. The closer the proposed distribution is to the target distribution, the less the number of particles used, and the better the filter effect. The lidar obtains more accurate measurement information relative to the odometer, so the probability distribution covariance of the motion model is large. However, the observed model has a small probability distribution covariance[3]. The RBPF uses the motion model as an importance probability density function, and the sample needs to fill in the entire distribution of the motion model. However, due to the narrow particle sampling range of the observation model, fewer particles meet the target distribution, and the number of effective particles is small after resampling, which requires a large number of samples for sampling. As shown in the figure:

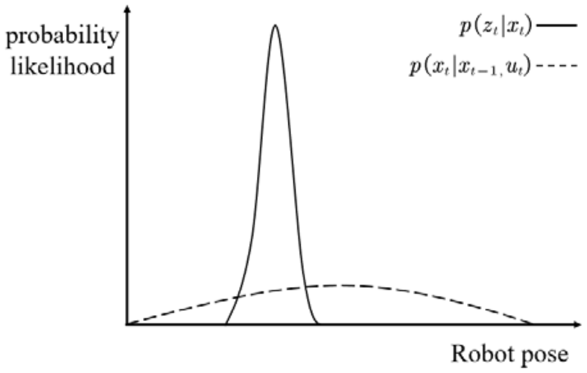


Figure 2: Probability Distribution In Fig.

For the above problems, the Gmapping-SLAM algorithm uses the prediction as the initial value to perform a scan matching after the odometer gives a prediction.If the matching is successful, improve the proposed distribution is selected for sampling, if the matching is unsuccessful, the motion model is selected to the proposed distribution. The motion information u_t and observation information z_t of lidar are fused in the improved proposal distribution, so that more samples gather within the scope of the observation model, which improves the accuracy of pose estimation, so as to obtain samples that are closer to the target distribution, and then reduce the number of samples, the utilization of computing resources is improved, and it is beneficial to slow down the degradation speed of particles. The improved important probability density distribution function is as follows.

$$\begin{aligned} q(x_t|x_{1:t}, z_{1:t}, u_{1:t}) &= p(x_t|m_{t-1}, x_{t-1}, z_t, u_t) \\ &= \frac{p(z_t|m_{t-1}, x_t)\, p(x_t|x_{t-1}, u_t)}{p(z_t|m_{t-1}, x_{t-1}, u_t)} \end{aligned} \tag{2}$$

At the same time, with the increase of resampling times, some particles are discarded, and no new particles are generated, which leads to a decrease in the diversity of particles in the space, For this problem, the Gmapping-SLAM algorithm introduces an adaptive resampling method and sets a threshold N_{th}, When the number of valid particles N_{eff} is less than the given threshold, If the number of valid particles is greater than the given threshold, it will not be performed resampling process.

3. The Study of Path Planning Algorithm

3.1. Path Planning Algorithm Introduction

After completing the positioning of the robot and the construction of the environment map, it is necessary to plan a feasible path for the robot according to the starting and ending positions of the robot. Path planning is divided into global path planning and local path planning. Global path planning refers to planning a shortest global path for the robot in a known environment. Local path planning is to plan a local path for the robot to achieve the function of autonomous obstacle avoidance when the environmental information is completely unknown or partially known. This paper will mainly study the A^* algorithm in the global path planning algorithm and the DWA algorithm in the local path planning algorithm.

3.2. Research on A* Algorithm

The A* algorithm is a heuristic search algorithm for local path planning, and its search direction is determined by the estimation function. Heuristic search is to establish heuristic search rules in the search process to measure the distance relationship between the real-time search position and the target position, so that the search direction is preferentially towards the direction of the target point, and improve the search efficiency. The expression of its estimation function is as follows:

$$f(n) = h(n) + g(n) \tag{3}$$

In the formula: f(n) is the cost value of node n from the initial node to the target node, h(n) is the heuristic function of the cost value consumed from any node n to the target node[4], g(n) refers to the actual cost function from the initial node to any node n. The key of this algorithm is to choose different heuristic functions. There are three heuristic functions: Manhattan distance, Chebyshev distance and Euclidean distance. The above three functions can be used for two points (x_1,y_1) and (x_2,y_2) in a two-dimensional environment.The specific expressions of the three functions are as follows:

Manhattan distance:

$$D=|x_1-x_2|+|y_1-y_2| \tag{4}$$

Chebyshev distance

$$D=\max(|x_1-x_2|,|y_1-y_2|) \tag{5}$$

Euclidean distance

$$D=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2} \tag{6}$$

In this paper, the algorithm with Euclidean distance as the heuristic function will be selected as the global path planning algorithm of the robot, and the simulation experiment of path planning will be carried out. The main idea of the algorithm is as follows：

(1) First create two lists, store the nodes to be explored in the OPEN list, and store the explored nodes in the CLOSE list. Initialize the two lists and add the starting node S to the OPEN list. At this time, the CLOSE list is still empty.

(2) Put the nodes around S into the OPEN list, and select the node with the smallest f(s) value in the OPEN list. Make the node the current node and add the node to the CLOSE list. Finally, delete the node in the OPEN list.

(3) Judge the current node. If it is the target node, the search is successful, return to the optimal path from the starting node to the target node and exit the search algorithm; otherwise, continue to execute the search algorithm.

(4) Extend the current node. Judge the child nodes around the current node. If it is in the CLOSE list, skip the child node. If the child node is not in the CLOSE set, execute the next judgement. If the child node is not in the OPEN list, add the child node to the OPEN list. Mark the current node as the parent node of the child node, and calculate the g(c) value and f(c) value from the starting node to the child node through the current node. If the child node is in the OPEN list, calculate the value of g(c_{new}) from the starting node through the current node to the child node, and compare g(c_{new}) and g(c_{old}). If the new g(c_{new}) value of the child node is less than the original g(c_{old}) value, mark the current node as the parent node of the child node, and replace the original g(c_{old}) value and f(c_{old}) value with the new g(c_{new}) value and f(c_{new}) value . If the new g(c_{new}) value of the child node is greater than the original g(c_{old}) value, the parent node of the child node remains unchanged.

(5) Judge the OPEN list. If the OPEN list is empty, the optimal path was not found; if OPEN is a non-empty list, continue to execute the search algorithm and repeat the steps above.

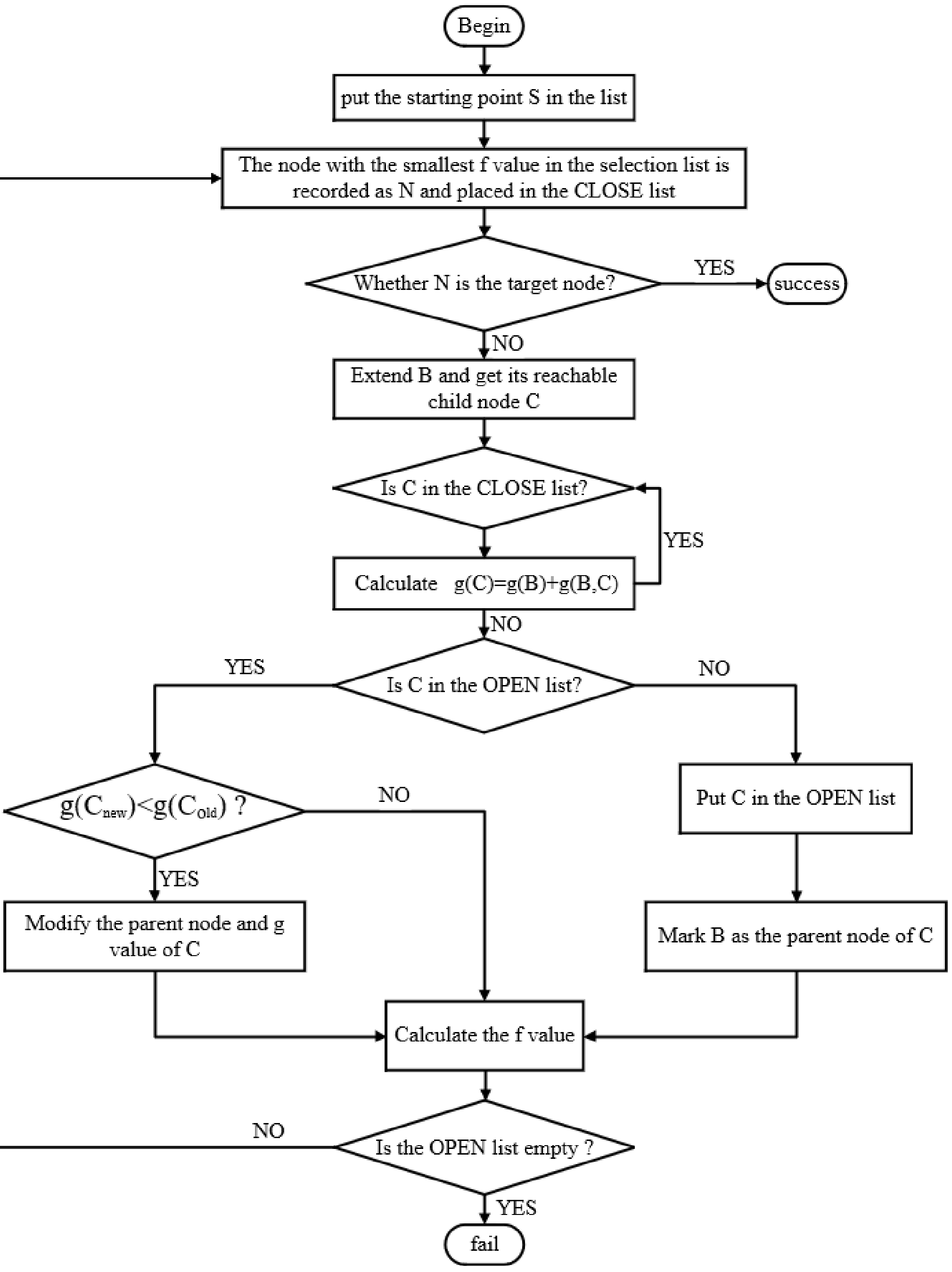

Begin
put the starting point S in the list
The node with the smallest f value in the selection list is recorded as N and placed in the CLOSE list
Whether N is the target node?
YES
success
NO
Extend B and get its reachable child node C
Is C in the CLOSE list?
YES
Calculate g(C)=g(B)+g(B,C)
NO
YES
Is C in the OPEN list?
NO
g(C_new)<g(C_old) ?
NO
Put C in the OPEN list
YES
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 3: A* algorithm flow chart.

3.3. Research on DWA Algorithm

When the robot travels along the global path in the actual environment, it will encounter some static or dynamic obstacles that are not in the known map. In order to realize the autonomous food delivery of robots, robots should be able to recognize obstacles in real time, and build a local planning path according to the actual environment, so as to achive real-time obstacle avoidance and complete the meal delivery task. The DWA algorithm combines dynamics and kinematics. First, it solves a control space composed of steering angular velocity and forward velocity, and discretizes the control space to obtain multiple sets of velocities, and then the motion trajectory of the robot in the simulation cycle is calculated according to the robot speed motion model. Finally, the trajectory is scored by the evaluation function, the trajectory with the highest score will be selected and send the velocity of that trajectory to the robot for execution. The algorithm flow chart is as follows:

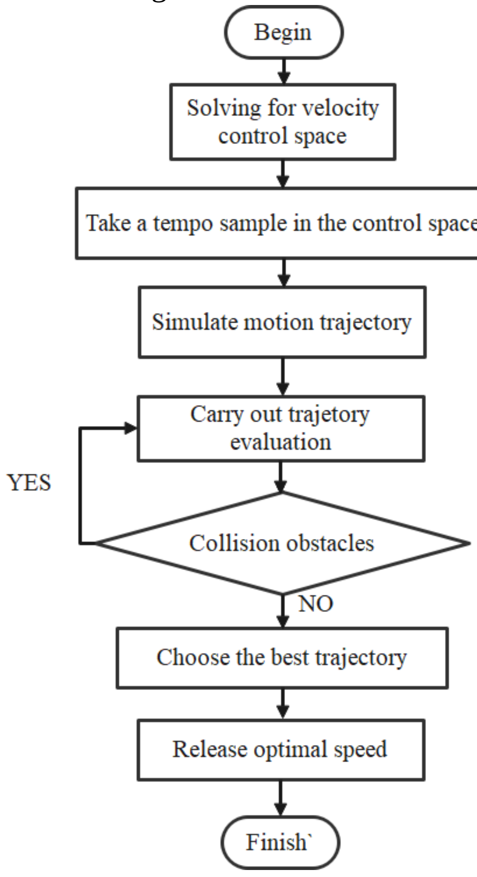

Begin
Solving for velocity control space
Take a tempo sample in the control space
Simulate motion trajectory
Carry out trajetory evaluation
YES
Collision obstacles
NO
Choose the best trajectory
Release optimal speed
Finish`

Figure 4: DWA Algorithm Flow Chart.

4. Simulation Experiment

4.1. Build Experimental Platform

First of all, the ROS robot operating system should be installed on the PC with the Ubuntu operating system installed. In this experiment, the PC has been installed with ROS, and the version is melodic. Then, install Gazebo and Rviz through the Ubuntu terminal. Gazebo is a platform for building a physical simulation environment in ROS, while Rviz is a visualization platform. In the simulation of autonomous mapping and path planning, gazebo and rviz are usually used in combination.

4.2. Gmapping -SLAM Experiment

The Gmapping-SLAM completes the estimation of its own pose and the construction of the environmental map by receiving the odometer information and lidar information. The Gmapping-SLAM completes the estimation of its own pose and the construction of the environmental map by receiving the odometer information and lidar information. The Gmapping topic information is as follows.

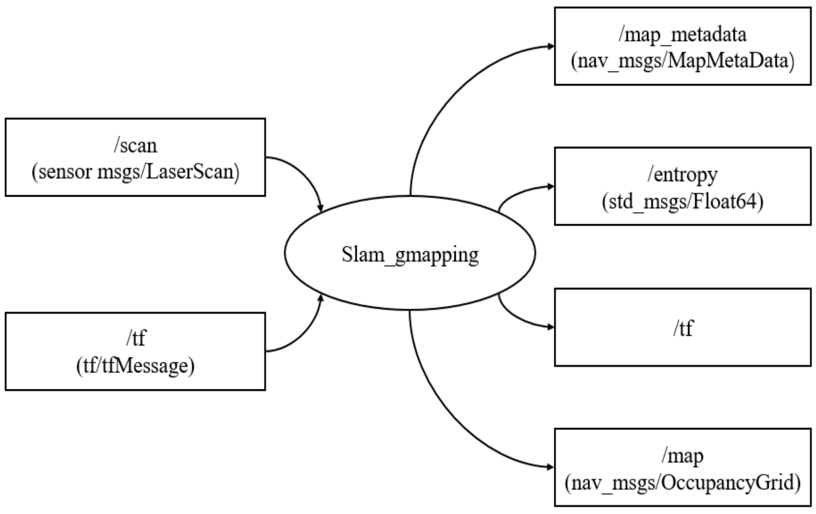


Figure 5: Gmapping SLAM Topic Chart.

The /Slam_gmapping topic receives the information of the lidar by subscribing to the /scan lidar topic, and subscribes to the /tf topic to realize the conversion between the robot coordinate system and the lidar coordinate system. At the same time ,/Slam_gmapping can publish four topics. /Map_MetaData is map-related information such as map size, map origin; /entroy is the degree of dispersion of robot pose estimation; /tf is the conversion between odometer coordinates and the global coordinate system; map is the established environment map.

Start the robot drive node, slam_gmapping node and each sensor node, and view the tf_tree, the specific coordinate system is as follows.

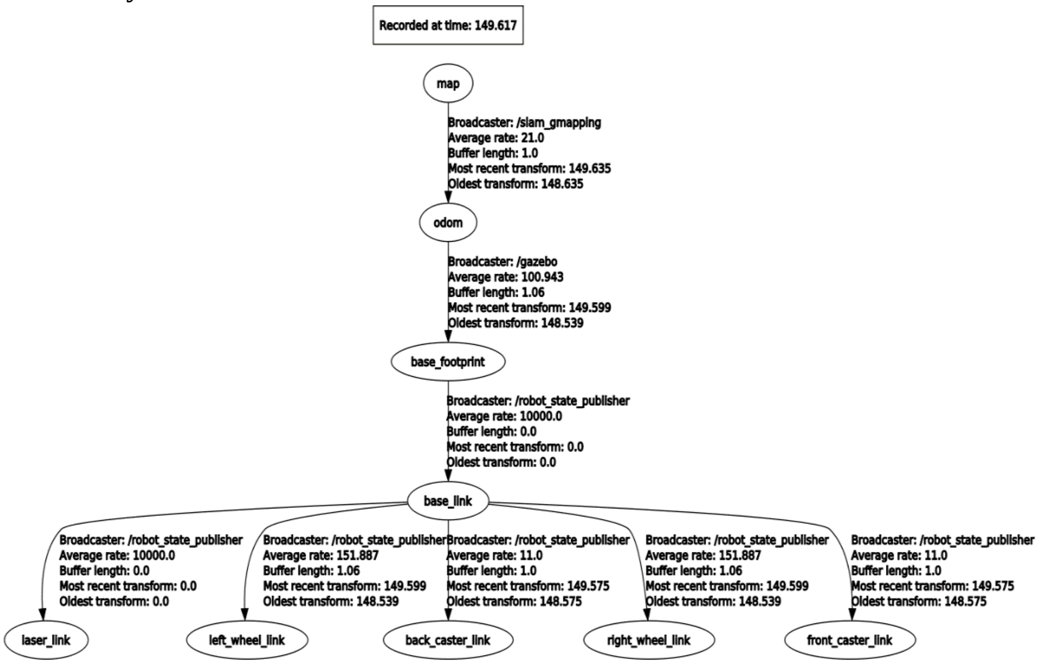

Recorded at time: 149.617
map
Broadcaster: /slam_gmapping
Average rate: 21.0
Buffer length: 1.0
Most recent transform: 149.635
Oldest transform: 148.635
odom
Broadcaster: /gazebo
Average rate: 100.943
Buffer length: 1.06
Most recent transform: 149.599
Oldest transform: 148.539
base_footprint
Broadcaster: /robot_state_publisher
Average rate: 10000.0
Buffer length: 0.0
Most recent transform: 0.0
Oldest transform: 0.0
base_link
Broadcaster: /robot_state_publisher
Average rate: 10000.0
Buffer length: 0.0
Most recent transform: 0.0
Oldest transform: 0.0
Broadcaster: /robot_state_publisher
Average rate: 151.887
Buffer length: 1.06
Most recent transform: 149.599
Oldest transform: 148.539
Broadcaster: /robot_state_publisher
Average rate: 11.0
Buffer length: 1.0
Most recent transform: 149.575
Oldest transform: 148.575
Broadcaster: /robot_state_publisher
Average rate: 151.887
Buffer length: 1.06
Most recent transform: 149.599
Oldest transform: 148.539
Broadcaster: /robot_state_publisher
Average rate: 11.0
Buffer length: 1.0
Most recent transform: 149.575
Oldest transform: 148.575
laser_link
left_wheel_link
back_caster_link
right_wheel_link
front_caster_link

Figure 6: Tf Tree.

By running the relevant launch file, starting each node, importing the robot model in the simulation environment, and using the keyboard to control the movement of the robot,the map construction is completed in the process of the robot movement. Fig.7 below is the physical simulation environment built in gazbeo. Fig.8 shows the grid map built by the robot. In the corridor environment, the robot can build a relatively high-precision map without obvious deviation. It can be seen that the Gmapping-SLAM algorithm is in line with the indoor food delivery environment and can meet the requirements of food delivery robot positioning and map construction.

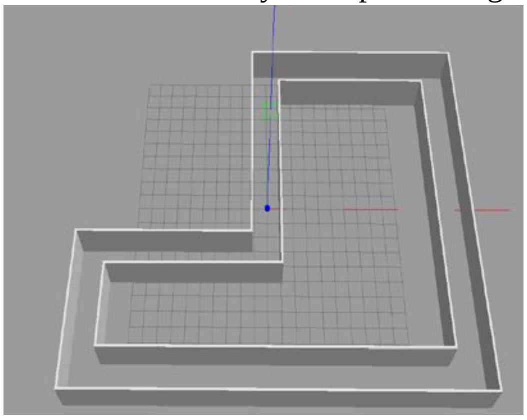


Figure 7: Simulation Environment.

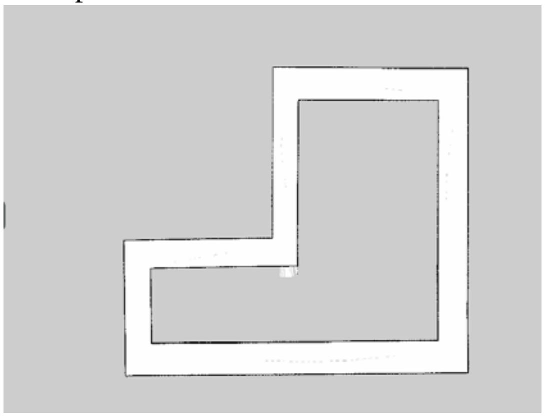


Figure 8: Gmapping Map.

4.3. Path Planning Experiment

In this paper, the simulation experiment of path planning will be carried out through the move_base function package integrated by ros. Make the robot complete its own positioning and path planning in a known simulation environment, or complete the map construction and its own positioning in an unknown map, and carry out path planning. The navigation frame based on move_base is as follows:

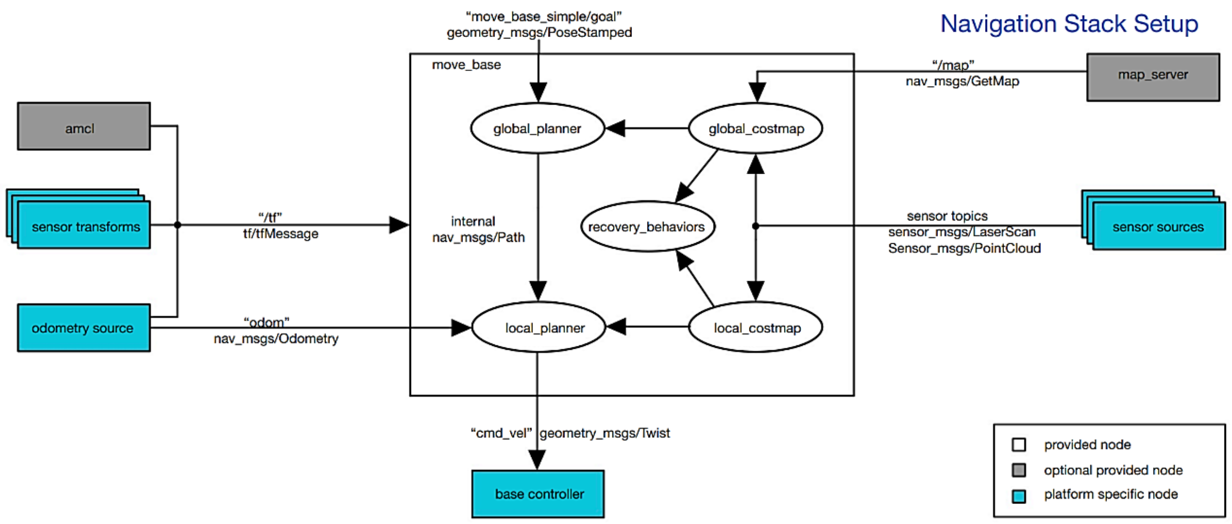


Figure 9: Navigation Frame Diagram.

The move_base node consists of three plugins, namely the global path planner, the local path planner and the abnormal behavior handler,as shown in Fig 10. In this paper, the global path planning is implemented by A^* algorithm, and the local path planning is implemented by DWA algorithm.In the process of robot path planning, move_base completes path planning by subscribing to topics.such as /map、/odom、/tf and /scan. When the robot performs path planning in a known environment, it obtains a map through map_server, and if it performs path planning in an unknown environment, it obtains a map through gmapping.

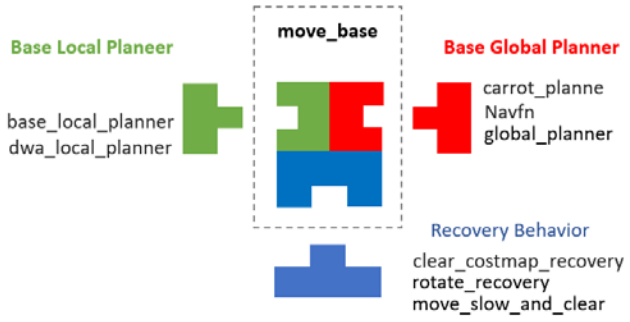


Figure 10: move_base Plugin.

Import the robot model into the physical simulation environment, and view the global path and local path through the visualization plugins rviz. Set the robot target point pose through 2D Pose Goal. As shown in Fig.11, the green curve that the robot travels is the global planned path, and the red curve shown in Fig.12 is the local path of the robot.

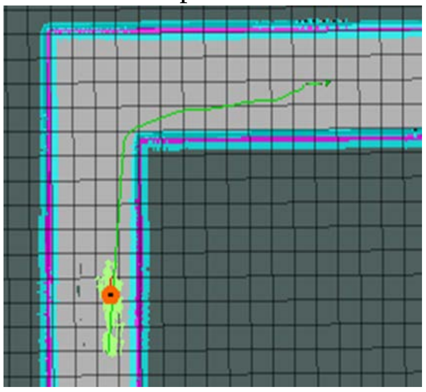


Figure 11: Global Path.

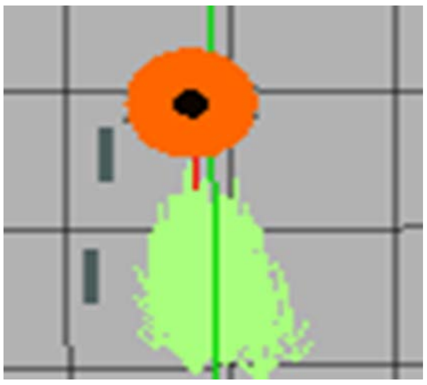


Figure 12: Local Path.

At the same time, add temporary obstacles in the gazebo, robot can recognize temporary obstacles and realize real-time obstacle avoidance, as shown in the figure below:

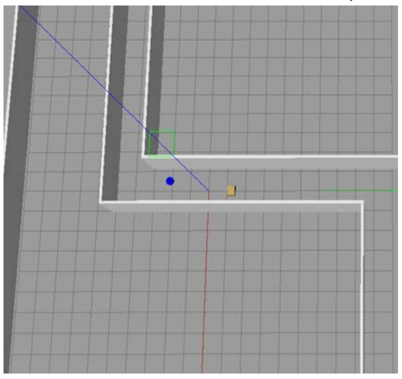


Figure 13: Temporary Obstacle

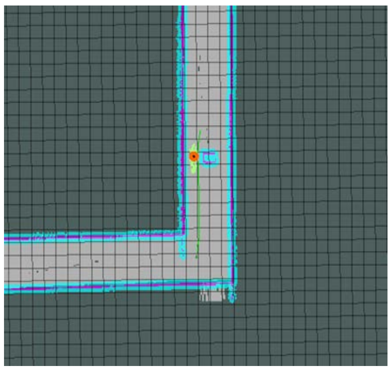


Figure 14: Autonomous obstacle avoidance

Robots should also have the ability to navigate within unknown maps, and be able to complete map construction and path planning in unknown environments. As shown in the figure below, by running the relevant launch file, the robot can complete autonomous navigation in an unknown map.

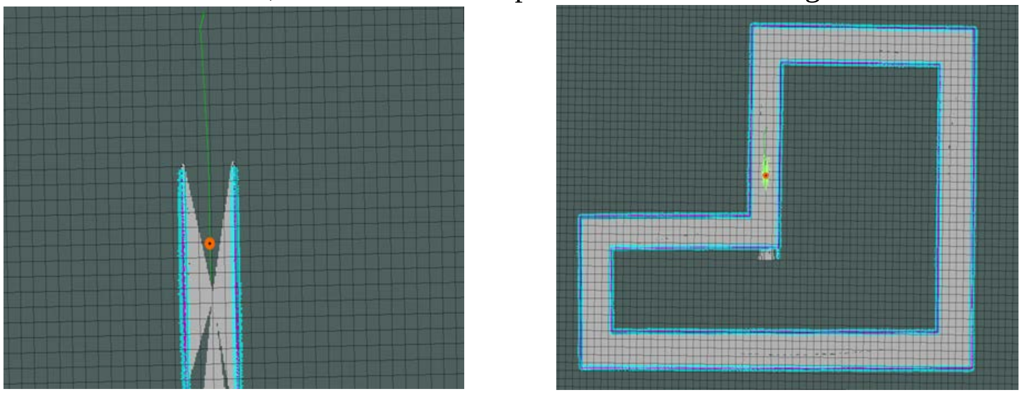


Figure 15: Autonomous Mapping and Path Planning for Unknown Environments.

5. Conclusion

First of all, this paper analyzes the Gmapping-Slam algorithm and studies its two improvements bas ed on the RBPF algorithm. Then the global path planning algorithm A algorithm and the local path planning algorithm DWA algorithm are explained, and the principles of these two algorithms are analyzed. Finally, the physical simulation environment was built through gazebo, combined with the visualization tool rviz, and simulation experiments were carried out. The simulation results sufficiently demonstrate the feasibility of the above algorithm and meet the needs of autonomous mapping and path planning for food delivery robots.

References

[1] Grisetti G, Stachniss C, Burgard W . Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling[C]//IEEE International Conference on Robotics & Automation. IEEE, 2005.

[2] Grisetti G, Stachniss C, Burgard W . Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters[J]. IEEE Transactions on Robotics, 2007, 23(1):34-46.

[3] Wenzhi Liu. Research and implementation of SLAM and path planning algorithm based on lidar [D].

[4] Cantao Zheng. Research on autonomous navigation and and pedestrian tracking of indoor mobile robot based on liadr [D].