Simulation Analysis of Fire-Fighting Path Planning Based On SLAM

Jingliang Jing *

School of International, Suzhou University of Science and Technology, Suzhou 215009, China

* Corresponding Author Email: 21200153119@post.usts.edu.cn

Abstract. Reasonable and optimal planning of fire fighting paths is crucial to improving the efficiency of fire fighting, and has attracted increasing research attention in recent years. The traditional fire path selection method is mainly to locate the hot spot of the fire scene, and at the same time, use the hot spot sensor to locate the path correction algorithm to realize the planning of the fire path. Thanks to the rapid development of artificial intelligence algorithms, the fire extinguishing path planning and disaster relief resource scheduling methods based on artificial intelligence algorithms have continuously made breakthroughs in accuracy and speed. Based on simultaneous localization and mapping (SLAM) technology, this paper solves the positioning and mapping problems simultaneously to determine the optimal path. Specifically, this paper mainly uses the map construction and navigation functions of SLAM, and uses Gazebo to simulate an open shopping mall environment and a factory environment with multiple obstacles. Through simulation experiments, the results verify the feasibility of this proposed method in fire evacuation path planning, and can realize the evacuation path planning in case of fire.

Keywords: SLAM; path planning; firefighting; simulation.

1. Introduction

With the acceleration of urbanization,more and more super-high-rise multifunctional buildings are appearing, which makes the evacuation of people become more and more complex. In the event of a fire, how to quickly evacuate stranded people and extinguish the fire has become a very challenging task. According to the website of China Fire Magazine, from January to June 2023, a total of 550,000 fire incidents were reported nationwide, where fires in small commercial premises accounted for a large proportion. Within the five-minute optimal time to extinguish a fire, reasonable route planning can enable fire engines to reach their destinations faster and safer, effectively shorten the time to extinguish a fire, minimize property damage, and avoid casualties. Therefore, the reasonable and optimal planning of firefighting paths has attracted more and more research attention in recent years.

A great deal of effort has been invested in the screening and selection of firefighting paths. Traditional firefighting route selection methods mainly locate hotspots at the fire scene, while using hotspot sensors to locate path correction algorithms to realize the planning of firefighting paths. The current research on path planning algorithms mainly includes the improvement of traditional algorithms and the application of intelligent bionic algorithms, including the A * algorithm, Dijkstra algorithm,genetic algorithm,ant colony algorithm, artificial potential field algorithm, etc. Among them, the most popular progamme for many applications are A* and a rapidly exploring random tree (RRT)[1] , and both of them are applicable in static environments. The A* algorithm locates the path with the lowest cost in static settings, which has some useful characteristics, such as intgrity, optimality, and optimal efficiency. The RRT (Rapidly exploring Random Trees) algorithm is suited for specific types of path planning problems, especially those with high dimensionality and real-time requirements. However, due to its randomness and greediness, RRT typically identifies sub-optimal solutions rather than global optimums. The Dijkstra algorithm is a classic for seeking the shortest path in static environments without obstacles. The basic problem of indoor path planning is to find a continuous path connecting the starting position and the goal position while avoiding collision with known obstacles. The commonly used path planning algorithms have ability to plan paths in the

environment with obstacles, while thier performance are still limited due to the complex application scenes.

Thanks to the development and integration of computer and network technology, firefighting path planning and disaster relief resource scheduling based on artificial intelligence algorithms are gradually emerging. Through network technology, real-time environmental information can be collected, and the data model of the fire scene can be reconstructed, which provides a key decision-making basis for firefighting and rescue deployment. At the same time, the use of computers can sort out complicated information, calculate effective paths, realize automatic distribution of firefighting paths, and automatically coordinate multi-party rescue functions. At present, fire trucks mainly use real-time information notification and drone mapping and positioning. In this paper, based on the SLAM (Simultaneous Localization and Mapping) technique, we simultaneously solve the localization and mapping problems and identify the optimal path. Specifically, this paper mainly uses the map building and navigation capabilities of SLAM and utilizes Gazebo to simulate an open shopping mall environment and a factory setting with multiple obstacles. Through the simulation text, the outcome verifies the feasibility of the method in fire evacuation path planning, which can realize the evacuation path planning in case of fire.

2. Method

2.1. Overview of Constructing RTAB-Map

SLAM algorithms are a technology used in robots or mobile devices. They enable devices to localize themselves in real-time within unknown environments and simultaneously build maps of the surroundings, enabling tasks such as navigation and path planning. For autonomous unmanned systems, SLAM technology can be used observe the surrounding static environment and construct a 3D map of the environment through sensors such as cameras and LiDARs installed on the robot [2]. The core idea of SLAM algorithms is to extract environmental information from sensor data and achieve autonomous localization and map building through data processing and fusion.SLAM system consists of two main components: the frontend and the backend. In the frontend step, the SLAM system extracts the constraints from sensor data, such as, by performing feature detection and matching, loop closure detection, etc. Nonlinear optimization is then applied to obtain the maximum likelihood estimation of the fren-like plant [3]. In this SLAM-based mapping and navigation project, four techniques are primarily utilized, which will be discussed in detail in the following subsections. SLAM is a technology that allows a robot to autonomously perform both localization and mapping simultaneously in its environment. Gazebo is a robot simulator that can be integrated with ROS, used for testing and debugging SLAM algorithms in a virtual environment. Testing a virtual robot in Gazebo is an efficient, cost-effective and hardware-independent method that allows for the validation of new concepts and methods, as well as the assessment of their suitability compared to existing robots [4]. ROS (Robot Operating System) is a widely used platform in robot development, providing libraries and packages related to SLAM. It allows execution of SLAM algorithms and facilitates mapping and navigation tasks. RViz is a visualization tool in ROS used to display the data and state of the robot system, including the output of SLAM algorithms.

The method in this paper is mainly based on Real-Time Appearance-Based Mapping (RTAB-MAP) technology, which uses RGBD cameras, stereo cameras or 3D lidars for simultaneous positioning and map construction. In the RGB-D/stereo case, the feature points with depth information of the previous frame are reprojected to the current frame [5]. RTAB-MAP assumes that are accessed more frequently are more likely to form a closed loop than other anchor points, and the number of times such an anchor point is visited continuously can be used to measure its weight for easily forming a closed loop. When it is necessary to transfer the anchor point from Working Memory (WM) to Long-Term Memory (LTM), the anchor point with the lowest weight is preferred. If there are multiple anchor points with the lowest weight, the anchor point with the longest storage time is preferred.

The RTAB-Map workflow mainly includes the following steps. (1) Data Acquisition. Typically, RGB-D cameras or LiDAR sensors are used. RGB-D cameras provide color images and depth maps, while LiDAR sensors provide distance and positional information. (2) Feature Extraction and Matching: RTAB-Map: For RGB images, feature extraction algorithms (e.g., ORB features) are used to extract visual features from color images. Simultaneously, the depth map is used to calculate the 3D positions of feature points. Feature matching is performed to estimate the relative poses between neighboring frames. For depth maps, point cloud data can be used to compute features. (3) Incremental Map Construction. By determining the coordinates of objects in the surrounding environment, it is possible to generate a map. This process is referred to as mapping [6]. RTAB-Map fuses feature points and depth image data to build an incremental map of the environment. It adds the feature points and pose information of each frame to the map and establishes connections between them. (4) Loop Detection and Optimization. RTAB-Map identifies and resolves loop closures, i.e., previously scanned areas, through loop detection. It checks if similar scenes have been previously visited by comparing the current frame with historical frames. Methods like RANSAC are used to estimate loop closure locations. (5) Robot Self-Localization. Through continuous motion estimation and loop closure detection optimization, RTAB-Map achieves real-time robot self-localization. It estimates the robot's current position in the map. (6) Map Maintenance and Optimization. RTAB-Map continuously updates and optimizes the map by incrementally incorporating new sensor data and utilizing graph optimization algorithms to enhance map consistency and accuracy. (7) Map Saving and Loading. RTAB-Map provides functionality to save the generated map to disk and load it when needed. (8) Loop closure update.

2.2. LTM and WM

Long-Term Memory (LTM) is a relatively persistent memory system used for storing and retrieving a large amount of information and experience. Working Memory (WM) is an essential component of short-term memory. WM is closely associated with higher-level cognitive functions such as thinking, decision-making, and problem-solving. It serves as a temporary storage and manipulation system, enabling us to process and manipulate information in a short period.

2.3. Location Creation

Feature Extraction: Feature extraction is the process of extracting key visual feature points or descriptors from sensor data. These feature points possess uniqueness and stability, allowing for matching across different perspectives. Feature extraction algorithms are used to identify salient points in images or key features in point clouds. Then, matching feature points in adjacent frames are computed and the geometric relationship is exploited to obtain the rotation matrix and translation matrix of the camera[7].

Vocabulary Building: Vocabulary building involves clustering the extracted feature points from the previous step into a set of visual words. Clustering algorithms are employed to group similar feature points together, representing visual feature patterns. Each visual word in the vocabulary model represents a specific visual feature pattern.

Feature Encoding: Feature encoding is the process of encoding the extracted feature points into vectors that represent their distribution within the vocabulary. Encoding methods capture spatial relationships and distribution information between feature points. These encoded vectors compress and represent the information of the feature points in a more concise form.

Vocabulary Matching: Vocabulary matching involves matching the feature points or descriptors with the vocabulary. Similarity measures are calculated between the feature points and each visual word in the vocabulary. Matching methods, such as nearest neighbor or approximate nearest neighbor search, are used to find the best matches. The matching results offer association information between feature points, which is utilized in map building and robot localization.

These four steps are closely interconnected and collectively transform sensor data into usable visual features for localization in RTAB-Map.

2.4. Weight Update

In RTAB-Map, localization points are represented by a collection of feature descriptors called signatures. These signatures are typically formed using a bag-of-words model, where each word represents a feature in the image or map. The words in the signature are stored in a dictionary, which can be calculated by equation (1).

$$s(z_t, z_c) = \begin{cases} \frac{N_{pair}}{N_{zt}}, \text{if}(N_{zt} \geq N_{zc}) \\ \frac{N_{pair}}{N_{zc}}, \text{if}(N_{zt} < N_{zc}) \end{cases} \quad (1)$$

Where N_{pair} is a metric that measures the number of word pairs matched between two localization point signatures. It serves as an indicator of the similarity between the two localization points. N_{zt} and N_{zc} refer to the total number of words in the signatures z_t and z_c, separately. z_t represents the signature words from the reference localization point L_t, while z_c represents the signature words from the candidate localization point L_c.

T similarity is a predetermined similarity threshold. If the similarity between two localization points (calculated using N_{pair}, N_{zt}, and N_{zc}) exceeds this threshold, the candidate localization point L_c is fused into the target localization point L_t. After fusion, the signature retains only the words from z_c, while any new words from z_t are removed.

Weights in RTAB-Map are used to represent the importance of localization points. During the fusion process, the weight of the target localization points L_t increases, typically by adding 1 to the weight of L_c. After fusion, the adjacency and loop closures are reconnected to the target localization point Lt, while the fused localization point L_c is removed from the RTAB-Map data structure, such as the Short-Term Memory (STM).

In summary, in RTAB-Map, localization points are represented by signatures composed of words from a bag-of-words model. N_{pair} measures the matched word pairs between two signatures, and T similarity is a threshold for their similarity. Fusion involves combining the signature words and adjusting the weights, and the resulting fused localization point replaces the candidate point in the data structure while retaining relevant connections.

2.5. Bayesian Filter Update

In the RtabMap solving process, the linear optimization uses a filter model based on Bayesian probability theory[8]. Bayesian filtering where all variables are treated as Gaussian random variables. includes two steps: time update (prediction) and measurement update (filtering)[9]. The Bayesian filter is a probability-based filtering technique that utilizes Bayesian theorem and is commonly used for state estimation and prediction problems. Its purpose is to estimate the probability of the current localization point Lt forming a loop closure with the previously stored localization points in WM to track loop closure hypotheses.

$$p(S_t \mid L^t) = \rho p(L_t \mid S_t) \sum_{i=-1}^{t_n} p(S_t \mid S_{t-1} = i) p(S_{t-1} = i \mid L^{t-1}) \quad (2)$$

$p(L_t \mid S_t)$measures the probability of observing the observation L_t given the state S_t. This is a conditional probability that describes the relationship between the state and the observation.

$$p(L_t \mid S_t = j) = \gamma(S_t = j \mid L_t) = \begin{cases} \frac{s_j - \sigma}{\mu}, & \text{if} s_j \geq \mu + \sigma \\ 1, & \text{otherwise} \end{cases} \quad (3)$$

L_t is a new localization point.

$$p(L_t \mid S_t = -1) = \gamma(S_t = -1 \mid L_t) = \frac{\mu}{\sigma} + 1 \quad (4)$$

$p(S_t \mid S_{t-1} = i)$means the Transition model, which measures the probability of transitioning from state St' to state St given the control input $S_{t-1} = i$. This is also a conditional probability that describes the evolution of the state.

$$p(S_t = -1 \mid S_{t-1} = -1) = 0.9 \tag{5}$$

The probability of generating a new positioning point at time t, assuming that there is no fever cycle closure at time $t-1$.

$$p(S_t = i \mid S_{t-1} = -1) = \frac{0.1}{N_{WM}} (i \in [0, t_n]) \tag{6}$$

Under the condition that no loop closure occurred at time $t-1$, The probability of a loop closure occurring between time t and i.

$$p(S_t = -1 \mid S_{t-1} = j) = 0.1(j \in [0, t_n]) \tag{7}$$

The probability of generating a new localization point at time t, given that a loop closure occurred between time t-1 and j.

$$p(S_t = i \mid S_{t-1} = j)(i, j \in [0, t_n]) \tag{8}$$

The probability of a cycle closure occurs between time t-1 and j, given that a loop closure occurred between time t-1 and j, is defined by a discrete Gaussian curve centered around j. This means that the probability values correspond to i within a neighborhood range of 16 units around j $(i = j - 16, \ldots, j + 16)$ when i is not empty. In the topological map, some localization points may have more than two adjacent localization points (if loop closures have occurred before), and these adjacent points may not all be present in the WM (as they may have been moved to LTM). Finally, the probability $p(S_L \geq 0 \mid S_{t-1} = j)$ is normalized to a distribution with a sum of 0.9.

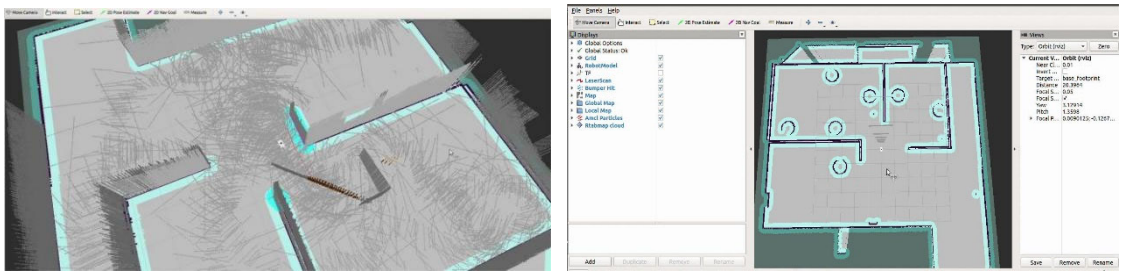
2.6. Loop Closure Hypothesis Selection

The selection of loop closure hypothesis refers to the process in SLAM (Simultaneous Localization and Mapping) tasks, where the best loop closure hypothesis is chosen when a potential loop closure is tested to optimize the map and robot trajectory. The goal of selecting loop closure hypotheses is to choose the hypothesis that best matches the actual loop closure in order to improve map consistency and accuracy. After normalization, if the probability p is smaller than a predefined threshold Ttop, the loop closure hypothesis $S_t = i$ with the highest probability in p is considered to be valid. When a loop closure hypothesis is validated, a closed loop connection is established between the new localization point L_t and the old localization point L_i. The weight of L_t is updated by adding the weight of L_i to the original weight, and the weight of L_i is set to 0. The loop closure link from L_i to L_t is added to the system.

3. Model and simulation scene construction

3.1. Original data

(1) Simulated Car: A four-wheeled car controlled by drive commands.
(2) RGBD Depth Camera: A depth camera that captures RGB color and depth information.
(3) Laser Sensor: A sensor that uses laser technology to measure distances and detect obstacles.
(4) Map Information: Basic mapping of a small open shopping mall and a factory map with additional obstacle settings, which is shown in Figure 1.



(a) Map of small open shopping mall (b) Map of factory

Figure1. Visualization of mappings with additional obstacle settings.

The SLAM (Simultaneous Localization and Mapping) mapping and navigation conducted in this paper can be roughly divided into the following seven parts.

(1) Initialization. The robot is set to start operating in an unknown environment. A simulation environment is established, which includes the map, robot model, and sensor model. The initial state of the SLAM algorithm is also initialized, including map initialization and the robot's initial positioning.

(2) Sensor Data Acquisition. The robot collects environmental information through radar, depth cameras, stereo cameras, or other sensors. This information primarily includes distance and visual data. In the simulation environment, sensor models generate this data. I used LIDAR (Light Detection and Ranging) data, which contains environmental information around the robot, such as obstacles and landmarks. Data is acquired through the "sensor_msgs/LaserScan" message in ROS (Robot Operating System). However, the aforementioned approach requires RGB-D inputs as it solely relies on neural network optimization of the camera pose without visual odometry, leading to poor initial localization.[10]

(3) SLAM Algorithm Execution. Preprocessing is applied to the sensor data, which is then processed using the SLAM algorithm for both localization and mapping. RTAB-Map, which was introduced earlier, is a vision-based SLAM algorithm.

(4) Navigation Planning. Navigation planning is done based on the constructed map model and the robot's state. This includes path planning and obstacle avoidance to enable safe navigation from the origin to the destination. The "move_base" package is used to apply a costmap, dividing each part of the map into occupied (e.g., walls or obstacles) and unoccupied areas. As the robot moves, a local costmap relative to the global costmap is continually updated, allowing the package to define a continuous path for the robot.

(5) ROS amcl Package. This variant is implemented by the "amcl" (Adaptive Monte Carlo Localization) package in ROS. We integrate this package with the robot to locate it within the provided map. "amcl.launch" is executed for this purpose.

(6) Control Command Generation. Based on the results of navigation planning, control commands like speed and direction are generated. These commands are sent to the robot to control its movements.

(7) Simulation Result Presentation. Simulation results are displayed visually, showcasing the robot, map, and navigation paths. The visualization tool "RVIZ" is used for real-time display and analysis of the simulation outcomes.

3.2. Analysis of simulation results

Based on the experimental results in Figure 2, it can be observed that the cart's simulation scanning in an open mall environment proceeds smoothly. During simulated navigation, the cart can effortlessly follow the planned route to simulate a search-and-rescue process. The SLAM (Simultaneous Localization and Mapping) algorithm achieves the expected results in the depicted open mall environment. Due to the mall's open spatial layout and minimal obstructions in the map design, the simulated cart's sensors can effectively perceive the environment and acquire complete map information. In an open mall setting, the SLAM algorithm provides excellent localization accuracy. The relatively simple design of the open mall environment, along with minimal visual

clutter, results in smaller localization errors, allowing the simulated cart to more precisely determine its own position. Using the map established through SLAM, path planning within the mall becomes more efficient and reliable. The cart can plan its route based on the map information, avoid walls and other obstructions, and select the optimal navigation path, thereby reaching the target location more quickly.

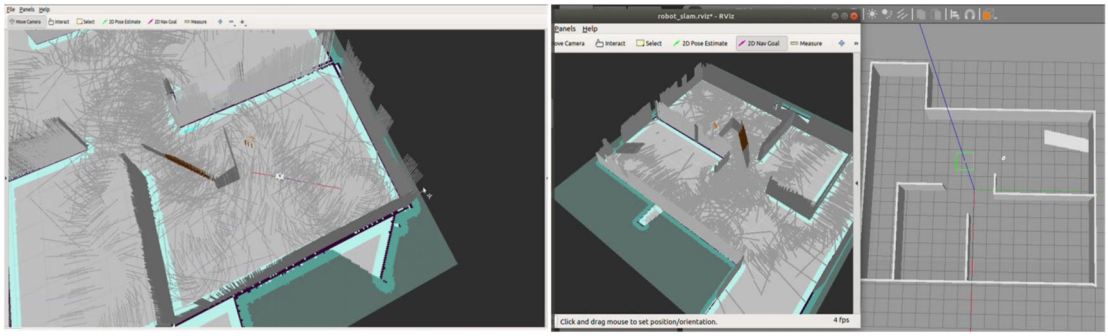


Figure 2. Simulation results in an open mall environment

In a simulated factory environment characterized by multiple obstacles, the task of mapping and localization for the cart presents significant challenges. Within such obstacle-rich factory settings, the SLAM (Simultaneous Localization and Mapping) algorithm might encounter difficulties in constructing a map. When localizing within an environment replete with obstacles, factors such as multipath effects, multiple reflections, and obstructions can lead to increased localization errors. Sensor data may be obstructed or affected, making the creation of an accurate map more challenging. Path planning becomes more intricate in environments laden with obstacles. The cart must avoid various obstructions and select the optimal route to achieve safe and efficient navigation. The SLAM algorithm needs to address the challenges of obstacle perception and planning to ensure the accuracy and feasibility of path planning.

As shown in Figure 3, in obstacle-dense factory settings, the fusion of data from multiple sensors can enhance the mapping and localization results. The cart should integrate data from RGBD depth cameras and laser sensors to precisely avoid obstacles. For the factory environment, the approach of localized mapping and incremental updates, which applies the concepts of LTM (Long-Term Memory) and WM (Working Memory), might be more suitable. The cart can construct a map of local regions and continuously update it as it moves. This method can reduce storage and computational requirements while improving the efficiency of mapping and localization, ensuring a more efficient and accurate attainment of the desired state during the mapping and localization process.

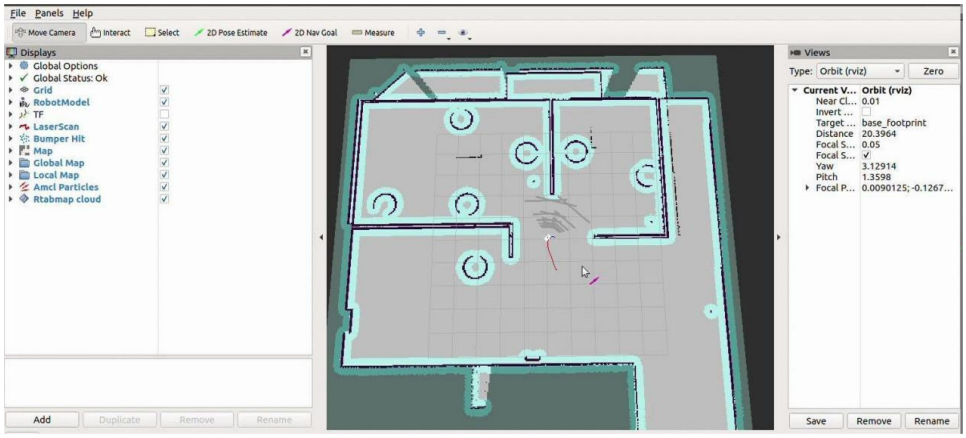


Figure 3. Simulation results in an obstacle-dense factory environment

In both multi-obstacle factory environments and open mall settings, experiments on mapping and localization with SLAM-simulated fire trucks can accurately produce map models, with precise vehicle localization and good path planning results. Due to the complex structure and obstacle interference in multi-obstacle factory settings, the performance of the SLAM algorithm may be

somewhat limited. However, the fire truck can still effectively navigate around obstacles and plan routes sensibly. The process is slower compared to more open areas and requires real-time data to some extent. The SLAM algorithm can achieve better results in mapping quality, localization accuracy, and path planning performance. It can be applied to fire scenarios in various environments for preliminary rescue path planning. By combining specific data information, reasonable routes can be set prior to actual rescue operations, thereby shortening rescue times and saving more lives.

Based on my personal experience, there are several limitations and areas for improvement in this research.1. The simulated car used in this study is relatively simplistic. In the future, it would be beneficial to introduce more physical parameters to enhance its realism.2. The selected map for this experiment is limited to a single 2D map. Currently, the car performs path planning in a relatively simple simulation environment. To address more complex scenarios, it would be valuable to incorporate three-dimensional maps to simulate more intricate fire environments. More applications can be considered in the future work, including: (1) Autonomous Driving and Autonomous Vehicles. RTAB-Map can be used in the field of autonomous driving and autonomous vehicles for map building and localization, assisting vehicles in perceiving their environment and planning optimal paths. It can combine sensor data from cameras, lidar, and other sensors to model and perceive the vehicle's surroundings. (2) Industrial Automation and Machine Vision. RTAB-Map can be applied in machine vision applications within industrial automation systems, such as object detection, pose estimation, and object tracking. It helps facilitate automated production and assembly processes.

4. Conclusion

Through the reasonable optimal planning of firefighting paths, it is crucial to improve the efficiency of firefighting, which can win valuable time for saving lives and property losses. In this paper, we propose a firefighting path planning algorithm based on SLAM. Specifically, this paper firstly introduces in detail the basic theory and key steps of RGBD-SLAM to construct a fixed map and carry out navigation. Simulation experiments for factories and shopping malls verify the effectiveness of the proposed method, which can provide new insights for the study of rapid-fire fighting path planning. At present, the research has only progressed to the computer simulation stage, and in the future, the algorithm can be combined with hardware devices to create practical products that can be truly applied to fire evacuation.

References

[1] Lee J, Park H, Kim Y, et al. Multi-Level Indoor Path Planning and Clearance-Based Path Optimization for Search and Rescue Operations [J]. IEEE Access, 2023.

[2] Chen C, Ma Y, Lv J, et al. OL-SLAM: A Robust and Versatile System of Object Localization and SLAM [J]. Sensors, 2023, 23(2): 801.

[3] Labbe M, Michaud F. Appearance-based loop closure detection for online large-scale and long-term operation[J]. IEEE Transactions on Robotics, 2013, 29(3): 734-745.

[4] Abbyasov B, Lavrenov R, Zakiev A, et al. Automatic tool for gazebo world construction: from a grayscale image to a 3d solid model [C]//2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2020: 7226-7232.

[5] Cheng J, Wang Z, Zhou H, et al. DM-SLAM: A feature-based SLAM system for rigid dynamic scenes [J]. ISPRS International Journal of Geo-Information, 2020, 9(4): 202.

[6] Zheng S, Wang J, Rizos C, et al. Simultaneous Localization and Mapping (SLAM) for Autonomous Driving: Concept and Analysis[J]. Remote Sensing, 2023, 15(4): 1156.

[7] Wang X, Fan X, Shi P, et al. An Overview of Key SLAM Technologies for Underwater Scenes [J]. Remote Sensing, 2023, 15(10): 2496.

[8] Ragot N, Khemmar R, Pokala A, et al. Benchmark of visual slam algorithms: Orb-slam2 vs rtab-map[C]//2019 Eighth International Conference on Emerging Security Technologies (EST). IEEE, 2019: 1-6.

[9] 0Zheng S, Wang J, Rizos C, et al. Simultaneous Localization and Mapping (SLAM) for Autonomous Driving: Concept and Analysis [J]. Remote Sensing, 2023, 15(4): 1156

[10] Chung C M, Tseng Y C, Hsu Y C, et al. Orbeez-slam: A real-time monocular visual slam with orb features and nerf-realized mapping [C]//2023 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2023: 9400-9406.