Comparison of SLAM Mapping Algorithms and Possible Navigational Strategies on Simulated and Real World Conditions

Xinyi Wang 12010239, Jianxiang Ni 11812723, Baoyi Huang 12010437,
Yujie Zhang 12010142, Wanghongjie Qiu 12010310

***Abstract*—Simultaneous Localization and Mapping (SLAM) problem has been an active area of research in robotics for more than a decade, and a lot of researchers have spent a great effort in developing new series of SLAM algorithms. In order to compare the difference between different SLAM algorithms for building maps under virtual and realistic conditions, we made the gazebo world based on the real experimental space. We use a four-wheeled cart with LIDAR to build maps with different SLAM algorithms and compare the final results.**

***Index Terms*—SLAM, Reinforced Learning, Point Cloud, Mapping, Localization, Trajectory Planning.**

I. Introduction

SLAM(Simultaneous localization and mapping)is one of the common approaches to solving the problem of localization and map construction. With SLAM technology, robots can explore and navigate autonomously in unknown environments, underpinning successful robot deployment in many application domains such as urban search and rescue, underground mining, underwater surveillance, and planetary exploration. [2] Currently, many SLAM algorithms exist in robotics: Cartographer, Karto, LIO-SAM, LOAM series, ORB-SLAM3, VINS-Fusion, and so on. In this project, we try to use the different map construction algorithms in virtual and realistic situations and compare the differences between the maps built by both. We built a gazebo world base on real-world experimental environments where TIANRACER building maps with Gmapping algorithm. In the simulation environment, we apply Gmapping, Hector, and Cartographer algorithms. At last, we compare the performance in the simulation environment and the real world. We found that:

(1)In terms of robot motion speed, Gmapping has a smaller limitation and a wider applicability range compared to Hector and Cartographer.

(2)If the motion of the robot is not taken into consideration, or in scenarios where high demands on the robot's motion speed are not required, Hector performs the best among the three algorithms confirmed by direct observations and quantitative error calculations.

(3)The potential of Cartographer remains to be investigated. It generally exhibits better root mean square error performance compared to Gmapping. However, there are also more limitations on the robot's movement speed.

(4)The map built in the real world is not as effective as the simulation environment, the reason is that glass walls in the environment are not well detected by laser radar.

II. Related Work

The SLAM community has made astonishing progress over the last 30 years, enabling large-scale real-world applications, and witnessing a steady transition of this technology to industry.[1]SLAM originated from efforts to formalize the production of topographic maps from aerial imagery. A seminal work in SLAM is the research of R.C. Smith and P. Cheeseman on the representation and estimation of spatial uncertainty in 1986.[8][7] Other pioneering work in this field was conducted by the research group of Hugh F. Durrant-Whyte in the early 1990s.[6] which showed that solutions to SLAM exist in the infinite data limit. This finding motivates the search for algorithms that are computationally tractable and approximate the solution. The acronym SLAM was coined within the paper, "Localization of Autonomous Guided Vehicles" which first appeared in ISR in 1995.[4] Three essential properties of the estimation theoretical solution to the SLAM problem in a linear Gaussian setting [3] was demonstrated in 2001. In the twenty-first century, the availability of large memory spaces has made it possible to store all observations gathered by the robot and adopt a nonlinear optimization framework to solve the SLAM problem. In this scenario, an objective function based on Maximum Likelihood (ML) is optimized [5], leading to more robust and consistent SLAM solutions.[2]

The self-driving STANLEY and JUNIOR cars, led by Sebastian Thrun, won the DARPA Grand Challenge, came second in the DARPA Urban Challenge in the 2000s, and included SLAM systems, bringing SLAM to worldwide attention. Mass-market SLAM implementations can now be found in consumer robot vacuum cleaners and virtual reality headsets such as the Meta Quest 2 and PICO 4 for markerless inside-out tracking.

III. method

Our work can be divided into two parts, the simulation and the experiment in the real world. Focus on the mapping task, we compared the performance of three different mapping algorithms, Gmapping, Hector, and Cartographer in the simulation environment. Then we deploy the Gmapping algorithm, which

has better conformance, on the manual vehicle to compare the performance in the simulation environment and real-world.

A. Mapping in a simulation environment

ROS[6] is an open-source software platform designed for programming and controlling robots. It provides libraries, programming tools, graphical tools, direct control communication with hardware, and sensor/data retrieval libraries.

By offering a common interface for operating the robot's hardware, ROS eliminates the need for developers to worry about specific hardware APIs. This shift in focus makes robot software development significantly easier, more independent, and more adaptable.

Within the ROS ecosystem, numerous SLAM techniques have been developed and are extensively utilized in research and industry settings. Notable examples of these techniques include HectorSLAM, Gmapping, Cartographer, LagoSLAM, and CoreSLAM. Among these, HectorSLAM, Gmapping, and Cartographer have demonstrated superior performance compared to other alternatives, particularly in indoor environments.

1) GmappingSLAM: Gmapping, a widely used SLAM package developed by Grisetti, employs the Rao-Blackwellized particle filter SLAM approach and is recognized as one of the most popular SLAM packages available. It utilizes laser-based methods for mapping and localization. To ensure accuracy, Gmapping is typically recommended to be used in conjunction with precise odometry data, such as wheel odometry.

Gmapping employs a Particle Filter (PF) technique for model-based estimation. In SLAM, the goal is to simultaneously build a map of the environment and determine the robot's position and orientation within that map. The PF algorithm is a probabilistic method that represents the robot's pose and the map as a set of particles. Each particle carries a hypothesis of the robot's pose and the environment's structure.

2) HectorSLAM: HectorSLAM is a SLAM method that utilizes the extended Kalman filter (EKF) [14] for mapping and localization. It leverages the high update rate and low noise output of a high-end LiDAR to accurately predict the robot's movements in real time. While odometry data may not be necessary, the IMU can be utilized for 3-D state estimation. However, it is important to note that HectorSLAM may encounter challenges when used with low-end LiDARs due to the typically noisy and slow update rate of their output.

HectorSLAM relies on a range sensor, typically a laser scanner, to gather information about the environment. The algorithm utilizes a probabilistic grid-based mapping approach known as Occupancy Grid Mapping. It divides the environment into a grid of cells and assigns probabilities to each cell representing the likelihood of occupancy. This grid-based representation allows the algorithm to model the environment and update the occupancy probabilities as new sensor measurements are obtained.

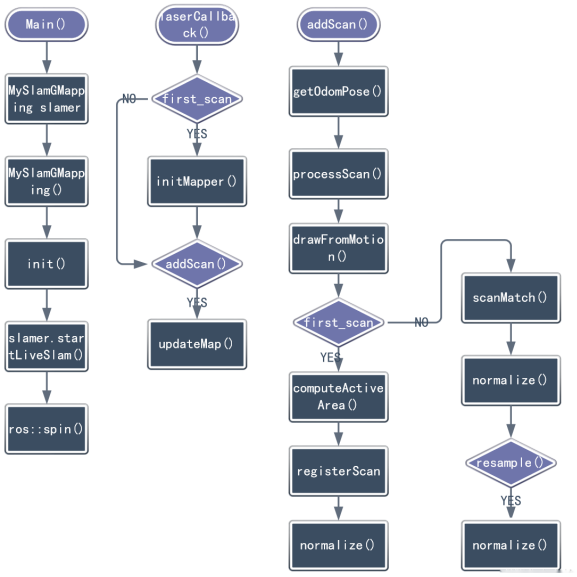

Fig. 1. GmappingSLAM process

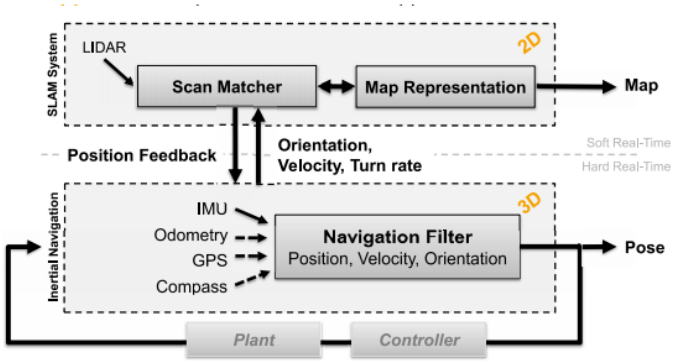

Fig. 2. HectorSLAM process

3) Cartographer: Cartographer, a graph-based SLAM system introduced in 2016, employs two subsystems: global SLAM and local SLAM. It reconstructs the environment by utilizing laser scanner data without requiring odometry information in the simplest 2-D mapping scenario. It is important to note that this technique can consume significant computational resources.

Cartographer utilizes a probabilistic approach called Graph-Based SLAM to solve the SLAM problem. It models the environment as a graph, where nodes represent robot poses, and edges represent the constraints between poses. The algorithm estimates the most likely trajectory of the robot and simultaneously builds a map of the environment.

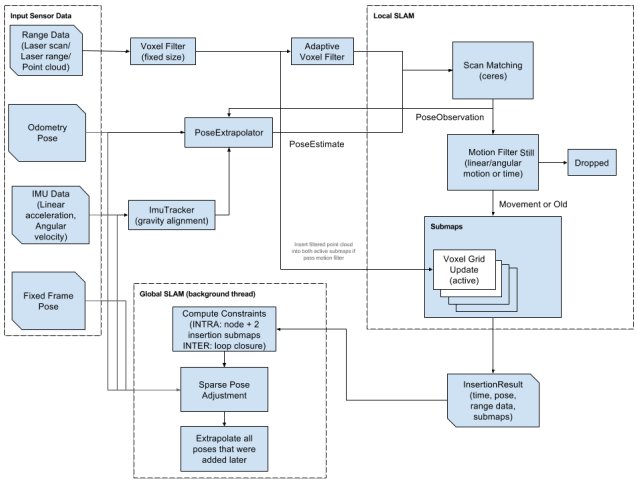

Fig. 3. CartographerSLAM process

The performance and capabilities of these SLAM libraries can vary depending on the specific application and the characteristics of the environment. Each library has its strengths and weaknesses, and the choice of which one to use depends on factors such as sensor configuration, computational resources, and the desired level of map quality and accuracy.

In our work (detailed description in the following experimental section), we applied these three algorithms separately to the indoor mapping of a limo car. Under the same conditions in a self-built Gazebo environment, we varied the car's speed and observed the final map results, the CPU usage during program execution, and the root mean square error (RMSE) by setting sampling points on the original map to quantify the indoor mapping accuracy of the three algorithms.

Since the size of the self-built simulated map in the Gazebo and the characteristics of each obstacle and wall are known, wall corner points (the right angle formed by the intersection of two walls) and fixed obstacle positions (location points of the obstacles) were selected as sampling points. A total of 35 sampling points were selected to compare the errors between the maps generated by the three SLAM algorithms and the actual map. This evaluation was conducted to assess map quality when it is difficult to discern the mapping effect with the naked eye. These 35 sampling points include the four corner points of the indoor boundaries, a total of 24 corner points from six compartments (4*6=24), two corner points near the birthplace of the Limo car obstacles, four corner points of the fixed physical bookshelf obstacle along the map's central axis, and one position of the fixed physical mailbox obstacle along the map's central axis. Two different types of obstacles (a movable car model and a soda can) were not considered in the selection of sampling points since they are not fixed and were placed on the map solely to test the recognition effectiveness of the three algorithms for movable objects and small objects.

Although the three algorithms differ in their mapping approaches, starting point positions, and the variations in errors between the maps and ground truth, there is one aspect that can be determined – the initial position (birthplace) of the Limo car in the simulated environment. By taking the birthplace of the car in the simulated environment as the coordinate origin, a world coordinate system is established for map construction. Therefore, regardless of whether it is in the actual map or in the maps generated by the SLAM algorithms, the positions of the 35 corner points can be determined by specific x-axis and y-axis coordinates. The coordinates of the 35 sampling points in the actual map are marked to obtain a two-dimensional point set representing the actual map. Similarly, the corresponding positions of the 35 corner points and obstacle points generated by the three algorithms are marked to obtain a two-dimensional point set representing the map to be evaluated. The root means square error formula is then used to quantitatively evaluate the mapping accuracy of each algorithm under different environmental variables.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(d_i - e_i)^2} \tag{1}$$

where n is the number of sampled points, d is the coordinate estimated by the SLAM approach, and e is the true coordinate known before mapping.

B. Mapping task in real world

In the process of the Mapping task, the unmanned vehicle platform used is TIANRACER, as shown in Fig.4. The vehicle has a Laser Radar, an RGB Camera, and a HUAWEI Atlas 500 computing platform, which uses a Hayes Hi3559A processor. The Laser Radar is RPLIDAR A1, which has a measuring range of 12m radius and a measuring accuracy of 1 degree at a scanning frequency of 5.5 Hz. The length of the vehicle is 380mm and the width of the car is 210mm.

In the manual vehicle, the Gmapping algorithm was deployed to perform a practical automatic mapping task. The result can reflect the performance of the algorithm in the real world. By comparing the result of the simulation and the experiment in the real world, the difference between the simulation and the real world can be figured out.

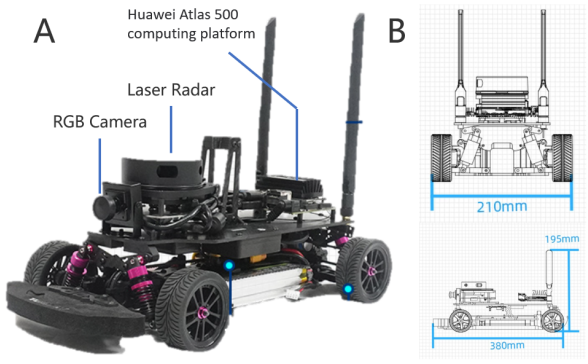

Fig. 4. Hardware introduction: (A)the Sensors and computing platforms, including a Laser Radar, an RGB camera, and a Huawei Atlas 500 computing platform (B)the parameter of the vehicle, the length, height, and width.

C. Path navigation under reinforcement learning

At present, dynamic programming algorithm is often used in path planning. However, it requires high computational power and is difficult to converge when applied in complex environment.We try to propose possible path planning strategies based on reinforcement learning for better performance.In this section, We compare the potential of three kinds of reinforcement learning for solving path planning problems through a simple path planning task.

1) navigation based on Monte Carlo method: The provided images illustrate the path planning in reinforcement learning using Monte Carlo methods for different iteration counts: 5000, 10000, and 50000. It is clearly noticeable that as the number of iterations increases, the robot's planning steps gradually decrease, resulting in a more streamlined and efficient movement pattern that approximates a straight line.

Monte Carlo methods, while known for their slower convergence, exhibit remarkable algorithm stability. This stability enhances their ability to identify and adopt optimal strategies for task completion. By iteratively refining the learned policy, Monte Carlo methods showcase their resilience in finding effective ways to accomplish tasks, even in complex scenarios.

These visual representations of the Monte Carlo method's progressive path planning highlight its ability to converge towards efficient and goal-oriented behavior, reinforcing its effectiveness and robustness as a reinforcement learning approach.

2) navigation based on SARSA method: The provided figure depicts path planning in reinforcement learning using SARSA (State-Action-Reward-State-Action) method for different iteration counts: 5000, 10000, and 50000. It is evident that as the number of iterations increases, the robot's planning steps progressively increase, showcasing its heightened sensitivity to environmental changes and decision-making.

Under SARSA, the robot demonstrates a higher level of adaptability, reacting promptly to variations in the environment. This increased responsiveness allows for more continuous and smoother decision-making, resulting in a more cohesive movement pattern that approximates a straight line toward the obstacle. The fit between the learned policy and the environment is noticeably improved.

These visual representations highlight the effectiveness of SARSA in path planning. With increasing iterations, SARSA demonstrates an enhanced ability to respond to dynamic environments while maintaining a consistent directionality toward the obstacle. This overall improvement in convergence and decision-making quality reinforces the proficiency of SARSA as a reinforcement learning method for optimizing path-planning tasks.

3) navigation based on Q-learning method: Under the Q-learning algorithm for path navigation, as the number of learning iterations increases, the number of decision-making instances also increases. However, the continuity of decision-making tends to decrease. The influence of environmental disturbances becomes more prominent, and the final approach often involves a turning maneuver.

Q-learning explores different state-action pairs to optimize its Q-values, which can lead to less consistent decision-making in terms of movement direction. The algorithm's focus on maximizing rewards in each state can result in more responsive behavior to immediate rewards or environmental factors, but it may sacrifice smoothness and continuity in the overall path.

Despite the reduced continuity, Q-learning demonstrates its effectiveness in capturing the overall impact of the environment. It is capable of adapting to various disturbances and finding suitable paths while considering the long-term cumulative rewards. The final approach, which often involves turning, suggests that Q-learning is sensitive to the layout and configuration of the environment.

Overall, Q-learning strikes a balance between exploring different options and exploiting the learned knowledge, enabling it to navigate the environment effectively while adapting to various challenges.

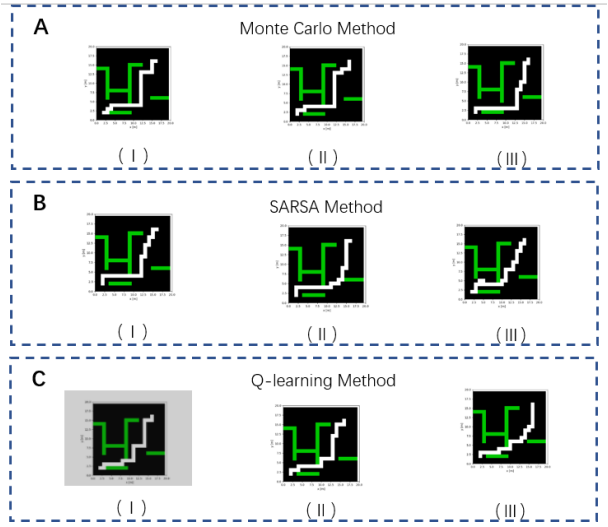

Fig. 5. Results of three reinforce learning method:(A)using Monte Carlo method for different iteration counts:(I)5000;(II)10000;(III)50000; (B)using SARSA Method for different iteration counts: (I)5000;(II)10000;(III)50000; (C)using Q-learning method for different iteration counts: (I)5000;(II)10000;(III)50000;

IV. EXPERIMENT

We do our experiment task in both the simulation environment and the real world.

A. Mapping in a simulation environment

1) Experiment setup: simulation environment in Gazebo: Gazebo is a widely used open-source robotics simulation software that provides a robust and flexible platform for simulating and testing robotic systems. It is designed to create realistic and dynamic environments where robots can be virtually operated and evaluated.

Gazebo offers a range of features that make it a powerful tool for robotic simulation. It provides a physics engine that accurately models the dynamics of objects, allowing for realistic interactions between robots, environments, and various physical entities. This enables users to simulate robot

movements, sensor readings, and even complex interactions with objects and environments.

In this experiment, we manually constructed an indoor environment in Gazebo to provide space for the movement of the car. The indoor environment has a length of 7 meters and a width of 5 meters. It consists of three horizontal and three vertical corridors, along with five types of physical obstacles. Among them, there are three fixed obstacle entities and two movable obstacle entities. The fixed obstacle entities include a bookshelf, a mailbox, and a human-shaped door, which provide seven sampling points for evaluating the completed map construction.

The robot car will move or turn in the Gazebo 3D environment using four different combinations of linear and angular velocities. In each mapping experiment, the car will follow a fixed route in the simulated environment to eliminate the influence of different paths on SLAM mapping.

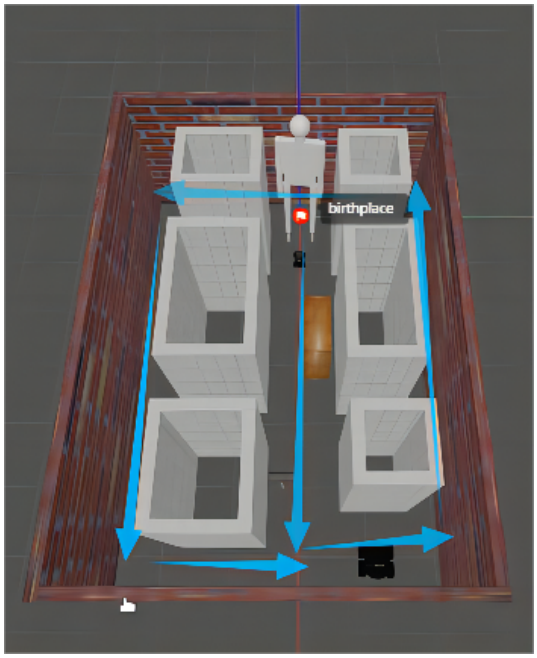

Fig. 6. The simulation environment in Gazebo and the specific route

The 35 sample points are shown below. The RED FLAG symbolizes the birthplace of the limo car, the RED TICK represents the sample points from the obstacles, and the GREEN TICK represents the sample points from the walls.

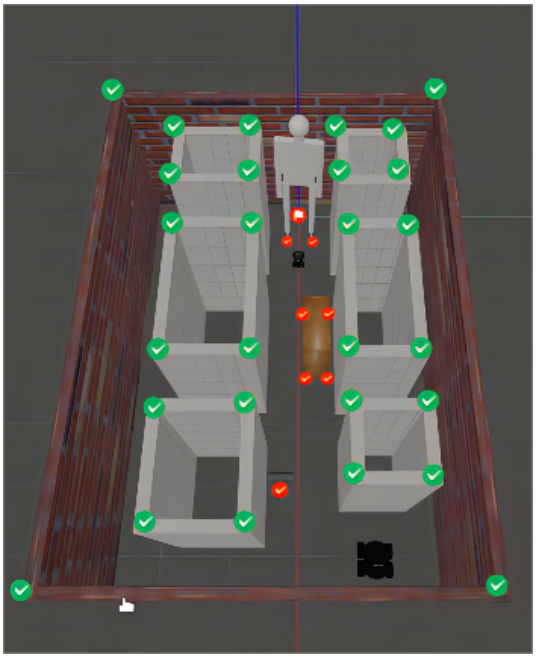

Fig. 7. 35 sample points in total

2) experiment results: For each SLAM algorithm, as at least four different car velocities were set, the number of experiments was at least four times. A total of twelve sets of two-dimensional maps were obtained, including successful and unsuccessful cases.

If only observed with the naked eye, during the mapping process using SLAM, the mapping results of the first three velocity groups were relatively stable. However, when the car velocity increased to 0.8mps/1.6radps, a significant drift phenomenon appeared, which typically resulted in a significant increase in the root mean square error.

If we only rely on visual observation, in the process of using the Hector mapping algorithm, the mapping results for the first three velocity sets are relatively stable, and it can be observed that the mapping is more accurate and visually appealing compared to Gmapping. However, when the car velocity is increased to 0.8 m/s and 1.6 rad/s, the drift phenomenon becomes much more severe compared to Gmapping, to the extent that it becomes impossible to distinguish the corresponding positions of the sampling points and calculate the root mean square error, resulting in mapping failure.

If observed only with the naked eye, the mapping results of the Cartographer algorithm appear to be on par with Hector, and even clearer. However, the mapping performance of Cartographer is significantly more affected by the car's velocity compared to Hector. Mapping failures were already encountered when the car's velocity was increased to 0.4 m/s and 0.8 rad/s.

The errors quantitatively calculated using the root mean square formula roughly correspond to the results obtained through observation alone. Compared to Cartographer and Hector, although Gmapping generally constructs coarser maps, it is more suitable for scenarios that require certain constraints

on the robot's motion speed. In our experiments, Gmapping did not exhibit any mapping failures. On the other hand, Hector showed excellent mapping performance when the robot's motion speed was not very fast, achieving the smallest root mean square error of 0.2174 in the experiment where the robot moved at a low speed. Cartographer has stricter requirements on the applicable scenarios and the robot's motion speed compared to Hector. When the robot's motion speed remains at a lower level, the two-dimensional maps constructed using Cartographer fall between the results obtained with Gmapping and Hector.

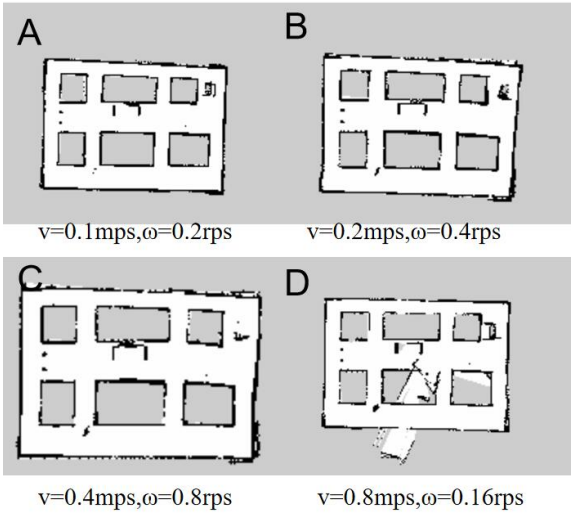

Fig. 8. GmappingSLAM, with the speed of the car (A) linear speed = 0.1 mps, angular speed = 0.2 rps, (B) linear speed = 0.2 mps, angular speed = 0.4 rps, (C) linear speed = 0.4 mps, angular speed = 0.8 rps, (D) linear speed = 0.8 mps, angular speed = 1.6 rps

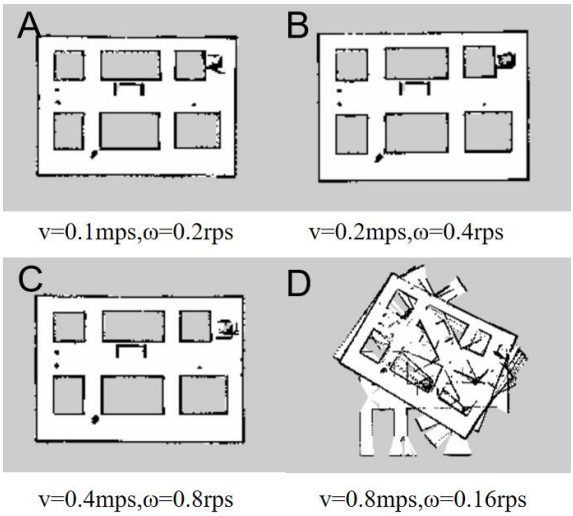

Fig. 9. HectorSLAM, with the speed of the car (A) linear speed = 0.1 mps, angular speed = 0.2 rps, (B) linear speed = 0.2 mps, angular speed = 0.4 rps, (C) linear speed = 0.4 mps, angular speed = 0.8 rps, (D) linear speed = 0.8 mps, angular speed = 1.6 rps

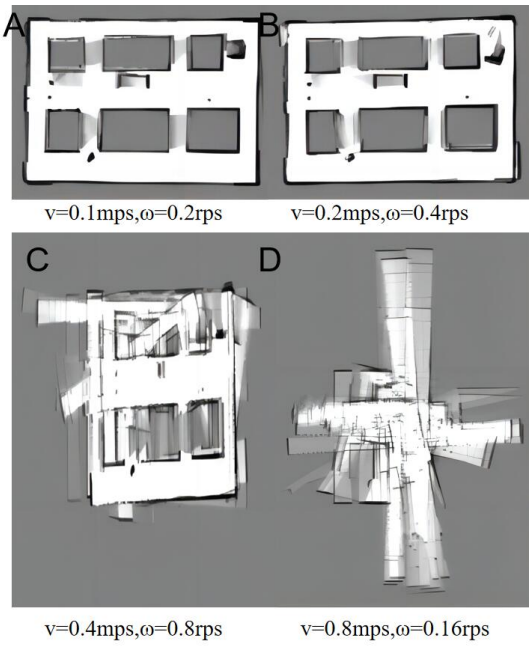

Fig. 10. Cartographer, with the speed of the car (A) linear speed = 0.1 mps, angular speed = 0.2 rps, (B) linear speed = 0.2 mps, angular speed = 0.4 rps, (C) linear speed = 0.4 mps, angular speed = 0.8 rps, (D) linear speed = 0.8 mps, angular speed = 1.6 rps

RMSE	Speed 1 time	Speed 2 times	Speed 4 times	Speed 8 Times
Gmapping	0.3742	0.4219	0.6688	0.7831
Hector	0.2174	0.2311	0.2289	Mapping failed
Cartographer	0.2264	0.2494	Mapping failed	Mapping failed

Fig. 11. Table: ACCURACY COMPARISON COMPLEX OFFICE

During the execution of the mapping program, the 'top' command can be used to directly observe the real-time changes in CPU usage in the command line window. Since the CPU usage of the Cartographer algorithm is significantly higher than the other two algorithms, the CPU data of the Hector algorithm and Gmapping algorithm were projected relative to the CPU usage of the Cartographer algorithm, obtaining a comparison of the CPU usage among the three algorithms.

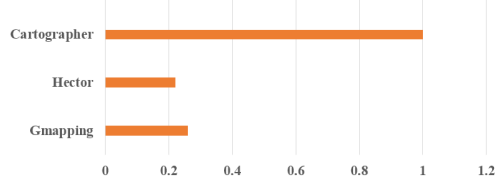

Fig. 12. CPU usage—normalized to Cartographer's value

3) Results analyze: Firstly, in terms of robot motion speed, Gmapping has a smaller limitation and a wider applicability range. Although a noticeable drift phenomenon can still be observed, compared to the mapping failures of Hector and Cartographer, Gmapping demonstrates a significantly better mapping performance.

Secondly, if the motion of the robot is not taken into consideration, or in scenarios where high demands on the robot's motion speed are not required, Hector is the optimal choice among these three mapping algorithms. Both direct observations and quantitative error calculations confirm the excellent performance of Hector in mapping when the robot is moving at low speeds.

Finally, the potential of the Cartographer remains to be investigated. Although it generally exhibits better root mean square error performance than Gmapping, its applicability is narrower than Hector's. To fully leverage its capabilities, there are also more limitations on the robot's movement speed.

B. Mapping in real-world

Considering that the Vehicle has a considerable radius of a turning circle, the experiment was launched on the second floor of the First scientific research Building. In this experiment, the scan matching algorithm, one of the most basic location algorithms, is applied. And focus on the mapping task, the Gmapping algorithm is used since it has better performance in the simulation environment. During the experiment, As shown in Fig.13, the equality of the mapping is not so good as in the simulation environment. The difference in complexity between the simulation environment and the realistic condition leads to this kind of result. Part of the reason is that the environment's glass walls are not well detected by laser radar. This problem would be solved by using multi—mode method to combine the radar and camera. Meanwhile, unenclosed areas bring difficulties to the mapping.

Fig. 13. experiment result of the mapping task in the real world

V. Conclusion

In this article, we implemented the real-time mapping and positioning of the vehicle in simulation and the real world.

At the simulation level, we built a simulation environment in the gazebo and captured its environmental data using Lidar and depth camera as data input for mapping and localization. The output point cloud data and mapping data can be obtained through three different SLAM algorithms to realize localization and mapping. After several experiments, we chose the SLAM algorithm that performed best in the simulated experiments for real-world experiments.

In the real-world section, the vehicle named TIANRACER is equipped with a variety of sensors and lidar to acquire information in the experimental environment. We choose the Gmapping algorithms for mapping and localization. Although The map-building results were not satisfactory, we speculate that this is because a Single sensor has limitations on obstacle recognition. We will add a depth camera to capture more information to optimize the process.

The current reinforcement learning progress can achieve the trajectory planning of the robot at the simulation level. In the future, we will promote the research to realize the real-time mapping and trajectory planning of the vehicle in the real world.

References

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J.J. Leonard. Past, present, and future of simultaneous localization and mapping: Towards the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[2] Gamini Dissanayake, Shoudong Huang, Zhan Wang, and Ravindra Ranasinghe. A review of recent developments in simultaneous localization and mapping. *2011 6th International Conference on Industrial and Information Systems*, pages 477–482, 2011. doi: 10.1109/ICIINFS.2011.6038117.

[3] M.W.M.G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241, 2001. doi: 10.1109/70.938381.

[4] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, 13(2):99–110, 2006. doi: 10.1109/MRA.2006.1638022.

[5] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. isam: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, 2008. doi: 10.1109/TRO.2008.2006706.

[6] J.J. Leonard and H.F. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. *Proceedings IROS '91:IEEE/RSJ International Workshop on Intelligent Robots and Systems '91*, pages 1442–1447 vol.3, 1991. doi: 10.1109/IROS.1991.174711.

[7] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, 4: 850–850, 1987. doi: 10.1109/ROBOT.1987.1087846.

[8] Randall C. Smith and Peter C. Cheeseman. On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research*, 5:56 – 68, 1986.