Hindawi
Journal of Advanced Transportation
Volume 2020, Article ID 8867937, 14 pages
https://doi.org/10.1155/2020/8867937

Research Article

2D Lidar-Based SLAM and Path Planning for Indoor Rescue Using Mobile Robots

Xuexi Zhang[1], **Jiajun Lai**[1], **Dongliang Xu**[1], **Huaijun Li**[2] and **Minyue Fu**[3]

[1]*School of Automation, Guangdong University of Technology, and Guangdong Key Laboratory of IoT Information Technology, Guangzhou 510006, China*
[2]*School of Automobile and Engineering Machinery, Guangdong Communication Polytechnic, No. 789, Tianyuan Road, Tianhe District, Guangzhou 510630, China*
[3]*School of Electrical Engineering and Computer Science, The University of Newcastle, University Drive, Callaghan, 2308 NSW, Australia*

Correspondence should be addressed to Huaijun Li; lhj@gdcp.cn

Received 23 June 2020; Revised 29 July 2020; Accepted 21 October 2020; Published 17 November 2020

Academic Editor: Zhiguang Cao

As the basic system of the rescue robot, the SLAM system largely determines whether the rescue robot can complete the rescue mission. Although the current 2D Lidar-based SLAM algorithm, including its application in indoor rescue environment, has achieved much success, the evaluation of SLAM algorithms combined with path planning for indoor rescue has rarely been studied. This paper studies mapping and path planning for mobile robots in an indoor rescue environment. Combined with path planning algorithm, this paper analyzes the applicability of three SLAM algorithms (GMapping algorithm, Hector-SLAM algorithm, and Cartographer algorithm) in indoor rescue environment. Real-time path planning is studied to test the mapping results. To balance path optimality and obstacle avoidance, A^* algorithm is used for global path planning, and DWA algorithm is adopted for local path planning. Experimental results validate the SLAM and path planning algorithms in simulated, emulated, and competition rescue environments, respectively. Finally, the results of this paper may facilitate researchers quickly and clearly selecting appropriate algorithms to build SLAM systems according to their own demands.

1. Introduction

Mobile robots are capable of moving around in their environment and carrying out intelligent activities autonomously, thus having extensive realistic applications, including rescue works. A key enabling technology is simultaneous localization and mapping (SLAM) which allows the robot to estimate its own position using onboard sensors and construct a map of the environment at the same time. With the SLAM technology, real-time path planning can be performed to fulfill complex manoeuvring tasks in rescue works.

SLAM-enabled mobile robots have achieved much success in various scenarios. Peng et al. [1] studied the positioning problem and implementation of SLAM for mobile robots with RGB-D cameras. Shou et al. [2] employed a Raspberry Pi module as the core controller and built a mobile robot for map construction and navigation in indoor environment. Zhang et al. [3] proposed path prediction planning based on the artificial potential field to improve obstacle avoidance. Liu et al. [4] combined the Q-learning algorithm with the deep learning algorithm for path planning, which enabled robots to make reasonable walking paths under complex environmental conditions. Yu et al. [5] applied an improved A^* path planning algorithm to unmanned underwater survey ships, enabling quick obstacle avoidance and return to the preset route. However, these studies did not take into account the impact of the rescue environment on the SLAM algorithm. If these algorithms are directly applied to rescue robots, it may deteriorate the accuracy of path planning and even cause incorrect path planning results. At present, there are still rare systems that

can combine SLAM and path planning for indoor rescue. Therefore, it is necessary to study the impact of the rescue environment on the SLAM algorithm and path planning algorithm and evaluate and select the SLAM algorithm suitable for the rescue environment.

In this paper, we evaluated the results of some commonly used SLAM algorithms in both simulation and real-world environment, tested the path planning algorithms (A^* algorithm and DWA algorithm), and conducted a combined experiment of mapping and path planning regarding the RoboCup competition. These experiments revealed the demerits of some algorithms and provided a benchmark for subsequent algorithm improvement.

The rest of the paper is organized as follows. Section 2 introduces the basic system structure of mobile robots; Section 3 presents the rationale of three commonly used SLAM algorithms; Section 4 briefly describes the A^* algorithm and DWA algorithm for path planning; Section 5 provides and analyzes the experimental and simulation results; Section 6 gives the conclusion.

2. System Structure

The hardware part of the robot studied in this paper is mainly composed of motion control module, Lidar module, vision module, power module, and industrial computer module. The system structure is shown in Figure 1, and the physical map of the robot system is shown in Figure 2.

The main function of the STM32 microcontroller is to acquire and process wheel encoder data and gyroscope data. Map information, path planning, depth camera, and Lidar data are processed by the industrial computer. The industrial computer and STM32 are connected via USB cable to exchange data and instructions. The depth camera is calibrated by using a printed black and white checkerboard. The OpenCV function called by the robot operating system (ROS) is used to extract the corner information from camera images, and then internal and external parameters are obtained through calculations [6]. The industrial computer is fitted with Intel Core i5 processor, 4G memory, 128G access space, and the ubuntu16.04 system.

3. SLAM Algorithms

For a mobile robot, SLAM involves both localization and mapping in an iterative manner by continuously fusing various measurements from the onboard sensors [7, 8]. The sensor module in our system includes Lidar and depth camera to collect environmental information as well as internal measurements from the IMU. A 2D map is to be generated after processing by a mapping algorithm. Depending on the purpose of the map, different SLAM algorithms are available. For our purpose, we will focus on the task of path planning in real time. After various considerations, we decide to study in detail three most suitable SLAM algorithms: GMapping algorithm, Hector-SLAM algorithm, and Cartographer algorithm. GMapping algorithm is based on particle filter pairing algorithm, Hector-SLAM is based on scan matching algorithm, Cartographer is a scan matching algorithm with loop detection, and RGB-D algorithm is an algorithm for mapping using depth images. These several algorithms are representative and widely used algorithms.

3.1. GMapping Algorithm. The GMapping algorithm is a laser-based SLAM algorithm for grid mapping [9, 10]. This is probably the most used SLAM algorithm, currently the standard algorithm on the PR2 (a very popular mobile manipulation platform) with implementation available on *openslam.org*. The algorithm was initially proposed in [10], and the main idea is to use Rao–Blackwellized particle filters (RBPFs) to predict the state transition function. The algorithm is also known as the RBPF SLAM algorithm, named after the use of Rao–Blackwellized particle filters. In [11], two major improvements were made by optimizing the proposal distributions and introducing adaptive resampling, making the algorithm much more suitable for practical applications. It is then dubbed GMapping (G for grid) due to the use of grid maps.

3.1.1. RBPF. Onboard measurements include sensor data from Lidar or camera for images and odometer data from the IMU. A large number of particles are used for state transition function predictions, with each particle representing a possible position of the robot.

The sensor data are denoted by $(z_{1:t} = z_1, z_2, \ldots, z_t)$ and the odometer data by $(u_{1:t} = u_1, u_2, \ldots, u_{t-1})$ for the time period from 1 to t. They are used to estimate the joint posterior probability $p(x_{1:t}, m|z_{1:t}, u_{1:t-1})$ of the robot pose $(x_{1:t} = x_1, x_2, \ldots, x_t)$ and the grip map of the environment, represented by m. Using the Bayes' rule, the posterior probability can be decomposed into

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

where $p(x_{1:t}|z_{1:t}, u_{1:t-1})$ is the positioning problem, whereas $p(m|x_{1:t}, z_{1:t})$ is the mapping problem. The so-called importance sampling is used in the RBPF. The procedure is as follows:

(i) Sampling: according to the given (previous) proposal distribution, particles $(x_{t-1}^{(i)})$ from the previous generation are sampled. They are then improved by incorporating the most recent observations. Then, new particles $(x_t^{(i)})$ and proposal distributions are generated.

(ii) Weights: the weight $w_t^{(i)}$ of each current particle $x_t^{(i)}$ is calculated using

$$w_t^{(t)} = \frac{p\left(x_{1:t}^{(i)}|z_{1:t}, u_{1:t}\right)}{\pi\left(x_{1:t}^{(i)}|z_{1:t}, u_{1:t-1}\right)}, \tag{2}$$

where $\pi(\cdot)$ is the proposal distribution, which usually is a probabilistic odometry motion model.

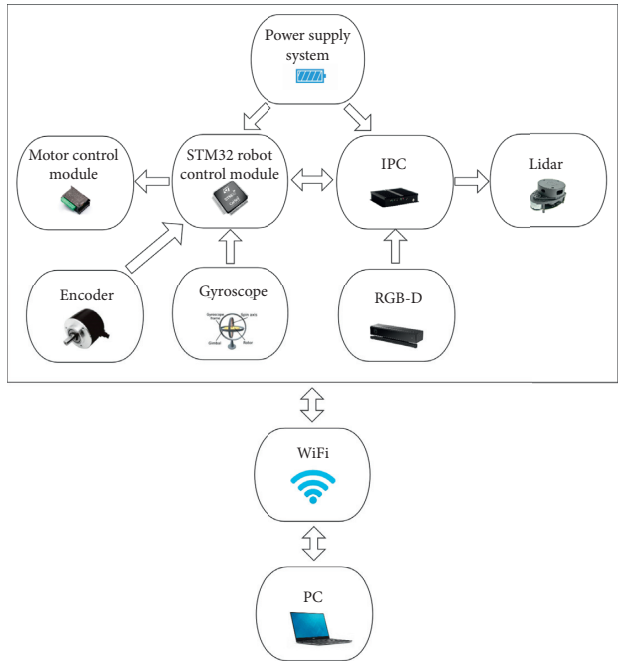
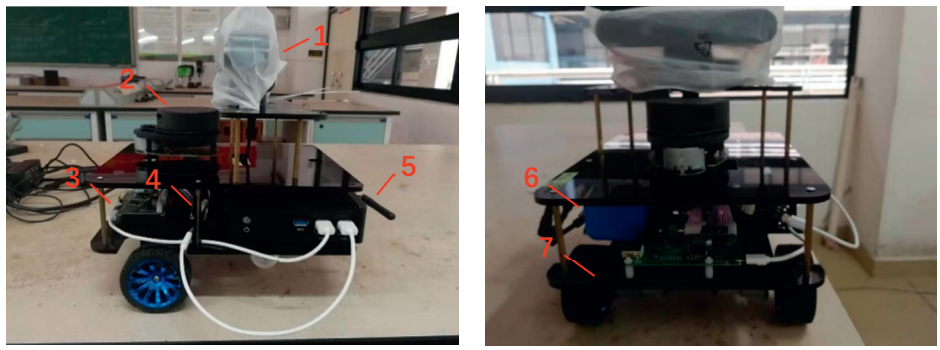


FIGURE 1: System structure of our mobile robot. "STM32 robot control module" gets information of motors from "encoder" and motion from "gyroscope" to control motors and transfer the motion and encoder information to "IPC" which is a microcomputer. Then, "IPC" gets information from Lidar, "STM32 robot control module," and RGB-D camera.



(a) (b)

FIGURE 2: Key modules of the robot. (1) "RGB-D," used to get RGB image and depth image; (2) "lidar," used to get 2D lidar point cloud; (3) "STM32 controller," used to control the movement of the car; (4) "gyroscope," used to get the attitude information; (5) "WI-FI," used to contact with the host computer; (6) "power supply system"; and (7) "motor and encoder," used to get the velocity feedback of the car movement.

(iii) Resampling: depending on the weights, particles with smaller weights are discarded and replaced by resampled particles, but the total number of particles in the resampled particle set is unchanged.

(iv) Map updating: the map update is implemented by the pose represented by each particle in combination with the current observation. To reduce the computational complexity, a recursive formula for weight update is used:

$$w_t^{(i)} = w_{t-1}^{(i)} \eta \frac{p\left(z_t | x_{1:t}^{(i)}, z_{1:t-1}\right) p\left(x_t^{(i)} | x_{1:t-1}^{(i)}, u_{1:t-1}\right)}{\pi\left(x_{1:t}^{(i)} | z_{1:t}, u_{1:t-1}\right)}, \quad (3)$$

where η is a normalisation factor.

3.1.2. Proposal Distribution. A large number of particles will cause a large amount of calculation and memory consumption. In order to reduce the number of particles, a proposal distribution is used. Our target distribution is the best distribution of the robot state according to the data of all sensors carried by the robot. Except for the odometer model, the laser observation data is the position information of 360-degree points, which is difficult to perform Gaussian modelling. Thus, there is no direct way to sample the target distribution, and the proposal distribution is used instead of the target distribution to extract the robot pose information at the next time instant. The proposal distribution considers not only the motion (odometer) information but also the most recent observation (laser) information. This can make the proposal distribution more accurate and closer to the target distribution.

The sensor observation information is added when calculating the proposal distribution, and the sampling process is concentrated in the peak region of the likelihood function to get the optimal proposal distribution:

$$p\left(x_t|x_t^{(i)}, m_{t-1}^{(i)}, z_t, u_{t-1}\right) = \frac{p\left(z_t|x_t, m^{(i)}\right)p\left(x_t|x_{t-1}^{(i)}, u_{t-1}\right)}{p\left(z_t|x_{t-1}^{(i)}, m^{(i)}, u_{t-1}\right)}. \tag{4}$$

Then, the weights are updated according to the above weight recursion formula:

$$\begin{aligned} w_t^{(i)} &= w_{t-1}^{(i)}\eta\frac{p\left(z_t|x_t, m^{(i)}\right)p\left(x_t|x_{t-1}^{(i)}, u_{t-1}\right)}{p\left(x_t^{(i)}|x_{t-1}^{(i)}, m^{(i)}, z_t, u_{t-1}\right)}, \\ &\propto w_{t-1}^{(i)}\eta\frac{p\left(z_t|x_t, m^{(i)}\right)p\left(x_t|x_{t-1}^{(i)}, u_{t-1}\right)}{p\left(z_t|x_{t-1}^{(i)}, m^{(i)}, u_{t-1}\right)}, \\ &= w_{t-1}^{(i)}p\left(z_t|x_{t-1}^{(i)}, m^{(i)}, u_{t-1}\right). \end{aligned} \tag{5}$$

The Gaussian distribution is used to approximate the approximated peak region of the observation, and the optimal proposal distribution is obtained. The Gaussian distribution parameters, i.e., the means $\mu_t^{(i)}$ and covariances $\sum_t^{(i)}$, are determined using K sampling points:

$$\begin{aligned} \mu_t^{(i)} &= \frac{1}{\eta^{(i)}}\sum_{j=1}^{K} x_j p\left(z_t|x_j, m_{t-1}^{(i)}\right)p\left(x_j|x_{t-1}^{(i)}, u_{t-1}\right), \\ \sum_t^{(i)} &= \frac{1}{\eta^{(i)}}\sum_{j=1}^{K} p\left(z_t|x_j, m_{t-1}^{(i)}\right)p\left(x_j|x_{t-1}^{(i)}, u_{t-1}\right), \\ &\cdot \left(x_j - \mu_t^{(i)}\right)\left(x_j - \mu_t^{(i)}\right)^T, \end{aligned} \tag{6}$$

where the normalising factor $\eta^{(i)}$ is given by

$$\eta^{(i)} = \sum_{j=1}^{K} x_j p\left(z_t|x_j, m_{t-1}^{(i)}\right)p\left(x_j|x_{t-1}^{(i)}, u_{t-1}\right). \tag{7}$$

3.1.3. Adaptive Resampling. Resampling may cause good particles to be removed from the filter, making the particles scarce. Therefore, it is necessary to judge the quality of the particles by the effective sampling scale standard and judging the time of resampling. The evaluation formula is as follows:

$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^{N}\left(\tilde{w}^{(i)}\right)^2}, \tag{8}$$

where N is the number of particles and $\tilde{w}^{(i)}$ is the weight of the ith particle. The worse the proposal distribution estimate, the smaller the N_{eff} is. When $N_{\text{eff}} < (1/2)N$, GMapping performs resampling.

3.2. Hector-SLAM Algorithm. The Hector-SLAM algorithm [12] differs from other grid-based mapping algorithms, as it does not require odometer information, but it needs laser data and a priori map. Hector-SLAM is based on the Gauss–Newton iteration formula that optimally estimates the pose of the robot as represented by the rigid body transformation $\xi = [p_x, p_y, \psi]^T$ from the robot to the prior map. The optimal estimation is done by optimally matching the laser data and the map in the sense that the optimal ξ^* below is solved:

$$\xi^* = \arg\min_\xi \sum_{i=1}^{N}\left[1 - M\left(S_i(\xi)\right)\right]^2. \tag{9}$$

Here, $M(S_i(\xi))$ is the value of the map at $S_i(\xi)$, and $S_i(\xi)$ is the world coordinate of scan end points $s_i = (s_{i,x}, x_{i,y})^T$, which obeys the following function:

$$S_i(\xi) = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix}\begin{bmatrix} s_{i,x} \\ s_{i,y} \end{bmatrix} + \begin{bmatrix} p_x \\ p_y \end{bmatrix}. \tag{10}$$

When an initial estimate of the pose ξ is given, an updated estimate $\xi + \Delta\xi$ is computed by approximating $M(S_i(\xi + \Delta\xi))$ using first-order Taylor expansion, and the result is as follows:

$$\Delta\xi = H^{-1}\sum_{i=1}^{N}\left[\nabla M\left(S_i(\xi)\right)\frac{\partial S_i(\xi)}{\partial\xi}\right]^T\left[1 - M\left(S_i(\xi)\right)\right], \tag{11}$$

where H is the Hessian matrix or some approximation of it, given by

$$H = \left[\nabla M\left(S_i(\xi)\right)\frac{\partial S_i(\xi)}{\partial\xi}\right]^T\left[\nabla M\left(S_i(\xi)\right)\frac{\partial S_i(\xi)}{\partial\xi}\right]. \tag{12}$$

3.3. Cartographer Algorithm. When the amount of data to process becomes too large, particle-based algorithms are not applicable due to their higher computing requirements on the processor. In this case, graph optimisation algorithms are more suitable.

Google's solution to SLAM, called Cartographer, is a graph optimisation algorithm. The Google open source code1 consists of two parts: Cartographer and Cartographer_ROS. The function of Cartographer is to process the data from Lidar, IMU, and odometers to build a map. Cartographer_ROS then acquires the sensor data through the ROS communication mechanism and converts them into the Cartographer format for processing by Cartographer, while the Cartographer processing result is released for display or storage. Impressive real-time results for solving SLAM in 2D have been described in [13] by the authors of the software.

3.4. Considering the Rescue Environment. In rescue environment, there are stairs and rugged surface which make the odometer inaccurate. It means we could not choose GMapping because it is very rely on odometer. Due to the

rugged surface, IMU is also inaccurate which means Cartographer may get bad results. So, we choose Hector-SLAM.

4. Path Planning

To achieve path planning is to solve three basic problems:

(1) The robot reaches the desired position

(2) The obstacle avoidance and completion of the strategic task are achieved in the moving process

(3) The optimal path is realized

However, in the actual environment, due to the accuracy of the robot sensor and the variability of the environment, the environmental information and location information of the map construction will be deviated [14]. Global planning in a static environment can meet the problem requirements, but to handle the deviation caused by the dynamic environment, local path planning needs to be introduced. That is to say, the local path planning pays more attention to obstacle avoidance, and the global path planning pays more attention to the shortest path. Therefore, the combination of local planning and global planning algorithms can successfully achieve accurate navigation of the robot. The global planning algorithm studied in this paper is a node-based A^* algorithm [15, 16], and the local planning algorithm is a dynamic window algorithm (DWA) [17]. Global path planning produces a high-level plan for the robot to follow to reach the goal location. Local path planning is responsible for generating velocity commands for the mobile unit to safely move the robot toward a goal. These properties are imbedded in the plan produced by the planners, using the cost function which takes into account both distance to obstacles and distance to the path.

4.1. Global Path Planning. Based on the global path planning of the grid method, the A^* algorithm is used to study the path planning. The A^* algorithm follows the cost function to make the robot to directionally search for the path toward the end point. The core valuation function of the A^* algorithm is

$$f(n) = g(n) + h(n), \tag{13}$$

where the node n is abstractly understood as the next target point, $f(n)$ represents the total valuation function of the current node n, $g(n)$ represents the actual cost of the starting point to the current point, and $h(n)$ represents the estimated cost of the current node to the end point. The value of $h(n)$ determines the performance of the algorithm. Typically, $h(n)$ uses the Euclidean distance or Manhattan distance between the two points in space. In the A^* algorithm, the Manhattan distance is used. The Manhattan distance between two points (x_1, y_1) and (x_2, y_2) is as follows:

$$D_{\text{Manhattan}} = |x_1 - x_2| + |y_1 - y_2|. \tag{14}$$

4.2. Local Path Planning. This paper mainly studies the corresponding action strategy and operation of robot for path navigation in indoor environment. For this reason, the DWA algorithm is selected as the main algorithm for local path planning. The DWA algorithm requires the robot to perform numerical simulation calculations on the path of the robot within a certain speed window. Thus, it is necessary to obtain the model state expression of the robot. The two-wheeled robot based on differential drive has no velocity in the y-axis direction. Since the robot is at the millisecond level in each sampling period of the program execution, the motion trajectory of the robot in the two adjacent sampling periods can be approximated as a straight line. In a period of time Δ, the robot travels a small distance at speed v, and it is at an angle θ_t to the x-axis; then, the movement increments Δx and Δy of the robot on the x-axis and the y-axis can be obtained, respectively:

$$\Delta x = x + v\Delta t \cos(\theta_t), \tag{15}$$

$$\Delta y = y + v\Delta t \sin(\theta_t). \tag{16}$$

The robot's movement trajectory is then given by

$$x_{t+1} = x_t + v\Delta t \cos(\theta_t), \tag{17}$$

$$y_{t+1} = y_t + v\Delta t \sin(\theta_t), \tag{18}$$

$$\theta_{t+1} = \theta_t + \omega\Delta t, \tag{19}$$

where ω is the angular velocity of the robot.

During the speed sampling of the robot, multiple sets of trajectory velocity values are collected. To make the robot safely perform path planning, some necessary speed limits are also needed. The speed velocity value and angular velocity value of the robot change within a certain range, and the range needs to be empirically calculated according to the physical characteristics of the robot and the operating environment. The range formula is as follows:

$$V_m = \{(v, \omega) | v \in [v_{\min}, v_{\max}], \omega \in [\omega_{\min}, \omega_{\max}]\}. \tag{20}$$

The robot has different torque performance parameters due to different motor models. When the current speed of the robot v_c and the angular velocity ω_c are known, the actual speed range for the next sampling time can be computed as

$$V_a = \{(v, \omega) | v \in [v_c - \dot{v}_d\Delta t, v_c + \dot{v}_a\Delta t], \omega \in [\omega_c - \dot{\omega}_d\Delta t, \omega_c + \dot{\omega}_a\Delta t]\}, \tag{21}$$

where $\dot{v}_a$ and $\dot{\omega}_a$ are the maximum accelerations and $\dot{v}_d$ and $\dot{\omega}_d$ are the maximum decelerations.

When the robot is safely avoiding obstacles in navigation, the speed (v, ω) during the whole locally planned trajectory must be within the range of speeds given by

$$V_d = \left\{(v, \omega) | \sqrt{2\text{dis}(v, \omega)\dot{v}_d} \geq v_c, \sqrt{2\text{dis}(v, \omega))\omega_d} \geq \omega_c\right\}, \tag{22}$$

where $\text{dis}(v, \omega)$ is the minimum distance from the current position to the point where the arc trajectory of v and ω intersects the nearest obstacle.

Performing the DWA algorithm for speed selection needs to satisfy equations (19)–(21) simultaneously. On the basis of the trajectory that satisfies these speed requirements, an evaluation function is used to measure a selected trajectory and to aim the selection of the optimal trajectory. The evaluation function is as follows:

$$G(v, \omega) = \sigma(\alpha\text{head}(v, \omega)) + \beta\text{dis}(v, \omega) + \gamma\text{vel}(v, \omega), \tag{23}$$

where $\text{head}(v, \omega)$ represents the angle difference between the estimated end of the route and the target; $\text{dis}(v, \omega)$ is the minimum distance from the obstacle to the planned trajectory, as explained above; $\text{vel}(v, \omega)$ indicates the moment speed evaluation; $\sigma(\cdot)$ is a smoothing function; and $\alpha, \beta, \gamma > 0$ are evaluation coefficients.

5. Experimental Results

5.1. Simulation Experiments

5.1.1. SLAM Simulation. In order to test the aforementioned algorithms, we carry out simulation experiments using the Gazebo platform to build the simulation environment shown in Figure 3. The virtual environment has real physical properties, and the simulation results have strong reference to the actual environment.

The following simulation experiment was performed according to the virtual environment. We first use Lidar as a sensor to simulate the GMapping, Hector-SLAM, and Cartographer algorithms. We then use depth camera (RGB-D) as a sensor to simulate the GMapping algorithm. Four mapping algorithms under the physical simulation platform of the ROS robot system are tested, and the simulation results are shown in Figures 4–7. The purpose of presenting these diagrams is to give an impression of the algorithms described above, which are parsed using text and formulas.

Figure 4 shows the robot simulation process using GMapping. Depth information of the Lidar is required. The robot is located in the lower left corner of simulation environment, the data collected by the Lidar is marked in red, and the established environment map is in light gray. Figure 5 shows the robot simulation process using Hector-SLAM. Figure 6 shows the robot simulation process using Cartographer. The area scanned by the Lidar changes from light gray to white until the whole map is completed.

Figure 7 shows the final grid map constructed using the RGB-D camera data. The depth data are first transformed using depthimage_to_laserscan before being applied to GMapping. The simulation results show that the constructed map is not ideal. This is because the RGB-D camera is affected by its own structure, causing a limited range of viewing angle [18]. In addition, the depth camera requires rich scene features to work. The simulated environment has smooth wall surfaces with very few scene features, making the depth camera unable to perform effective feature matching. We see that map construction cannot be successfully completed.

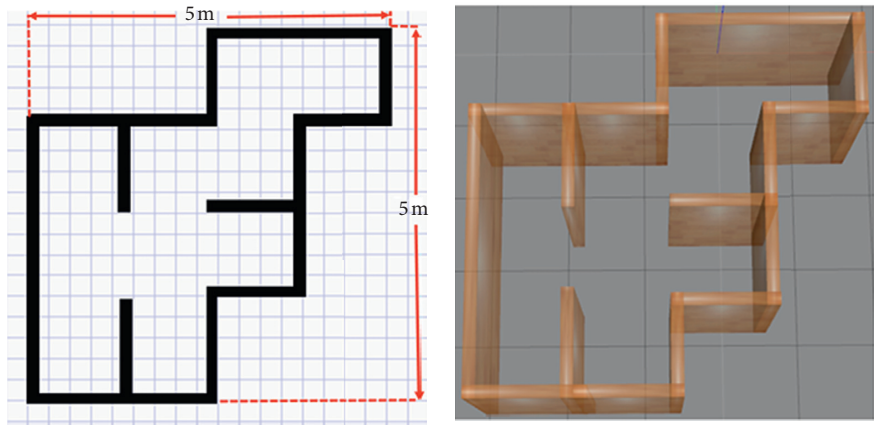
5.1.2. Path Planning Simulation. After obtaining the environment map, the map was loaded under the ROS framework for path planning purposes. We use the rviz package under the ROS framework for path planning and navigation simulation. The constructed map is shown in the left part of Figure 8. Here, we also see the black dot indicating an obstacle, and the cyan portion indicates the safe distance between the robot and the obstacle. The green arrow points to the target point and direction of the robot. The right part of Figure 8 shows the path planning and navigation results, as indicated by the green line. The starting position for the robot is slightly below the obstacle. We see that the simulated path not only successfully avoids the obstacle but also is a nearly straight path.

5.2. Algorithm Verification in Lab Environment

5.2.1. Emulated Rescue Experiment. In this experiment, a 1.25 m × 1.25 m square wooden structure was used to splicing the actual environment in an open laboratory, as shown in Figure 9. This was built with reference to the RoboCup rescue venue to simulate an enclosed indoor rescue environment after a disaster. The aforementioned SLAM algorithms are applied, and the results are shown in Figure 10. Comparing with the simulation results, the maps constructed in the actual environment using Lidar data (GMapping, Hector-SLAM and Cartographer) are all satisfactory. In the RGB-D experiment [19, 20], the environmental features are not sufficient for depth measurements, resulting in an overlapped map. Therefore, the mapping algorithm based on the depth camera needs further optimisation.

5.2.2. Lab Office Experiment. Next, we test the SLAM algorithms in a real lab office, as shown in Figure 11. The map in Figure 12 is constructed using the Hector-SLAM algorithm. We see that not only desks are clearly identified but also the chair legs are clearly shown.

Path planning is carried out using the A^* algorithm for global path planning and DWA algorithm for local path planning, and the results are shown in Figure 13. In the picture on the left, the navigation target point and direction of the robot are set by the green arrow. The small green patch at the bottom shows many arrows representing the particles (their positions and directions) of the starting pose of the robot. The right picture shows that, after the execution of the path planning and navigation, the robot moves to the target



(a) (b)

Figure 3: Simulation environment diagram and Gazebo display.

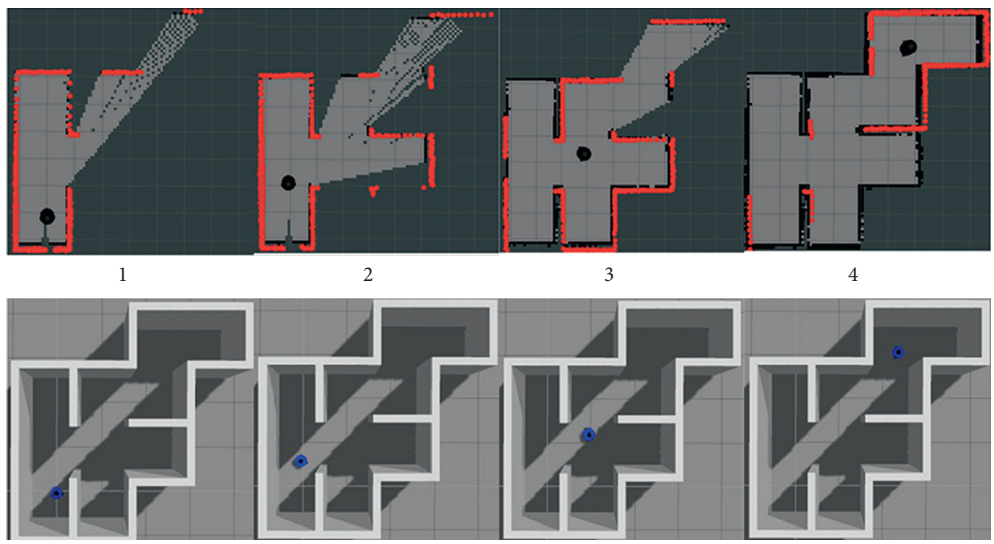


Figure 4: Mapping result of gmapping (simulation).

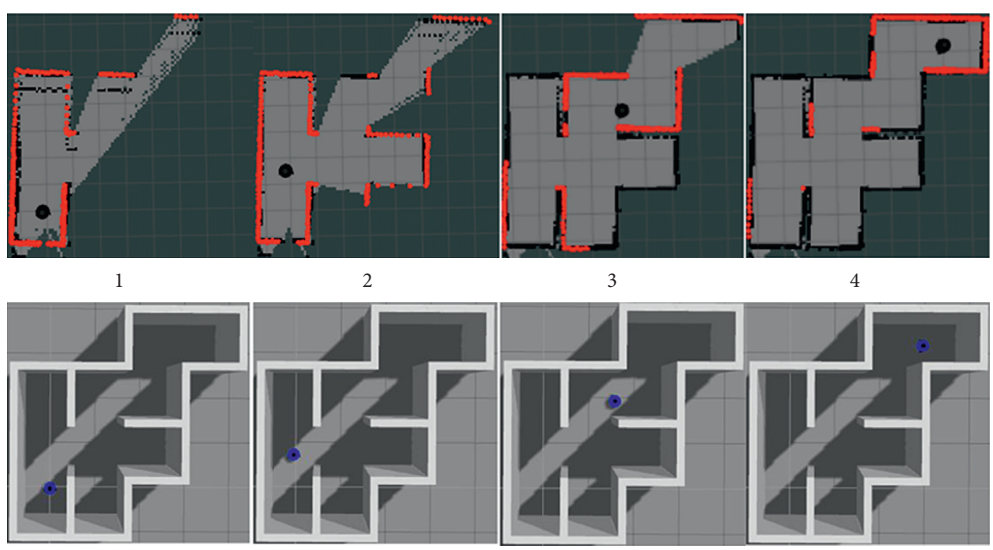


Figure 5: Mapping result of Hector-SLAM (simulation).

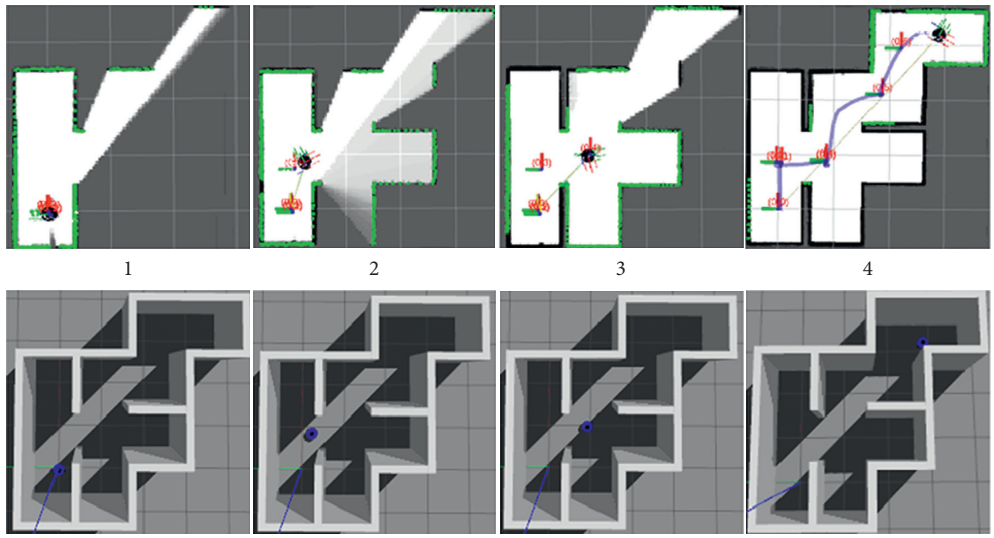


Figure 6: Mapping result of Cartographer (simulation).

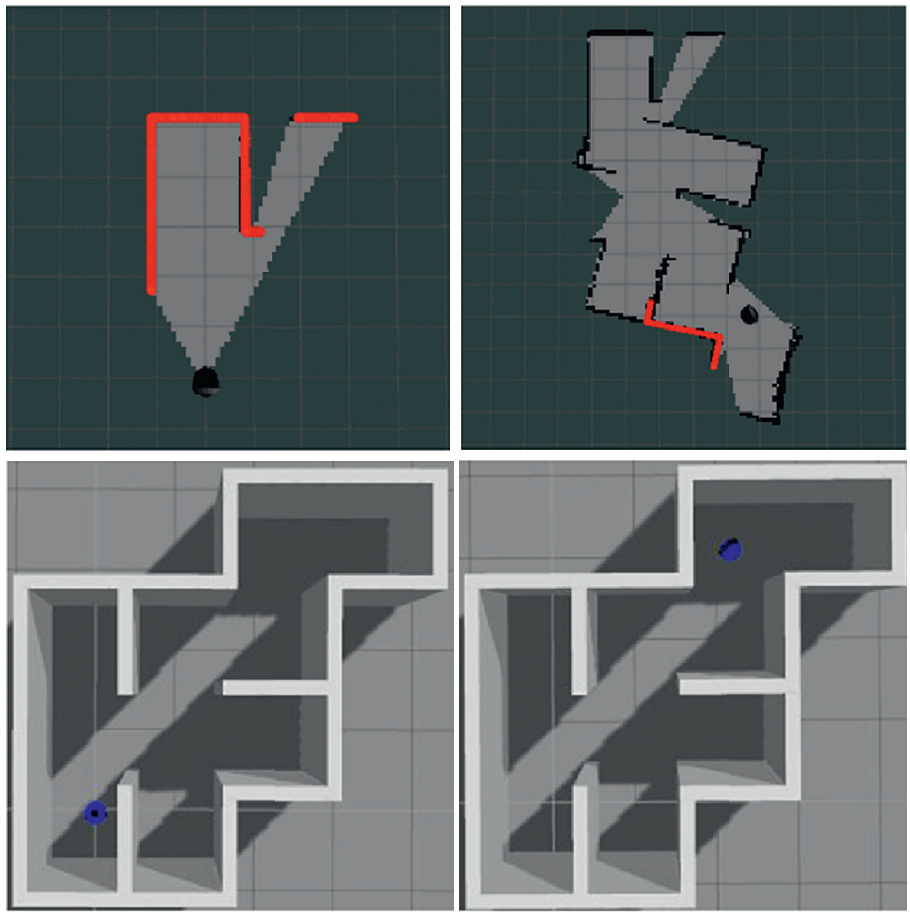
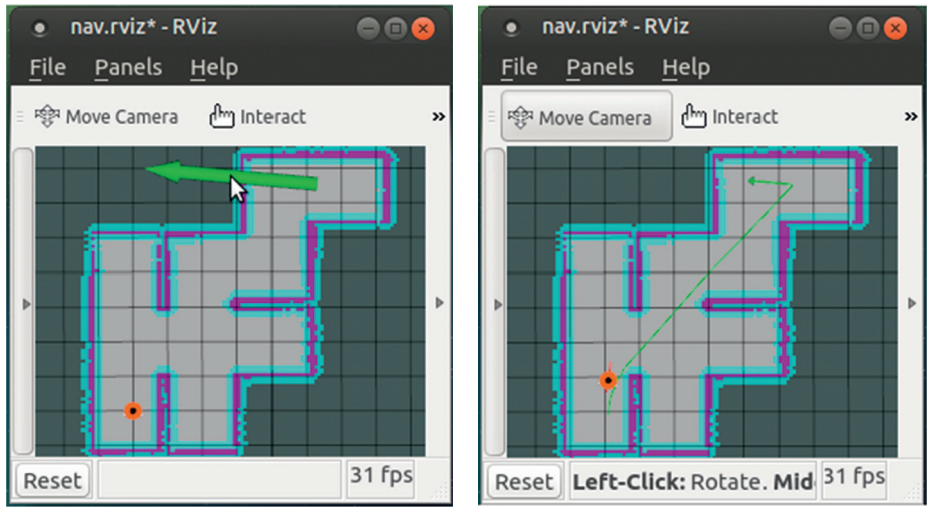


Figure 7: Mapping result of RBG-D (simulation).

point, as indicated by the (smaller) green patch representing the particles of the final pose. We see that the experimental results verify the effectiveness of the path planning algorithms.

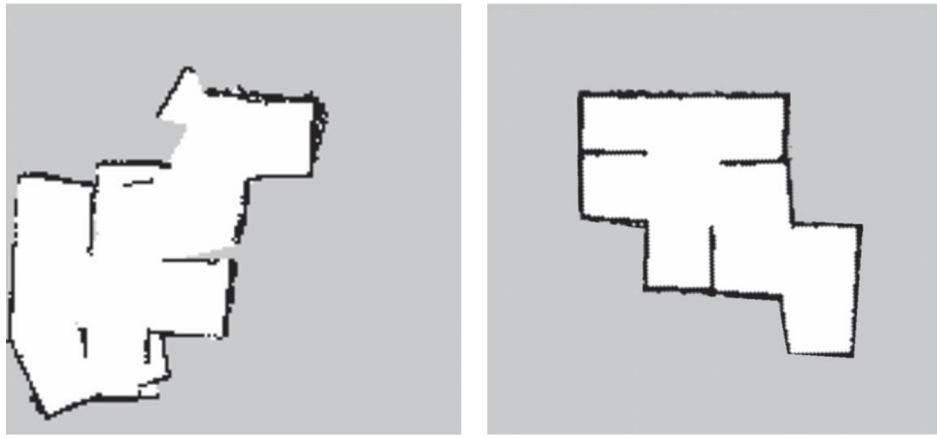
5.2.3. RoboCup Competition Test. The mapping and path planning algorithms above are used in our entry of the 2019 RoboCup competition for indoor rescue [21]. The competition venue is shown in Figure 14. The competition re-



(a) (b)

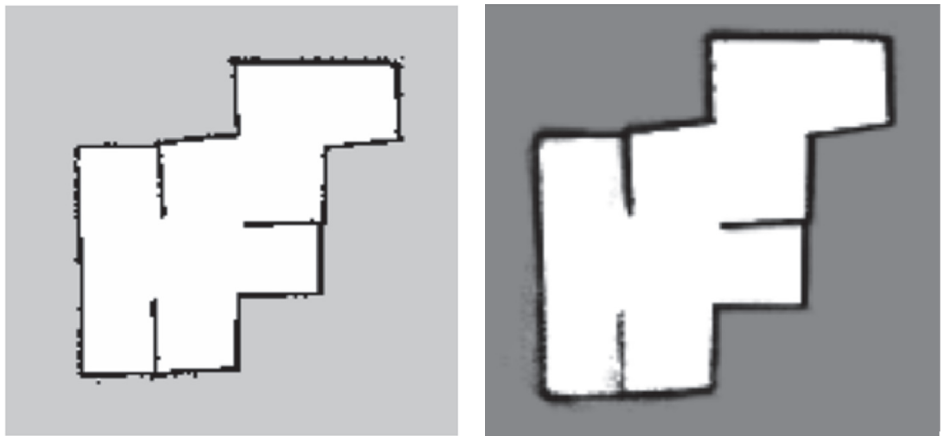
Figure 8: Simulation result of path planning. The green arrow on the left picture means the orientation of the robot when it reaches the end.

Figure 9: Emulated rescue environment.



(a) (b)

Figure 10: Continued.



(c) (d)

Figure 10: Constructed map: (a) RGB-D, (b) Hector-SLAM, (c) GMapping, (d) Cartographer.

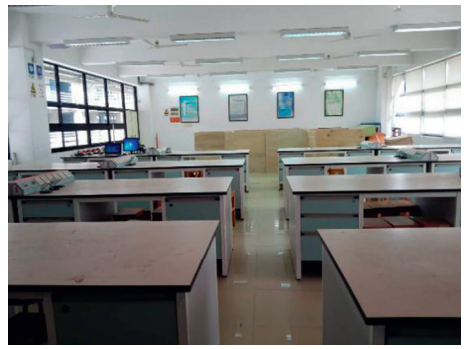
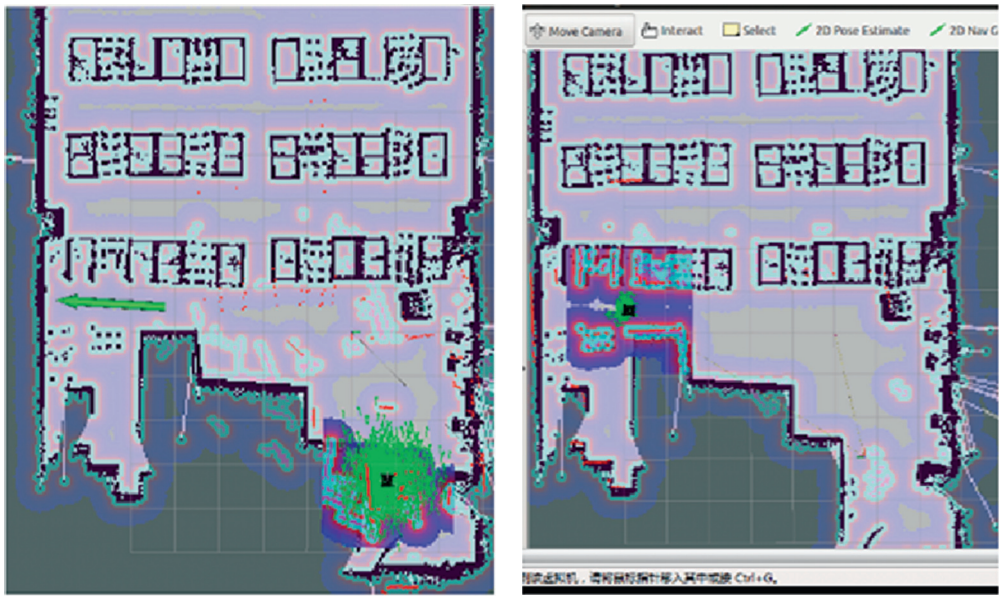


Figure 11: Lab office environment.

Figure 12: Hector-SLAM map for lab office.

quirements are as follows: robot constructs the map of the competition venue by self-exploration and completes the recognition of the doll and the marking of the QR code. The robot operates by either remote control or through an autonomous exploration algorithm. In addition, the robot must avoid obstacles. The quality of the mapping result is assessed by the number of closed grids identified in the constructed grid map. Because GMapping and Cartographer need IMU to assist positioning, and in uneven terrain in the competition, IMU data will have very large errors, which leads to very large errors in these two SLAM algorithms. The RGB-D algorithm also does not perform well in flat terrain,



(a) (b)

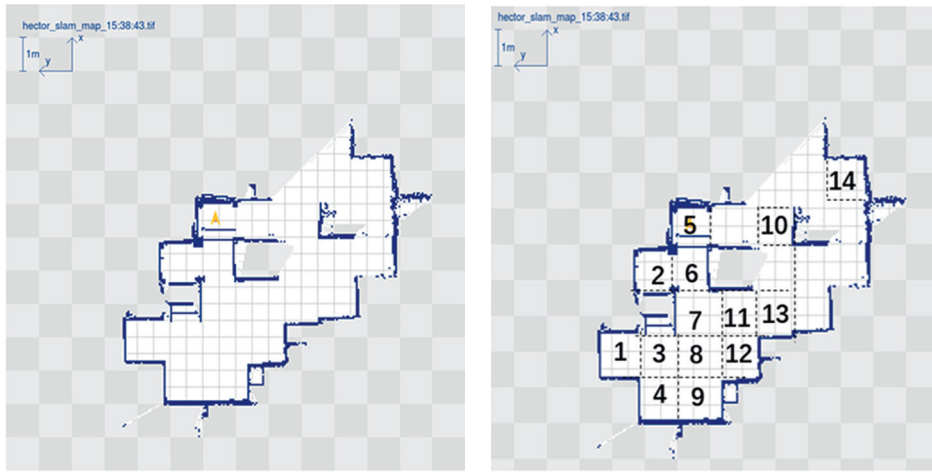
Figure 13: Path planning for lab office.

(a) (b)

Figure 14: RoboCup venue.

so we use the Hector-SLAM algorithm for mapping. And we use the Hector_navigation open source software package [22] for robot self-exploration. As shown in Figure 15, due to the complex environment of the competition venue, the robot is unable to pass some obstacles, such as the 15-degree slope and stairs, so certain parts of the venue cannot be scanned by the Lidar, and the constructed map is incomplete. Due to the limitation of the robot hardware, it is not possible to finish all the mapping. But all the places reached by the robot have been well-mapped. Finally, we scored 14 points (out of 27 points) in the self-exploration session.

5.2.4. China Robot Competition. Compared with the RoboCup competition venue, the 2019 China Robot Competition venue has a larger site area and a larger slope, which means that the terrain is more complicated, as shown in Figure 16. The competition requires rescue robots to independently explore the map of the field and identify the two-dimensional code on the box in the simulated post-disaster environment. In this experiment, we continue to use the Hector_navigation open source software package for robotic exploration. The route of the robot's autonomous navigation is shown in Figure 17. We used two servos to keep the Lidar level, which is used to automatically adjust the Lidar to a horizontal position. However, due to the fact that the competition field is not flat, which causes the robot to have large fluctuations during the movement, the Lidar is unable to adjust the pose in time. As shown in Figure 18, the yellow arrow indicates the starting point of the rescue robot, the purple line marks the movement trajectory of the rescue robot, the dark blue line represents the map constructed by the SLAM algorithm on the competition venue, and the dark blue circle with numbers represents the location of the QR code. However, the robot is navigated outside the wall. As a result, there is a positioning error. The map cannot be quickly updated for corrections, and the navigation algorithm may incorrectly navigate the robot into an obstacle.

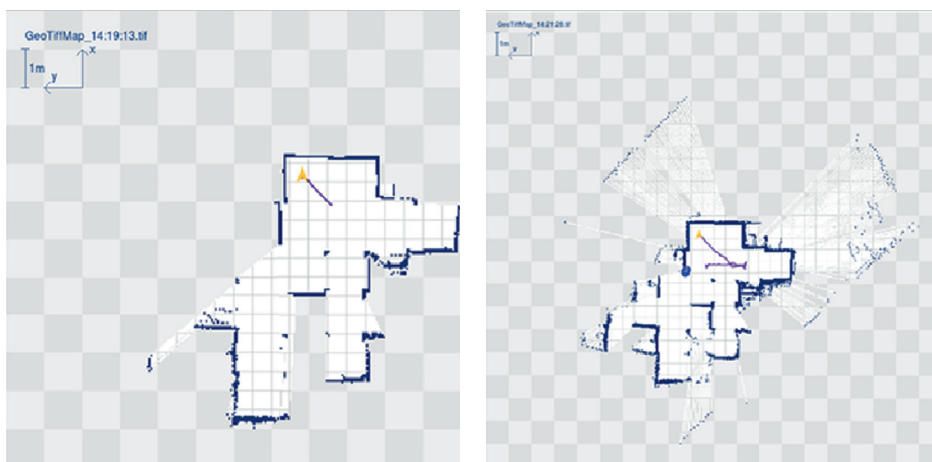


(a) (b)

Figure 15: Mapping result of Hector-SLAM (RoboCup).

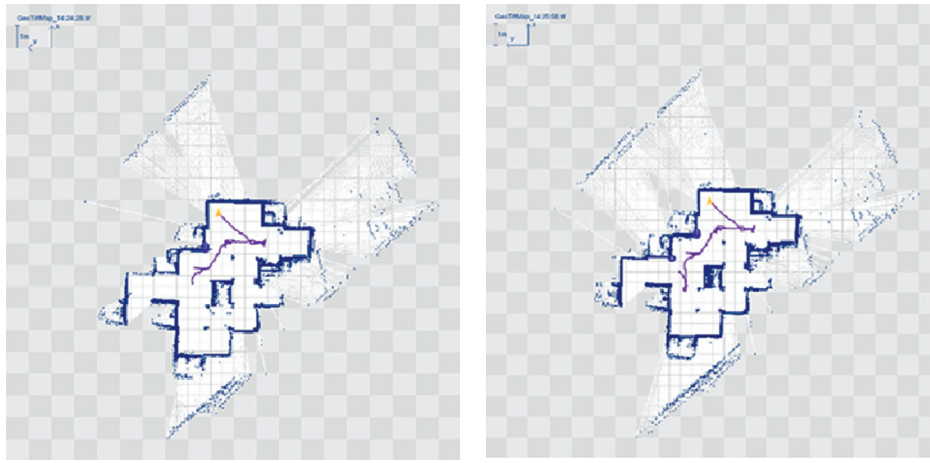
(a) (b)

Figure 16: China robot competition venue.



(a) (b)

Figure 17: Continued.



(c) (d)

Figure 17: Mapping result and autonomous navigation route (China robot competition). (a) 1, (b) 2, (c) 3, (d) 4.

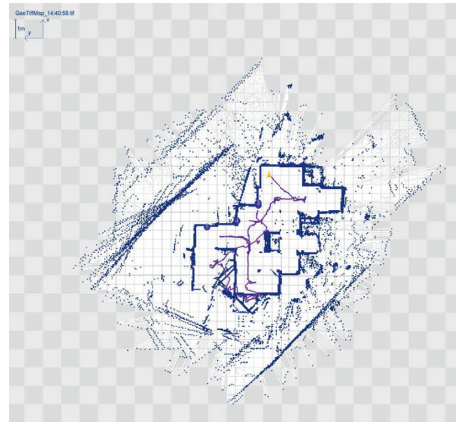


Figure 18: Mapping result of Hector-SLAM (China Robot Competition). Because the level keeping platform of the radar is not sensitive enough, and it cannot be adjusted and keep the platform in time when the robot crosses obstacles, the radar may scan the walls outside the field. Hector cannot judge and filter, so it is recorded on the map.

Therefore, the Hector-SLAM algorithm needs to be improved in the update speed, and the navigation algorithm should adopt a more cautious strategy according to the application of the rescue scenario. As shown in Figure 18, an unnecessary closed point appears on the periphery of the constructed map. In addition, the number in the figure is the position information of the recognised QR code.

In the above simulation experiments and field experiments for GMapping, Hector-SLAM, Cartographer, and RGB-D mapping algorithms, GMapping, Hector-SLAM, and Cartographer perform better in a flat indoor environment. The RGB-D mapping algorithm has poor mapping effect due to poor lighting conditions and lack of environmental features. Therefore, in a relatively flat indoor rescue environment, it is more appropriate to choose the first three algorithms as the basis of the SLAM system. In the RoboCup competition and the China Robot Competition, the uneven rescue environment seriously interfered with the IMU data, so the SLAM algorithm (GMapping and Cartographer) incorporating IMU data could not perform SLAM tasks normally. This causes the path planning algorithm that relies on map information to not work correctly. The Hector-SLAM algorithm, which does not rely on IMU data, can perform tasks in the competition terrain relatively correctly, but there are still problems with inability to filter wrong Lidar information and poor stability.

6. Conclusions

In this paper, the problem of indoor rescue using mobile robots was studied. Comparisons were done on the GMapping, Hector-SLAM, and Cartographer algorithms for SLAM. The path planning was done by combining the A^* algorithm for global path planning and the DWA algorithm for local path planning. Simulation, emulation, as well as real environment experiments were conducted to compare and validate the results on map construction and path planning. In the future, further optimisation needs to be carried out in the mapping algorithms to make them more suitable to the real rescue environment.

Data Availability

Our experiment data can be found in https://github.com/9393dl/Rescue_Robot.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant nos. 61633014, 61803101, and U1701264).

References

[1] W. Peng, F. Yuan, and Z. Zhou, "Positioning analysis and implementation of mobile robot based on RGB-D camera," *Intelligent Computer and Applications*, vol. 9, no. 3, pp. 168–170, 2019.

[2] J. Shou, Z. Zhang, Y. Su, and Z. Zhong, "Design and implementation of indoor positioning and navigation system of mobile robot based on ROS and lidar," *Machinery and Electronics*, vol. 36, no. 11, pp. 76–80, 2018.

[3] H. Zhang, X. Zhang, and Q. Wang, "Path planning based on path prediction artificial potential field method for automatic following trolley," *Computer Measurement and Control*, vol. 27, no. 1, pp. 237–240, 2019.

[4] Z. Liu, S. Jiang, W. Yuan, and C. Shi, "Robot path planning based on deep *Q*-learning," *Measurement and Control Technology*, vol. 38, no. 7, pp. 24–28, 2019.

[5] B. Yu, X. Chu, C. Liu, H. Zhang, and Q. Mao, "Path planning method for unmanned waterway survey ships based on improved A^* algorithm," *Geomatics and Information Science of Wuhan University*, vol. 44, no. 8, pp. 1258–1264, 2019.

[6] S. Sugiyama, H. Okuda, S. Inagaki, and T. Suzuki, "SLAM using ROS and operational assist for electric wheelchairs," in *Proceedings of the JSME Annual Conference on Robotics and Mechatronics (Robomec)*, November 2017.

[7] B. Sun, "Mobile robot SLAM technology," *Electronic Technology and Software Engineering*, vol. 1, no. 2, p. 95, 2018.

[8] Z. Lin and S. Zheng, "Research on image matching and camera pose resolution in mobile robot vision SLAM process," *Machine Design and Manufacturing Engineering*, vol. 46, no. 11, pp. 13–18, 2017.

[9] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with Rao-Blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[10] A. Doucet, J. F. G. de Freitas, K. Murphy, and S. Russel, "Rao-Blackwellized particle filtering for dynamic bayesian networks," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 176–183, Stanford, CA, USA, June 2000.

[11] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2149–2154, Sendai, Japan, September 2004.

[12] S. Kohlbrecher, O. Von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable slam system with full 3D motion estimation," in *Proceedings of the 2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, Kyoto, Japan, November 2011.

[13] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D Lidar SLAM," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2016)*, Stockholm, Sweden, May 2016.

[14] X. Wang, L. He, and T. Zhao, "Mobile robot for SLAM research based on lidar and binocular vision fusion," *Chinese Journal of Sensors and Actuators*, vol. 31, no. 3, pp. 394–399, 2018.

[15] P. Hart, N. Nilsson, B. Raphael et al., "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[16] Z. Cao, S. Jiang, J. Zhang, and H. Guo, "A unified framework for vehicle rerouting and traffic light control to reduce traffic congestion," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 7, pp. 1958–1973, 2017.

[17] M. Seder and I. Petrovic, "Dynamic window based approach to mobile robot motion control in the presence of moving obstacles," in *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, pp. 1986–1991, Roma, Italy, April 2007.

[18] Z. Cao, H. Guo, J. Zhang, D. Niyato, and U. Fastenrath, "Finding the shortest path in stochastic vehicle routing: a cardinality minimization approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 6, pp. 1688–1702, 2016.

[19] Y. Li, S. Wang, and N. Yu, "RGB-D-based SLAM and path planning for mobile robots," *CAAI Transactions on Intelligent Systems*, vol. 13, no. 3, pp. 445–451, 2018.

[20] J. Cai, K. Chen, and Y. Zhang, "Improved V-SLAM for mobile robots based on Kinect," *CAAI Transactions on Intelligent Systems*, vol. 13, no. 5, pp. 734–740, 2018, http://kns.cnki.net/kcms/detail/23.1538.tp.20180423.0957.004.html.

[21] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa, "RoboCup: the robot world cup initiative," in *Proceedings of the International Conference on Autonomous Agents*, Marina Del Rey, CA, USA, February 1997.

[22] S. Kohlbrecher, J. Meyer, T. Graber, K. Petersen, U. Klingauf, and O. Von Stryk, "Hector open source modules for autonomous mapping and navigation with rescue robots," in *Proceedings of the Robot Soccer World Cup*, pp. 624–631, Joao Pessoa, Brazil, January 2014.