Research on Global Map Construction and Location of Intelligent Vehicles Based on Lidar

Biyao Wang, Yi Han(✉), and Jing Jin

School of Automobile, Chang'an University, Xi'an 710064, China

Abstract. The research and application of lidar in the establishment of high-precision maps and path planning of intelligent driving vehicles are being developed as a technical support for intelligent driving technology. In this paper we mainly focus on the research and application of lidar real-time positioning and map building technology in intelligent driving vehicles, including the preprocessing of raw data required by Simultaneous Localization and Mapping (SLAM), the front-end algorithm of SLAM, the back-end optimization of SLAM and the final algorithm experiment. In the front-end algorithm of SLAM, we propose a boundary line feature extraction scheme based on the traditional curvature feature point extraction method and Random Sample Consensus (RANSAC) algorithm, and realize point cloud registration and pose estimation based on optimized Iterative Closest Point (ICP). In the back-end processing algorithm, we propose a sparse algorithm of graph nodes for selection, and propose boundary verification based on an improved loop detection and relocation strategy design. The results showed that the algorithm in this paper was less affected by noise, which considerably saves time in the operation of the loop frame determination algorithm and improves the safety of the intelligent vehicle in the process of driving.

Keywords: Intelligent driving car · Lidar · Real time positioning · Map building

1 Introduction

Currently, with the development of science and technology, the improvement and pursuit of vehicle performance, intelligence, safety and comfort, and transportation efficiency are constantly changing and strengthening [1]. People's pursuit and exploration of vehicle intelligence have become the driving force for the development of the intelligent driving industry. Among them, safety, efficiency and economy are driving the development of intelligent driving [2].

The construction of high-precision maps and the acquisition of real-time accurate positions are important contents of unmanned vehicle environmental perception, and

Supported by the National Key R&D Program of China under Grant 2018YFB1105304, and National Natural Science Foundation of China under Grant U1664264, and National Natural Science Foundation of China under Grant U1864204.

© Springer Nature Switzerland AG 2021
J. Cheng et al. (Eds.): CSS 2020, LNCS 12653, pp. 210–224, 2021.
https://doi.org/10.1007/978-3-030-73671-2_19

have an important impact on the path planning and decision-making control of unmanned vehicles. At present, the most widely used sensors in the environmental perception of intelligent driving vehicles are laser sensors, GPS and visual sensors. Depending on different application environments [3, 4], the selection of sensors has varying emphases; however, lidar is a necessary sensor for all types of intelligent driving vehicles. The main reason is that other sensors have their own defects to some extent, and they cannot independently undertake the environmental perception of intelligent driving vehicles.

In this paper we summarize the problems of intelligent vehicle navigation based on GPS and vision Simultaneous Localization and Mapping (SLAM) technology [5, 6]. (1) Navigation systems based on GPS have limited use scenarios, and signal loss is easily caused as a result of blockage. In addition, the GPS signal is unstable. Therefore, the application of this navigation scheme in intelligent driving vehicles has serious drawbacks and security problems. (2) The SLAM technology, which is based on vision, is greatly affected by light and weather. For example, in rainy days, foggy days, and strong and weak light conditions, the sensors' perception of the environment is affected, and there are also serious security problems. Compared with the vision sensor, the lidar sensor has fast response speed, strong anti-interference ability and high data accuracy, which solves the problems inherent to GPS and camera technology [7].

2 Related Work

At the present stage, generally speaking, the sensors used in lidar SLAM mainly include an inertial measurement unit, a wheel odometer and lidar. The map types constructed include a coverage grid map and a point cloud map. The coverage grid map has clearer divisions and annotation of the environment and is widely used. According to the assumption of the mapping environment, there are two types of SLAM technology: one is a dynamic environment, which means that there are moving objects (high dynamic objects) in the lidar field of vision, or that the position difference of the same object changes (low dynamic object) when lidar is observed twice; the other is a static environment, which is the type most SLAM algorithms widely use at present. In the latter, it is assumed that the environment is static in the process of mapping and navigation [8, 9]. On the basis of the static environment assumption, the environment is either divided into a scale map, a topological map or a hybrid map according to the output process. At present, almost all SLAMs are scale maps.

On the basis of a scale map, there are different branches varying in their main principles. At present, SLAM based on filters and SLAM based on graph optimization are widely used.

2.1 SLAM Based on Filters

According to the different filters, the SLAM method based on filters is divided into SLAM based on a Kalman filter and SLAM based on a particle filter. However, the main basis of the two filters is the same, and the main principle is recursive Bayesian state estimation. In order to avoid the problem of feature point definition and corresponding data association to a certain extent, the FastSLAM algorithm, as one of the most representative algorithms

based on a particle filter SLAM algorithm, has become a hot topic of scholars at home and abroad. Therefore, the algorithm has been constantly improved by different scholars, which makes the algorithm capable of achieving good results in specific scenarios.

The core idea of the SLAM algorithm based on a recursive Bayesian estimation is to use the motion model and observation model of intelligent driving vehicles to forecast and observe the state of vehicles under the assumption of a hidden Markov model. Although this method has the advantages of online real-time map updating, due to its own characteristics, it inevitably has its disadvantages. Because the method of map construction is incremental, and sensors errors and other aspects of the vehicle are constantly accumulating during the driving process; this cumulative error will lead to inconsistency of the constructed map when the vehicle driving environment is large. Therefore, the application of SLAM based on recursive Bayesian state estimation in large-scale environments is not ideal.

2.2 SLAM Based on Graph Optimization

With the widening application of SLAM technology, its application scenarios are also growing. In order to adapt to large-scale application scenarios, more and more scholars have carried out research on SLAM based on graph optimization. SLAM based on graph optimization can correct the trajectory of intelligent driving vehicles through global optimization and build a more realistic map. At present, SLAM based on graph optimization is the most widely used and researched method, with good effect. This SLAM method can have absolute position information and can recover to a better state from the error accumulation process. Through the maximum likelihood estimation of the position and attitude, the local minimum value is non convex optimization, so as to obtain the position and attitude information of the vehicle at a certain time in the future.

Grisetti G et al. [11] proposed a solution based on least square error, and proposed optimization using the framework of SLAM problem W et al. [12] and others proposed to use the branch binding method to effectively solve the mapping problem of subgraphs, and has good loopback ability. It has high accuracy in building maps, and can solve slam problems in large environment. In 2016, vysotska et al. [13] and others will build new subgraphs, public maps and data from laser sensors So as to improve the drawing effect. Most of these graph optimization SLAM algorithms are basic nonlinear optimization problems, and the least squares method is also the most commonly used algorithm framework [14, 15].

Based on the lidar, we carry out the research and application of real-time positioning and map building technology in intelligent driving vehicles, including the preprocessing of raw data required by SLAM, the front-end algorithm of SLAM, the back-end optimization of SLAM and a final algorithm experiment. The main contributions of this paper are as follows: in the front-end algorithm of SLAM, based on the traditional curvature feature point extraction method and Random Sample Consensus (RANSAC) algorithm, a boundary line feature extraction scheme is proposed, and point cloud registration and pose estimation are realized based on the optimized Iterative Closest Point(ICP). In the back-end processing algorithm, the sparse algorithm of graph nodes is proposed to select, and the boundary test is proposed based on the improved loop detection and relocation strategy. Through a series of front-end and back-end algorithm optimization,

the influence of noise on the matching algorithm is reduced, and the time consumed by loop frame determination algorithm is considerably reduced.

3 Design of SLAM Front End Processing Algorithm for Lidar

Due to the low density of lidar remote point cloud data, the effect and accuracy of these point cloud data in point cloud registration and other post-processing are usually not very good [16, 17]. Therefore, this paper sets conditions for the x-axis, y-axis and z-axis to remove the remote data [18]. In this paper, we set the filter to specify the value range from three dimensions, and then check whether a point is in the specified value domain of the specified dimension as the condition to traverse the cloud data, remove the external points and retain the internal points.

Suppose the midpoint P (x, y, z) of the original point cloud, if:

$$x \in (X_1, X_2) \cap y \in (Y_1, Y_2) \cap z \in (Z_1, Z_2) \tag{1}$$

When Eq. (1) is satisfied, the point is retained, otherwise it is deleted.

On the other hand, in order to ensure the processing speed and efficiency of the later point cloud data, the point cloud data must be downsampled in the process of preprocessing the original input point cloud. Considering the effect of down sampling and the fidelity of the original point cloud, the following methods are used to downsample the input point cloud.

The main idea of this method is as follows: firstly, the input 3D point cloud is divided into a series of 3D grids. Assuming that there are n point cloud data in a grid, the center of gravity is as follows:

$$(x, y, z) = \left(\frac{\sum_{i=1}^{n} x_i}{n}, \frac{\sum_{i=1}^{n} y_i}{n}, \frac{\sum_{i=1}^{n} z_i}{n} \right) \tag{2}$$

For each grid point cloud, the center of gravity of these point clouds is approximately replaced, and other point cloud data are removed and the center of gravity points are retained to achieve the purpose of down sampling.

3.1 Feature Point Extraction from Point Cloud Data

At present, the curvature-based feature point extraction method is mainly used. The theory of this method is relatively simple, the calculation speed is fast, and the extraction effect of feature points is relatively ideal. However, this method is sensitive to noise and does not have good robustness. Therefore, on this basis, this paper combines this method with the RANSAC algorithm, and proposes a boundary line feature extraction scheme to ensure the integrity of the method. The feature information of a point cloud is used to prepare for the later point cloud registration. The boundary line feature extraction scheme is as follows:

(1) Taking a grid point cloud Q_{data} as input, the model is set as linear model M, and two points in Q_{data} are randomly selected as sample points to fit model M. The schematic diagram is shown in Fig. 1.

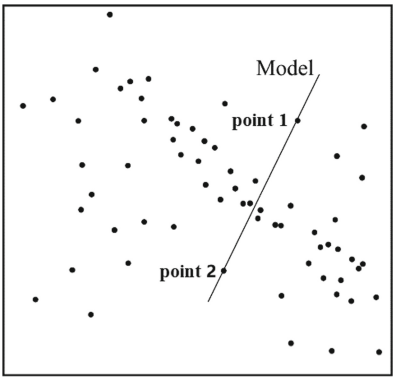


Fig. 1. Schematic diagram of any two points fitting model M.

(2) Because the point cloud data are not strictly in linear distribution, there are some fluctuations in the data points. The tolerance range is set as δ, and then it is found out if the point cloud Q_{data} falls within the tolerance range δ and the number of the points N is counted. Figure 2 is a schematic diagram of the points within the tolerance range.

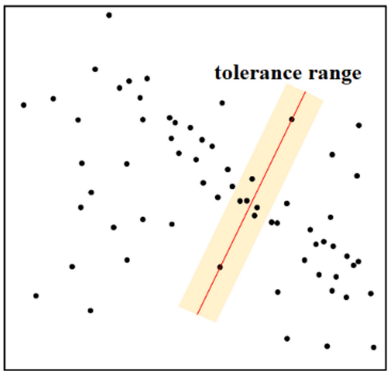


Fig. 2. Schematic diagram of points within tolerance range.

(3) In a point cloud Q_{data}, two points are randomly selected as sample points, and the steps in (1) and (2) are repeated K times. The number of iterations K is determined by the total number of data points N in the point cloud Q_{data} and the number of points within the tolerance range N_{in}. Suppose that the probability of n points out of all N points of point cloud Q_{data} falling within the tolerance range is P_{in}, while the probability of falling outside the tolerance range is P_{out}. Therefore, the probability that at least one of these N points falls outside the tolerance range is $1 - P_{in}$, while $P_{out} = \frac{1-N_{in}}{N}$, and then $(1 - P_{out})^n$ represents the probability that all points in point cloud Q_{data} fall within the tolerance range. Therefore, when a point in point cloud Q_{data} falls outside the tolerance range, the probability is $1 - (1 - P_{out})^n$. Therefore, in the process of K iterations, the probability that each point falls out of the tolerance range is $(1 - (1 - P_{out})^n)^K$, where

K is obtained by:

$$1 - P_{in} = (1 - (1 - P_{out})^n)^K \tag{3}$$

$$K = \frac{\log(1 - P_{in})}{\log(1 - (1 - P_{out})^n)} \tag{4}$$

Figure 3 is the schematic diagram of a certain iteration for reselecting sample points.

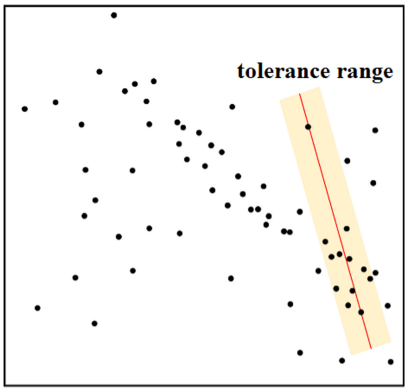


Fig. 3. Schematic diagram of a certain iteration for reselecting sample points.

(4) After each iterative fitting, there will be a corresponding number of local points N_{in} in the tolerance azimuth δ. After iteration, the line model M corresponding to the maximum value of N_{in} is the final extracted line feature. The schematic diagram is shown in Fig. 4.

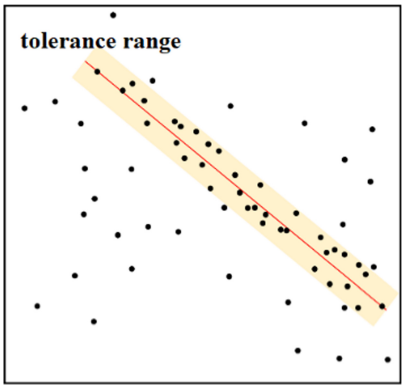


Fig. 4. Schematic diagram of line model corresponding to N_{in} Max in iteration process.

At the same time, according to whether there are point cloud data points in the grid, it can be divided into a real grid and an empty grid. We binarize these two kinds of grids

and define that $f(x, y, z) = 1$ is the real grid and $f(x, y, z) = 0$ is the empty grid. Real grids can be expressed as:

$$g(x, y, z) = \sum_{-1 \le i \le 1} \sum_{-1 \le j \le 1} \sum_{-1 \le k \le 1} |f(x, y, z) - f(x + i, y + j, z + k)| \quad (5)$$

where I, J and K denote the topological directions of the grid, and satisfy the equation $i + j + k = \pm 1 \,\&\, (i = 0 \| j = 0 \| k = 0)$.

At this time, if $g(x, y, z) \ge 3$, we determine the real grid as the boundary grid, and use the RANSAC algorithm to calculate the straight line boundary in the same topological direction in all of the boundary grids. The calculation effect is shown in Fig. 5.

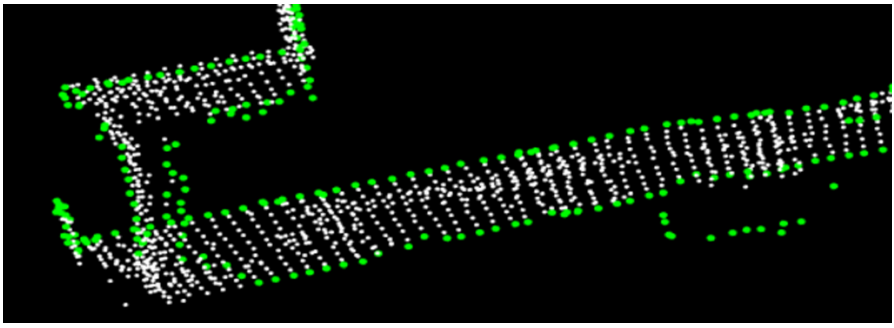


Fig. 5. Effect of boundary line feature extraction.

So far, the feature information has been extracted based on curvature and the RANSAC algorithm. The next step is to filter the feature information.

4 Design of Laser Radar SLAM Back-End Processing Algorithm

4.1 Selection and Extraction Strategy of Graph Nodes

Due to the low speed of intelligent driving vehicles at present, the speed is controlled below 30 km/h in complex environments, and the speed can reach 50–60 km/h only when the environment is simple and the road is straight. When the vehicle speed is low in complex environments, there will be a large amount of redundant data in the front-end of SLAM, while in a flat road environment, only a small amount of position and attitude data is needed. Therefore, in order to effectively simplify the data and reduce the complexity of the back-end optimization algorithm, this paper proposes a sparse algorithm to sample the vehicle trajectory.

The algorithm used in this paper is based on the vertical distance between a point and the line where two points are located, and it sets a threshold to screen the pose points. The specific algorithm process is described as follows.

(1) A straight line is determined according to the starting position and pose point x_1 and the ending position and pose point x_n of the vehicle track.

(2) The position and pose points x_i on the vehicle track are detected one by one; then the distance d_{1i} between them and the straight line x_1x_n is calculated, and then the position and attitude point x_k corresponding to the maximum distance d_{1max} is determined.
(3) Comparing d_{1max} with the threshold value ε, if $d_{1max} < \varepsilon$, it can be considered that the trajectory of this segment is approximately equal to the segment, so it is replaced by this segment.
(4) If $d_{1max} \geq \varepsilon$, the vehicle trajectory is divided into two segments by taking the pose point x_k as the boundary, and the process of steps (1) to (3) is repeated separately for these two segments.
(5) In the same way, all parts of the vehicle track are processed in the same way, and the position and posture points on the track are compressed.

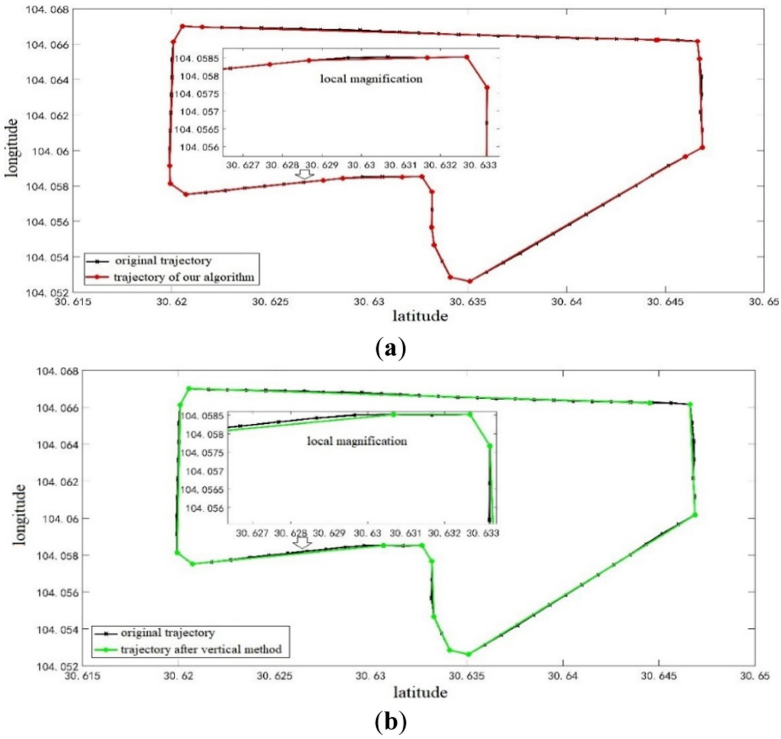


Fig. 6. Comparison of trajectory before and after using two kinds of thinning algorithms: (a) effect of thinning algorithm; (b) vertical pumping effect.

Another widely used thinning algorithm is the vertical distance limit method, but this method cannot control the fidelity of the original trajectory globally. Figure 6 shows the comparison of the two algorithms. It is obvious that the proposed algorithm has better fidelity while compressing the number of pose points.

4.2 Improved Loop Detection and Relocation Strategy Design

The framework of the loop detection method proposed in this paper is mainly composed of the following steps: 1. point cloud preprocessing; 2. feature point extraction; 3. preliminary determination of candidate loop frame; 4. descriptor extraction; 5. loop frame accurate determination; 6. target matching; 7. boundary verification.

This paper proposes boundary validation. When the target matching work is finished, due to the determination of the looping frame and a small amount of matching inaccuracy, in order to ensure the correctness of the whole process, the point cloud in the candidate matching list in (3) is further verified. The specific method is as follows: match the boundary line feature points extracted from two groups of point clouds based on the RANSAC algorithm to obtain the transformation matrix R_v and T_v. If R_m and T_m are consistent with R_v and T_v, it is judged that the loop detection match is consistent, that is, the vehicle returns to the historical position in the track after a period of time, and then a loop detection edge is added to the pose image; otherwise, when the loop frame appears in the case of false detection, the values of R_m, T_m and R_v, T_v will be very different, and the looping frame will be eliminated.

5 Experimental Verification and Result Analysis

The operating platform of the industrial control equipment is a 64 bit Ubuntu 16.04 system. The data processing and algorithm experiments involved in this paper are based on the Robot Operating System (ROS) robot operating system.

5.1 Experimental Analysis of Front End Processing

Feature Point Extraction Experiment

The strategy of feature point extraction in this paper is a curvature-based and RANSAC-based boundary line extraction method. The common curvature based feature point extraction method has fast calculation speed but is sensitive to noise. The reliability of feature points extracted in a complex environment will be greatly reduced. As shown in Fig. 10, the feature points extracted from the point cloud data of the experimental scene based on curvature extraction are shown in Fig. 7.

The red line in the above figure is the part with trees in the experimental scene. The scene of this part is more complex and changeable, which leads to the reliability of the feature points in this part to a certain extent. Therefore, based on this, this paper extracts the boundary line features of the scene based on RANSAC, and the extraction effect is shown in Fig. 8.

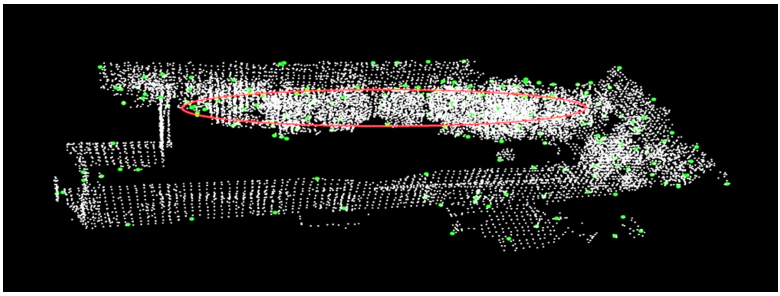


Fig. 7. Feature point extraction based on curvature. (Color figure online)

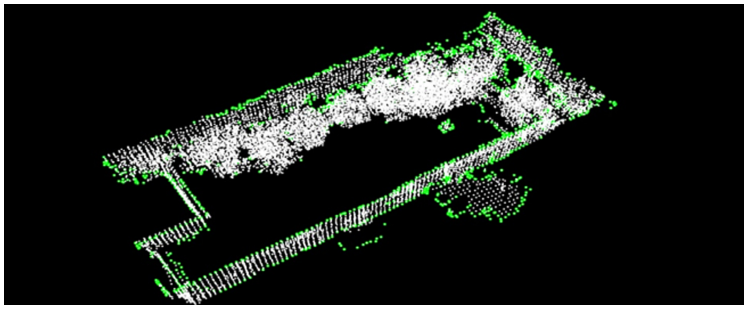


Fig. 8. Boundary line feature extraction based on Random Sample Consensus (RANSAC).

Point Cloud Matching Experiment

In this paper, the point cloud matching is optimized on the basis of the original classical iterative closest point (ICP) algorithm. After optimization, (1) when approaching the minimum value of the objective function, the descent direction of the iteration gradually approaches the Gauss Newton method, which is similar to the second derivative information, so as to effectively improve the convergence speed of the algorithm; (2) when far away from the minimum value of the objective function, the iterative descent method can search the global solution and avoid the local optimal solution.

In order to show the advantages of point cloud matching after optimization, we compared the two matching algorithms before and after optimization in terms of iterative efficiency and robustness to noise in the same experimental scene.

In the comparison experiment of iterative efficiency, this we iterated 10 times, 20 times, 40 times, 80 times, 160 times and 320 times to carry out the point cloud matching experiments of the two methods. The comparison of the time-consuming of the two methods is shown in Fig. 9. It can be seen from the graph that the efficiency of the classical algorithm was greatly improved after optimization, and with the increase of the number of iterations, the efficiency of the algorithm was increasingly different.

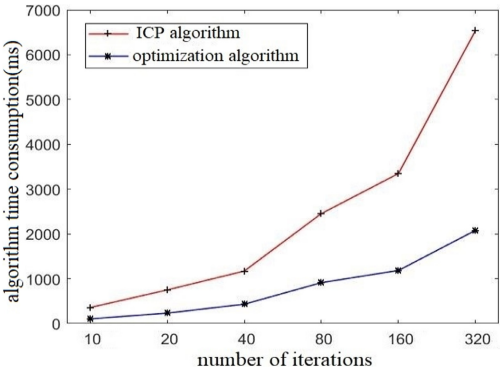


Fig. 9. Time consumption comparison of point cloud matching between classical Iterative Closest Point(ICP) algorithm and our algorithm.

In the contrast experiment of robustness to noise, Gaussian noise was added to the original data of the experimental scene to compare the matching accuracy of the two matching algorithms. The added Gaussian noise was divided into seven levels, from weak to strong. The first level did not add Gaussian noise, and then the standard deviation of each level increased by 0.5 mm until the seventh level. As shown in Fig. 10, the point cloud data with the third level of Gaussian noise were added. Green points are Gaussian noise points, and white points are original data points.

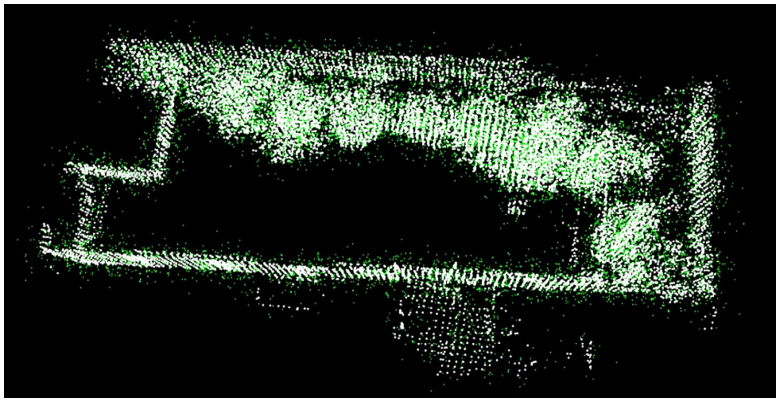


Fig. 10. Point cloud data after adding level 3 Gaussian noise.

The comparison of matching accuracy scores of the two matching algorithms under different levels of Gaussian noise is shown in Fig. 11. It can be clearly seen from the figure that the matching accuracy of the classical ICP algorithm for point cloud matching was significantly reduced when there was too much noise in the environment, while the optimized matching algorithm presented in this paper was less affected by the noise.

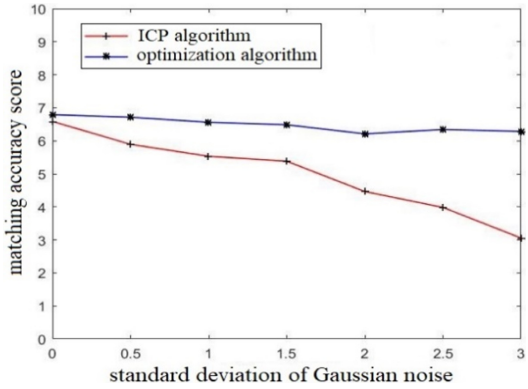


Fig. 11. Comparison of matching accuracy under different levels of Gaussian noise.

5.2 Back-End Optimization Experiment Analysis

The back-end of SLAM in this paper refers to the optimization of the pose map with loop constraints. For this part, a comparative experiment was set up to examine two aspects: the optimization degree of the algorithm to the trajectory of the pose map and the time consumption of the loop frame detection.

Regarding a comparative experiment of in situ pose map trajectory, we compared the effect before and after the optimization of the back-end pose map from two perspectives of the whole process of three-dimensional trajectory and of height change. In Fig. 12, the 3D trajectory comparison effect map of the pose map before and after the optimization of the rear end is shown. The coordinate point (0, 0, 0) in the figure is the starting point of the experiment, and the vehicle stopped at the entrance of the college after a circle around the college. It can be seen from the figure that the red and blue tracks basically coincided at the beginning of the journey. When entering the first corner, the errors of the two kinds of trajectories began to appear. The later trajectory for back-end optimization appeared more and more drift. When the red curve was close to the starting point of the experiment, a good closed-loop was formed for the red curve, while the error accumulation of the blue track was caused by the distance. There was a big deviation in the red trajectory.

On the other hand, the comparison effect of height error before and after optimization corresponding to the three-dimensional trajectory is shown in Fig. 13. The route of the experimental scene in this paper basically had no large height fluctuation. It can be seen from the figure that the height after the rear end optimization always fluctuated slightly up and down based on the starting position, while the height before the rear end optimization fluctuated greatly due to the impact of vehicle bumps and other factors.

In addition, in order to reflect the advantages of this algorithm in the back-end loop optimization, the running effect of large-scale environmental offline data sets containing loopback was compared with whether to carry out graph optimization and loop detection. As shown in Fig. 14, the trajectory had obvious drift without loop detection for pose optimization.

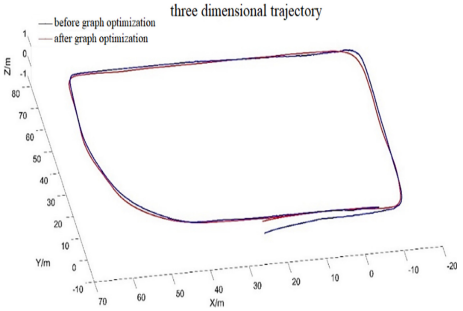


Fig. 12. Trajectory comparison before and after optimization. (Color figure online)

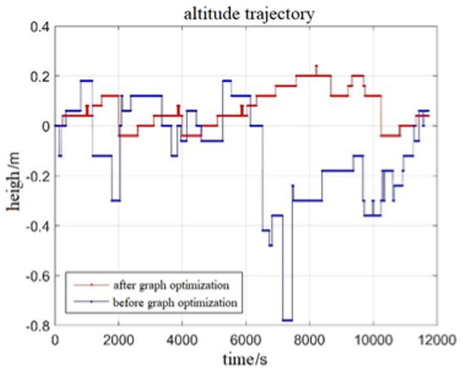
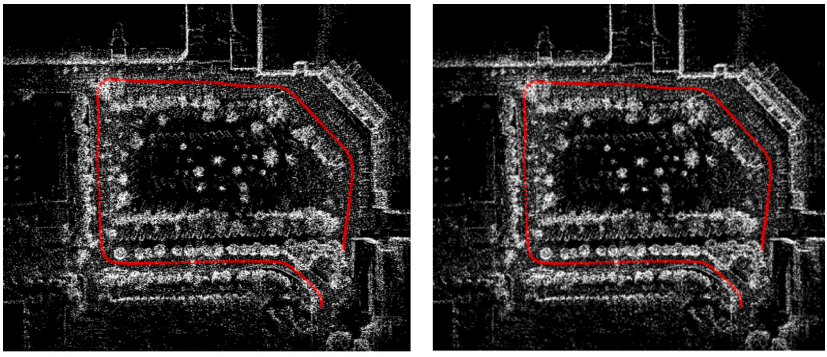


Fig. 13. Comparison of height error before and after rear end optimization.



(a) (b)

Fig. 14. Large scale environmental mapping effect including loop: (a) the Simultaneous Localization and Mapping (SLAM) algorithm presented in this paper; (b) SLAM algorithm without loop detection and graph optimization.

In this paper, the number of feature points is taken as the constraint condition to carry out the prescreening work, which greatly reduces the time consumption of the loop frame determination. In this regard, experiments were set to verify the optimization effect of the algorithm's time consumption by the number of feature points. The experimental comparison is shown in Fig. 15. Experimental results showed that when the vehicle passed through the straight road to a new environment, the time consumption of the loop frame detection was greatly reduced, the algorithm time consumption increased within the distance of less than 50 m at the beginning of the experiment, and the remaining parts significantly reduced the time consumption of the loop frame determination algorithm.

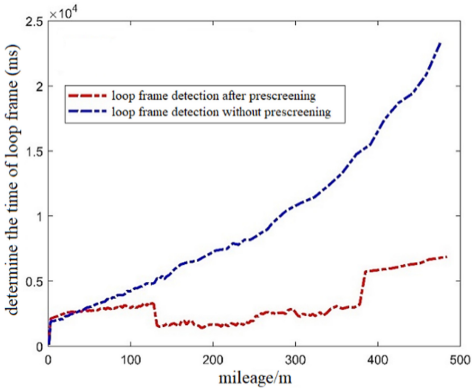


Fig. 15. Loop frame determination process time comparison.

6 Conclusions

The map and location technologies that intelligent driving vehicles depend on is one of the hot topics of research. In this paper, considering the advantages of lidar data stability and good development prospects of laser SLAM, the analysis and processing of raw data, front-end processing and back-end optimization of lidar SLAM were studied. In this paper, a boundary line feature extraction method was proposed to supplement the curvature-based feature points, so as to enhance their anti-interference ability to noise. Regarding graph nodes, reasonable sparse sampling was carried out on the nodes of the pose map through the environmental characteristics and trajectory characteristics of the intelligent driving vehicle. Regarding the constraint edge between nodes, the loop edge was determined through the determination of a candidate loop frame, the accurate determination of a loop frame and boundary verification. Finally, the back-end optimization algorithm was used to optimize the whole pose map to reduce the cumulative error caused by the large scale of the environment. The experimental results showed that the optimized matching algorithm was less affected by noise, and the loop frame detection optimization algorithm was reduced by 75%, which greatly reduced the time consumption of the loop frame determination algorithm and proves that the optimization algorithm in this paper is effective. The results provide ideas for the accurate mapping of intelligent vehicles in the process of driving, and improve the safety of vehicle driving.

References

1. Colak, H.E., Memisoglu, T., Erbas, Y.S., Bediroglu, S.: Hot spot analysis based on network spatial weights to determine spatial statistics of traffic accidents in Rize, Turkey. Arab. J. Geosci. **11**, 151 (2018)
2. Arif, F., Bayraktar, M.E.: Current practices of transportation infrastructure maintenance investment decision making in the United States. J. Transp. Eng. Part A. Syst. **144**, 4018021 (2018)
3. Pring-Mill, D.: Drone taxi service could impact urban infrastructure. Eng. News-record **280**, 49 (2018)
4. Gao, Y.B., Liu, S.F., Atia, M.M., Noureldin, A.: INS/GPS/LiDAR integrated navigation system for urban and indoor environments using hybrid scan matching algorithm. Sensors **15**, 23286–23302 (2015)
5. Nelson, G.: Car guys to Google: Move over - Mercedes' concept of a driverless car is no toy. Autom. News **89**, 1–42 (2015)
6. Goodwin, A.: Google reveals driverless car. Diesel Car: The UK's Leading Magazine for Diesel & Alternative Fuel Vehicles, pp. 64–65 (2014)
7. Crain, K.E.W.: Relax! This self-driving car is normal. Autom. News **92**, 46 (2018)
8. Ji, Z., Singh, S.: LOAM: lidar odometry and mapping in real-time. In: Robotics: Science and Systems Conference, vol. 6, pp. 12–19 (2019)
9. Dhiman, N.K., Deodhare, D.; Khemani, D.: Where am I? Creating spatial awareness in unmanned ground robots using SLAM: a survey. In: Sadhana: Academy Proceedings in Engineering Science, vol. 40, pp. 1385–1433 (2015)
10. Grisetti, G., Kummerle, R., Stachniss, C., et al.: A tutorial on graph-baesd SLAM. IEEE Intell. Transp. Syst. Mag. **2**(4), 31–43 (2010)
11. Hess, W., Kohler, D., Rapp, H., et al.: Real-time loop closure in 2D LIDAR ALAM. In: IEEE International Conference on Robotics and Automation. IEEE (2016)
12. Vysotska, S.: Exploiting building information from publicly available maps in graph-based SlAM. In: Intelligent Robots and Systems (2016)
13. Lenac, K., Cesic, J., Markovic, I., Petrovic, I.: Exactly sparse delayed state filter on Lie groups for long-term pose graph SLAM. Int. J. Robot. Res. **37**, 585–610 (2018)
14. Cheng, J.T., Kim, J., Shao, J.L., Zhang, W.H.: Robust linear pose graph-based SLAM. Robot. Autonom. Syst. **72**, 71–82 (2015)
15. Jaakkola, A., Hyyppa, J., Kaartinen, H.: Object classification and recognition from mobile laser scanning point clouds in a road environment. IEEE Trans. Geosci. Remote Sens. **54**, 1226–1239 (2016)
16. Macfaden, S.W., Pelletier, K.C., Royar, A.R.: An object-based system for LiDAR data fusion and feature extraction. Geocarto Int. **28**, 227–242 (2013)
17. Taie, S.A., Sayed, H.M., Abdelrahman, I.F.: Point clouds reduction model based on 3D feature extraction. Int. J. Embedded Syst. **11**, 78–83 (2019)
18. Marani, R., Renò, V., Nitti, M.: A modified iterative closest point algorithm for 3D point cloud registration. Comput.-Aided Civ. Infrastruct. Eng. **31**, 515–534 (2016)
19. Li, F., Hitchens, C., Stoddart, D.: A performance evaluation method to compare the multi-view point cloud data registration based on ICP algorithm and reference marker. J. Mod. Opt. **65**, 30–37 (2018)